
Charity Yaa Mansa Baidoo, University of Ghana, Ghana
Winfred Yaokumah, University of Ghana, Ghana*
Ebenezer Owusu, University of Ghana, Ghana

ABSTRACT

Estimating the energy and memory consumption of machine learning (ML) models for intrusion detection ensures efficient allocation of system resources. This study investigates the impact of supervised ML algorithms on the energy and memory consumption of intrusion detection systems. Experiments are conducted with seven ML algorithms and a proposed ensemble model, utilizing two intrusion detection datasets. Pearson correlation coefficient (PCC) and Spearman correlation coefficient are employed for the selection of optimum features. Regarding energy consumption, the findings reveal that the PCC with the UNSW-NB15 dataset uses the least amount of DRAM and CPU power. For ML methods, SVM utilizes the highest energy for both feature selection methods and datasets. Concerning memory consumption, the results show that decision tree uses the most current memory with PCC on the UNSW-NB15. The proposed ensemble model demonstrates the highest performance. These findings offer practical guidelines to ML experts when choosing the optimum model with the most efficient utilization of energy and memory.

KEYWORDS

Energy Consumption, Intrusion Detection, Machine Learning, Memory Consumption, Pearson Correlation Coefficient, Spearman Correlation Coefficient

INTRODUCTION

An intrusion detection system (IDS) provides active network security protection mechanisms that detect network anomalies to mitigate cybercriminal activities (Liang et al., 2019). An IDS keeps track of all incoming and outgoing network traffic to detect malicious packets (Liu & Lang, 2019; Rajasekaran, 2020; Taher et al., 2019). However, the basic operation of IDS through packet inspection and analysis
burdens packet routing (Migliardi & Merlo, 2013). This leads to high memory and energy consumption, thereby reducing the reliability of the network traffic (Xia et al., 2015). The active exchanges of data between nodes on the network also degrade network performance (Manthira & Rajeswari, 2013). Thus, although intrusion detection systems offer predictive attack detection, conducting deep packet inspection increases the utilization of computing resources at the network element level (Baddar et al., 2018). Therefore, in the design of IDS, consideration should be made not only of the typologies (host-based IDS or network-based IDS) and the identification methods (anomaly-based IDS or signature-based IDS) but also of the algorithms that perform the packet inspection and analysis. These algorithms can have significant impacts on the overall network by consuming considerable amounts of energy and memory. This may decrease network Quality of Service (QoS). For instance, when the IDS consumes a large number of resources, the system will run out of resources and packet loss may occur. Eventually the network cannot provide services to the existing network traffic. Even with the introduction of large-scale computing devices, the extensive use of computer services can cause high energy misuse (Jiang & Xu, 2017).

In most computing systems, the central processing unit (CPU) and main memory are the two major sources of high energy consumption. Dayarathna et al. (2016) estimates that CPU consumes about 30%–60% of energy while the main memory uses 28%–40%. Thus, the CPU and memory usage are crucial performance variables for assessing the performance of IDS. Currently, much attention is focused on the architecture and deployment of low-power consumption systems for efficient energy performance of security systems (Tsikoudis et al., 2016). However, the energy and memory requirements of the algorithms implemented on these systems are still largely unchanged (Rashid et al., 2015). Accordingly, reducing the power consumption of network security solutions such as network intrusion detection systems (NIDS) decreases the operating costs of the devices (Tsikoudis et al., 2016). Besides, with machine learning algorithms analyzing network traffic, errors may be caused by memory depletion. This results in massive wastage of computational power as well as a substantial drop in process performance (Gao & Lin, 2020). Consequently, the memory and energy consumption of computing systems have become increasingly important (Zhang et al., 2021).

In a previous work, García-Martín et al. (2019) asserts that there is an insufficient knowledge in the present methods to estimate energy consumption in machine learning frameworks. Subsequently, Kesrouani et al. (2020) develop a software-based model that determines energy consumption of computing devices. Further, Berthou et al. (2020) propose a method for estimating the power consumption of peripheral devices for a low-power embedded micro-controllers. While memory consumption is critical in determining the performance of systems, these works do not estimate the energy used by the DRAM. Consequently, Wu et al. (2020) experiment to determine the actual memory usage of machine learning algorithms, while Corda et al. (2021) use an ensemble machine learning model to predict memory utilization. Here, the authors determine memory usage of system processes and near-memory computing offloading, but not in connection to CPU and DRAM memory usage. Further, Katsaragakis et al. (2020) provide memory optimization techniques for machine learning applications, focusing only on Python program only.

Therefore, this study aims at investigating the impact of supervised machine learning algorithms on energy and memory consumption of IDS. Few studies examine energy efficiencies and memory consumption of machine learning algorithms (Elmasri et al., 2020; Maseer et al., 2021). Thus, there is a significant knowledge gap in selecting a specific machine learning IDS algorithm, based on its performance in terms of energy consumption and memory usage. Ascertaining the energy and memory consumption of a machine learning algorithm is critical in many ways, including determining which hardware will be required for system optimization. This current work differs from existing techniques. Besides experimenting with SVM, Decision tree, Gradient Boosting, Logistic regression, Naïve Bayes, Random Forest, and K-nearest Neighbor algorithms, the study proposes an ensemble machine learning model (combining Random Forest and Gradient Boosting Decision Tree) that measures the overhead performance (energy consumption and memory utilization) of intrusion detection systems.
The feature selection uses two statistical-based techniques (Pearson correlation coefficient and Spearman correlation coefficient) with two prominent intrusion detection datasets (UNSW-NB15 and CICIDS2017) to reduce the dimension of the feature space and to increase the performance of the algorithms. The main contributions of this study are as follows:

- An ensemble learning model based on Random Forest and Gradient Boosting Decision Tree is proposed to improve the performance of intrusion detection systems.
- Energy consumption and memory usage are measured with seven state-of-the-art machine learning classifiers as well as the proposed ensemble learning method.
- Provides practical guidelines to machine-learning experts in the selection of optimum intrusion detection machine learning models for efficient energy consumption and memory utilization.

LITERATURE REVIEW

This section presents selected machine learning algorithms, reviews various applications of machine learning in intrusion detection systems, and discusses previous works pertaining to energy and memory consumption of computing devices and processes.

Machine Learning Methods

Several prominent machine learning methods with different features have been applied in intrusion detection systems. Naïve Bayes classifier determines the conditional probabilities for several classifications for each sample and the class with the highest probability is assigned to the sample (Liu et al., 2019). Unlike the Naïve Bayes, the Logistics Regression estimates the probabilities by employing a logistic function, commonly known as the sigmoid function (Sarker, 2021). But the Support Vector Machine maps the training data into a higher-dimensional space using a kernel function. It is primarily useful when the number of attributes is large and the number of data points is minimal (Khraisat, 2019). The Decision Tree method constructs a tree-structured model by employing split metrics and using the training data to train the model (Balamurugan & Kannan, 2016). The advantages of the Decision Tree are speed and accuracy (Zhang & Wu, 2019). Similarly, the Gradient Boosting Decision Tree (GBDT) is a boosting decision tree classifier that employs the additive model and the forward stepwise algorithm to implement the learning optimization process (Tian et al., 2018). The Random Forest algorithm integrates the attributes of multiple eigenvalues and the optimal values of training data are used to enhance the accuracy of the prediction (Dai et al., 2018). Altogether, the ensemble learning methods integrate multiple machine learning models to categorize patterns and enhance prediction accuracy (Jabbar et al., 2017).

Application of Machine Learning in IDS

Gharae and Hosseinivand (2017) proposed anomaly-based IDS using a genetic algorithm and a support vector machine (SVM) with a new feature selection approach based on a genetic method to decrease the dimension of the data and enhance true positive detection. KDD CUP 99 and UNSW-NB15 datasets were used to evaluate the experimental model. For KDD CUP 99, the results obtained with the GF-SVM model improved detection accuracy to 99.05 percent for normal traffic, 99.95 percent for DOS class, 99.06 percent for PROBE class, 98.25 percent for R2L, and 100 percent for U2R. For UNSW-NB15, the results obtained were 97.45 percent for Normal, 96.39 percent for Fuzzers, 91.55 percent for Reconnaissance, and 99.45 percent for Shellcode. Likewise, Jing and Chen (2019) evaluated SVM with a new scaling method using the UNSW-NB15 dataset. Both binary and multi-classification tests were carried out. The accuracy was 85.99% and the false positive rate (FPR) of 16.50% was obtained for the binary classification of the proposed system. The accuracy result of the multi-classification was 75.77% and 3.04% for FPR.
Srivastava et al. (2019) also used the UNSW-NB15 dataset and applied novel feature reduction-based machine learning algorithms to identify anomalous patterns. Decision Tree, Multinomial Logistic Regression, Multinomial Naive Bayes, and Random Forest were the machine learning algorithms used. The prediction accuracies were 71.25 percent for Multinomial Naive Bayes, 74.5 percent for Multinomial Logistic Regression, 85 percent for Random Forest, and 86.15 percent for Decision Tree, which was the maximum. Jabbar (2016) combined AD Tree (a supervised boosting algorithm) and pre-processing techniques to classify various types of attacks. NSL-KDD dataset was used in the experiment. The proposed approach outperformed the naïve Bayes model in terms of detection rate, accuracy, and lower false alarm rate. But, Waskle et al. (2020) suggested a method using the Principal Component Analysis (PCA) and the Random Forest Classification Algorithm to establish effective IDS. The PCA was used for dimensionality reduction of the dataset, and the random forest was used for the classification. A new machine learning technique was developed to forecast network intrusion based on a random forest and support vector machine. Random Forest and SVM were employed for the classification using the KDD CUP 99 dataset. The result indicated that the 14 features chosen obtained a higher rate of detection.

Similarly, Yang et al. (2020) suggested SAVAER-DNN, a new network intrusion detection model that detects both known and unknown attacks while also improving the detection rate of low-frequency attacks. SAVAER is a regularized supervised variational auto-encoder that learns the latent representation of the original data using WGAN-GP. To assess the proposed model’s accuracy, the NSL-KDD (KDDTest+), NSL-KDD (KDDTest-21), and UNSW-NB15 datasets were used. In the experiments, the proposed solution outperformed eight popular classification models in terms of prediction accuracy including K-Nearest Neighbor, Gaussian Naive Bayes, Decision Tree, Logistic Regression, Support Vector Machine, Random Forest, Deep Belief Network, and Deep Neural Network. Also, Al-Yaseen et al. (2017) analyzed a multi-level hybrid intrusion detection model using SVM and extreme learning machines to increase the effectiveness of known and unknown attack detection. The study used the KDD CUP 99 to evaluate the proposed technique and the results compared to other models exhibited a higher detection and accuracy of 95.75% rate and with a false alarm rate of 1.87%.

**Energy Consumption**

Energy consumption is the amount of energy utilized in the execution of a process (Taborda et al., 2015). The quest to optimize modern computing platforms has resulted in a significant increase in energy consumption, resulting in increased running costs and failure rates (Sundriyal & Sosonkina, 2013). Therefore, assessing the power usage of computing equipment is critical for improving energy efficiency and adherence to power budgets. Rouhani et. (2016) proposed DeLight, which is an automated deep learning framework that allows for efficient training and modeling of Deep Neural Network (DNN) energy consumption. The study modeled energy utilization in a distributed training context using fundamental arithmetic operations and communication of shared weights. Multiply-add (which is a function of the number of connections between neurons in two adjacent layers) and the activation function (which is scalar multiplication) were the two types of operations modeled. Micro-benchmarks on the CUDA cores of the Nvidia TK1 embedded platform were used to determine the coefficients for modeling these operations. On vast computer systems, RAPL gives accurate energy readings for CPUs and DRAM.

Similarly, Cai et al. (2017) suggested NeuralPower as a method for accurately estimating power, runtime, and energy consumption. NeuralPower uses sparse polynomial regression. In terms of accuracy, the framework exceeded the best current predictive model with an improvement of up to 68.5 percent. The study further evaluated prediction accuracy at the network level by estimating the runtime, power, and energy. In terms of runtime, power, and energy, NeuralPower achieved an average accuracy of 88.24%, 88.34%, and 97.21%, respectively. Also, Sarood et al. (2013) used the Running Average Power Limit (RAPL) to examine the prospect of enhancing an application execution time
efficiency by capping power while adding more nodes. The study employed distinct power limitations for both the CPU and memory subsystems to profile the powerful scaling of an application. The interpolation approach makes use of an application profile to optimize the number of nodes and power distribution between CPU and memory subsystems to reduce execution time. The experimental results closely match the model predictions, with speedups of more than 1.47X for all applications when compared to not using capping CPU and memory power. Similarly, Marcus et al. (2012) undertook a study that monitored the energy consumption of short code lines using the RAPL energy sensors found in modern Intel CPUs. The study investigated the granularity at which RAPL measurements can be made as well as the practical challenges that can be encountered on today’s complex CPUs.

Kesrouani et al. (2020) developed a model to estimate how much energy is consumed by a device while accounting for CPU usage. The model typically had a 1.25 percent error rate. The model was as used to analyze how various compilers, programming languages, and algorithms affect energy consumption. Despite estimating the energy used by the CPU, this work did not estimate the energy used by the DRAM. This study estimates the energy used by both the CPU and the DRAM memory and energy usage. Moreover, Berthou et al. (2020) performed a simulating to estimate the power consumption of peripheral devices for a low-power embedded micro-controller. The method was tested using the low-power MSP-EXP430FR5739 platform, which had a few peripherals and nonvolatile RAM for inconsistent processing demands. The power consumption of the peripherals had a 5% error rate. Unlike many other similar research that have been examined, this study analyzed peripheral power consumption.

Memory Consumption

The amount of memory used by a program during its operation is referred to as memory consumption (Hsieh & Chen, 2018). Wu et al. (2020) experimented to determine the actual memory usage of machine learning algorithms, including linear regression, ridge regression, Lasso, KNN, decision tree, random forest, and bagging. The experimental results showed that memory used by processes on the platform can be predicted fairly accurately (R² = 0.95). The linear regression, ridge regression, and lasso regression were the most effective. Here, the authors determined processes memory usage, but not in connection to CPU and DRAM memory usage. Also, Shirota et al. (2019) developed hybrid access to memory hierarchical control mechanism, which adaptively alternated between SCM-aware low power Aggressive Paging (AP) with a small Dynamic Random-Access Memory (DRAM) as cache and direct access to memory bus coupled byte-addressable SCM. Using an optimal control prediction model created by ML, the study proposed an auto-tuning framework that dynamically determined the optimal control and optimal DRAM size when AP is selected. Based on time-series performance data the hybrid access was found to produce a cost effective unified main memory that could handle a wide range of data access patterns while using DRAM of moderate size.

Dhalla (2020) employed native JSON parsers in five major programming languages (Java, Python, PHP, MS.NET Core, and JavaScript) to compare parsing speed and resource consumption. The experimental data used a Java technique that generated ten separate JSON files, each with an increasing number of key-value pairs as the level of JSON nesting increased. The result found that JavaScript parsed the JSON string more effectively than the other languages. Moreover, Cheng et al. (2015) compared the CPU and memory utilization of the rehabilitation game using the Java and Python programming languages. When the two languages were compared, the Python server utilized fewer resources than the Java server. Accordingly, the study recommended Python as particularly useful in the creation of back-end computing servers.

Katsaragakis et al. (2020) provided memory optimization techniques for machine learning applications that used Python. The proposed techniques aimed to reduce static memory allocation and promote dynamic memory management in order to maximize memory usage and execution latency. The results were examined using a biomedical application, and the outcome demonstrated
significant memory consumption and performance improvements, with a 64% reduction in memory space requirements and a 51% reduction in execution time.

Corda et al. (2021) developed a high-level framework called Near-Memory Computing Profiling and Offloading (NMPO). It used an ensemble machine learning model to predict whether Near-memory computing (NMC) offloading is appropriate. Wen et al. (2020) proposed a guided fuzzing memory consumption tool known as Memlock. The framework determined excessive memory consumption inputs as well as initiate uncontrolled memory consumption defects. In order to make the technique devoid of domain-specific knowledge, the fuzzing process was directed by memory usage data. Fourteen frequently used real-world programs were used to evaluate the MemLock. The test results demonstrated that MemLock significantly outperformed orthodox fuzzing methods, such as AFL, AFLfast, PerfFuzz, FairFuzz, Angora, and QSYM, in identifying memory consumption bugs.

**METHODOLOGY**

**Research Process and Simulation Phases**

Figure 1 presents the research process of the study. The research process begins with the selection of recent intrusion detection datasets, followed by data pre-processing and feature selection methods. A framework for the UNSW-NB15 and CICIDS2017 datasets was built using both the Pearson correlation coefficient and Spearman correlation coefficient as feature selection approaches for each classifier: Random Forest, SVM, Naive Bayes, Logistics Regression, Gradient Boosting, Decision Tree, and K-Nearest Neighbors. Python was used for the pre-processing of the data, implementation, evaluation of the IDS models, and the estimation of energy and memory consumption of the models.

**Description of Datasets**

The UNSW-NB15 and CICIDS2017 are commonly used datasets for performing experiments on models of intrusion detection. The UNSW-NB15 dataset was developed in the Cyber Range Lab of the Australian Center for Cyber Security (ACCS) using the IXIA PerfectStorm tool (Moustafa & Slay, 2015a). The dataset consists of 49 attributes. There are nine types of attacks and one type of normal packet: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. The training set consists of 175,341 records, whereas the testing set contains 82,332 records for both attack types and normal attacks. Likewise, the CICIDS2017 dataset was developed by Canadian Institute for Cybersecurity using the CICFlowMeter tool (Lashkari et al., 2017). There are 79 features in the CSV file. The dataset contains data from five days of attacks and normal data. It contains the most current common attacks namely: DoS, DDoS, Brute Force, Infiltration, Port Scan, Heartbleed Attack, and Botnet (Sharafaldin et al., 2018).

**Feature Selection**

Feature selection is to reduce data dimensionality and eliminate characteristics with low degrees of correlation (Yu et al., 2020). This study used two feature selection approaches - Pearson and Spearman correlation coefficients. For both feature selection methods, the threshold value of 0.70 was used (Feng & Fan, 2019). When the Pearson correlation coefficient was used to determine the most relevant characteristics in the UNSW-NB15 dataset, 16 features were found as strongly correlated. These features were eliminated based on the threshold value. The remaining 23 features were used for the training of the models. In the same way, when the Spearman correlation coefficient was applied for the same dataset, 17 features were identified as highly correlated and were removed. Similarly, for the CICIDS2017 dataset which had 79 features, when Pearson Correlation Coefficient was applied, 36 features were identified as highly correlated. The Spearman Correlation coefficient identified 47 as highly correlated. Table 1 summarizes the results of the feature selection. The selected features were utilized in the training of the model.
Figure 1. Research Process
A hyperparameter setting procedure was performed to find the optimal collection of hyperparameters for reducing a loss function and enhancing the outcome. For both the ensemble and the seven other classifiers employed, the RandomSearchCv module in the scikit-learn package was used to speed up the process of hyperparameter tailoring and for producing efficient classifiers. For the various classifiers, the following functions were used:

a) **Kernel Function**: Sigmoid, polynomial, sigmoid, radial basis function (RBF), and linear were the kernels for SVM. The efficiency of an algorithm, as well as the categorization of classes, was determined by the preference kernel. RBF function was used because it gave the highest within a minimum time. RBF is often the most utilized kernel (Jayasumana et al., 2015).

b) **The Number of Trees (N Estimators)**: A random forest is a collection of trees that are grouped. The efficiency of a model is proportional to the number of trees used (Bingzhen, 2020). For both Random Forest and Gradient boosting Decision Tree, the RandomSearchCv module was used to find the best $n$ estimators.

c) **Number of Neighbours** ($k$): For KNN, the number of neighbors to search to produce a prediction, is denoted in the value of $k$.

### Model Training

The Scikit-learn train test split was used to split the datasets. To find the most split efficiency, several dataset split techniques were employed. The efficiency of intrusion detection was evaluated using an average of 20 runs to assess classification techniques. This comprised 50% training and 50% testing, 60% training and 40% testing, 70% training and 30% testing, 80% training and 20% testing, and 90% training and 10% testing. The 80% for training and 20% for testing choice proved to be the most effective for both datasets.

### Overhead Performance Measurement

Two overhead performance metrics are applied in the experiments - Energy Consumption Measurement and Memory Consumption measurement. Real-time energy usage can be determined in one of two ways: As a direct method that uses circuit-based mechanisms such as current sensing resistors, and an indirect approach that employs a power model that applies a collection of activity counters and predefined weights (David et al., 2010). Using the pyRAPL module, the indirect method is used in this experiment. In systems, CPUs have predominantly been seen to be the most power-hungry component (Acar et al., 2016; Salem, 2017). As a result, the focus was on determining the CPU’s energy usage while executing the algorithms. The pyRAPL module was built to run several times, resulting in a total of 20 runs, which aids in determining the actual CPU energy consumption. Following that, the average energy usage of all runs was calculated, with the units of measurement in micro-Joules. For memory usage, the Python Memory Profiler module was used. This is a Python module for measuring a process’s memory usage as well as line-by-line memory consumption analysis. The memory usage of code was measured with the tracemalloc module (tracemalloc.get traced memory() function). This

### Table 1. Feature Selection with Pearson and Spearman Correlation

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Feature Selection Method</th>
<th>Input Features</th>
<th>Features Removed ($\geq 0.7$)</th>
<th>Selected Features ($&lt; 0.7$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNSW-NB15</td>
<td>Pearson Correlation</td>
<td>39</td>
<td>16</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>Spearman Correlation</td>
<td>39</td>
<td>17</td>
<td>22</td>
</tr>
<tr>
<td>CICIDS2017</td>
<td>Pearson Correlation</td>
<td>62</td>
<td>36</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Spearman Correlation</td>
<td>62</td>
<td>47</td>
<td>15</td>
</tr>
</tbody>
</table>
module is a debugging tool that allows for keeping track of how much memory is used. It produces an int, representing the memory use of the tracemalloc module, which is used to keep traces of memory blocks. It keeps track of all connected memory units’ current and maximum sizes.

EXPERIMENTAL RESULTS

The Ubuntu software version 20.04 operating system was used on a Del Intel® Core TM i7 – 6500U CPU @ 2.50GHz 2.60GHz, 16GB RAM for the experiments. The experiments were performed in two categories: (a) evaluating the energy consumption of machine learning models, and (c) estimating the memory consumption of machine learning models.

Energy Consumption of Machine Learning Models

The energy consumption of the CPU and DRAM are the two system components that are investigated. The experiments were performed using two intrusion detection datasets (UNSW-NB15 and CICIDS2017), Pearson correlation coefficients (PCC), and Spearman correlation coefficients (SCC). Figure 2 shows the CPU energy consumption results for Pearson and Spearman correlation coefficients on the UNSW-NB15. Observably, the result from the experiment shows that SVM consumed the most CPU energy of 9620.756 Joules on the UNSW-NB15 when Spearman correlation feature selection was employed. Moreover, the energy utilized by the DRAM for all models on the UNSW-NB15 dataset was evaluated and the results are presented in Figure 3. Again, SVM used the highest DRAM energy of 2136.178 Joules.

Similarly, the CPU energy consumption for the CICIDS2017 was measured for all models. Figure 4 depicts that SVM consumed the most CPU energy of 980610.111 Joules on the CICIDS2017. Furthermore, energy utilized by the DRAM for all ML models on the CICIDS2017 dataset was evaluated, and the results are shown in Figure 5. The experimental results show that SVM used the highest DRAM energy of 33686.488 Joules.

Memory Consumption of Machine Learning Models

This section measures the memory consumption (current and peak memory) of machine learning models for Pearson and Spearman correlation coefficients on UNSW-NB15 and CICIDS2017 datasets.

Figure 2. CPU energy consumption for PCC and SCC on the UNSW-NB15 dataset
Figure 6 displays the current memory (CM) consumption of each machine learning model on the UNSW-NB15 dataset. It can be observed in the figure that the Pearson correlation coefficient feature selection method on the Decision Tree classifier consumed the most current memory of 511.121MB. Also, Figure 7 shows the peak memory (PM) consumption of each ML model during the program execution. The proposed ensemble approach used the greatest peak memory of 2679.476 MB. Moreover, in Figure 8 the experimental results for the CICIDS2017 dataset show the current memory (CM) utilization of all the ML models. Here the results reveal that Logistics Regression used the greatest memory of 269.9811MB for both PCC and SCC. Finally, Figure 9 displays the peak memory (PM) consumption for the proposed ensemble approach on the CICIDS2017 dataset.
memory utilized by the ML models during the program execution for the CICIDS2017 dataset. Here the results reveal that SVM used the most peak memory of 4063.305 MB.

**DISCUSSION OF RESULTS**

The study discussed the performance of the machine learning models, the energy consumption, and the memory utilization of the models. It further compared the results of the datasets and the feature selection techniques used.
When utilizing Spearman correlation coefficient feature selection on the UNSW-NB15 dataset, SVM recorded the most energy consumption by the CPU (9620.756 J), whereas Logistics Regression recorded the least energy consumption by the CPU (1735.151 J). Overall, the Spearman feature selection technique used the least amount of CPU power on average with the UNSW-NB15 dataset. As can be seen in Table 1, the Spearman method utilized fewer features (8) for classification, which may explain the resulting lower resource consumption. In comparison, the Pearson approach used approximately three times more features (23) for classification, resulting in higher energy and memory use for the UNSW-NB15 dataset.
Moreover, the energy consumed by the DRAM for each model using Pearson and Spearman correlation coefficients on the UNSW-NB15 dataset was examined. When using Spearman correlation coefficient feature selection, SVM required the greatest energy (2136.178 J) from the DRAM, whereas Logistics Regression consumed the least energy (473.116 J). The SVM’s intensive memory requirements can be attributed to the fact that it keeps an NxN kernel matrix, which can be huge if N is large. SVM also uses a linear combination of all support vectors during the prediction stage. Because the dataset used is large, more support vectors must be stored, which necessitates the usage of more memory and energy. As a result, using the SVM model in a low-memory system may not be recommended. In comparison to SVM, the training of a Logistic Regression model needs a simple establishment of w (Normal) and b (y-intercept) parameters to separate both classes, which serves as the classification decision boundary, and hence may consume less energy. On average, Spearman feature selection approaches consumed the least amount of DRAM power. Altogether, the result shows that the CPU consumed the most energy for all models in the UNSW-NB15 dataset. Thus, the findings of this experiment confirm that the CPU is the most power-demanding component of computing processes (David et al., 2010).

Likewise, the study investigated the energy required by the CPU for each model using Pearson and Spearman correlation coefficients on the CICIDS2017 dataset. Using Pearson correlation coefficient feature selection, SVM utilized the highest energy of (80610.111 J), whereas Naive Bayes consumed the least energy (12387.660 J). On average, the Pearson feature selection approach utilized the least amount of CPU power. Also, the SVM demanded the most energy of (33686.488 J) from the RAM on the CICIDS2017 dataset while employing Pearson correlation coefficient feature selection, whereas Naive Bayes required the least (10036.219 J). The Naive Bayes algorithm’s lower energy usage can be attributed to the fact that it just computes the probability of each subclass given different input (x) values. As a result, the Naive Bayes classification algorithm may use fewer resources. Generally, on average, the Pearson feature selection techniques used the least amount of RAM power. Again, all the results demonstrate that the CPU uses the most energy for all models with the CICIDS 2017 dataset.
The energy consumption of the machine learning algorithm was profiled by Garcia-Martin et al. (2017). The authors created a methodology that analyzes the energy consumption of Very Fast Decision Trees (VFDT). The most energy-intensive processes were investigated. Following, the parameter adjustments were made to significantly reduce energy use. The results show that by addressing the VFDT’s most energy-intensive component, its energy use can be reduced by up to 74.3%. In contrast to their work, which only examined energy consumption of Decision Trees, our analysis measured the energy consumption and memory consumption based on CPU and DRAM of seven different algorithms.

Memory Consumption of ML Models

The memory used by the models revealed insightful findings. When Pearson correlation coefficient feature selection was applied to the UNSW-NB15 dataset, Decision Tree used the most current memory (511.121MB) while KNN used the least memory (17.154 MB). Because of the iterative nature of the algorithm, the Decision Tree technique used a large amount of memory. This can be caused by the enormous number of candidates splitting conditions associated with each numerical feature. Almost all models consumed less current memory when using Spearman feature selection methods. Also, the Pearson and Spearman correlation coefficients on the UNSW-NB15 dataset were investigated to determine the maximum (peak) memory required by each model. The suggested ensemble approach used the greatest peak memory of 2679.476 MB while utilizing Pearson correlation coefficient feature selection, whereas GBDT used the least peak memory of 665.641 MB. In as much as the proposed ensemble method gave the maximum accuracy, one should be able to afford their extra overhead such as the large memory footprint of such models. According to Duo et al. (2017), because dynamic random-access memory (DRAM) requires a constant supply of energy to store its data, excessive memory usage leads to high energy consumption.

On the CICIDS2017 dataset, the Logistics Regression used the greatest current memory (269.9811MB), utilizing Pearson correlation coefficient feature selection, whereas the Decision Tree used the least (2.949 MB). Moreover, when utilizing Pearson feature selection approaches, almost all models use less current memory on average. Additionally, on the CICIDS 2017 dataset, while using Spearman correlation coefficient feature selection, the SVM used the most peak memory (4063.305 MB), but the GBDT used the least peak memory (665.641 MB). Almost all the models utilized less current RAM on average when using the Pearson feature selection technique.

Comparison With Related Works

Table 2 compares the ML techniques and their memory and energy consumption with related works. It can be observed from the table that the Random Forest performed moderately well across all feature selection methods and datasets used since it utilized relatively less memory and energy.

CONCLUSION

This study applied supervised machine learning methods to two benchmark intrusion detection datasets (UNSW-NB15 and CICIDS2017) using two filter-based feature selection methods (Pearson and Spearman correlation coefficients) to determine their performance accuracy and resource utilization. A series of experiments were performed to investigate the overhead metric (energy and memory consumption) of SVM, DT, NB, LR, RF, GBDT, and KNN, as well as a proposed ensemble learning classification model by merging the top two performing models (RF and GBDT). For the UNSW-NB15 dataset with the SCC feature selection method, SVM consumed the most DRAM energy whereas Logistics Regression consumed the least energy. On average, the PCC feature selection approach utilized the least amount of CPU power. For CICIDS2017 with PCC feature selection, SVM again utilized the highest energy whereas Naive Bayes consumed the least energy. The SCC feature selection approach consumed the least amount of DRAM power. For the UNSW-NB15 dataset with
Table 2. Comparison of ML Techniques, Memory, and Energy Consumptions with Related works

<table>
<thead>
<tr>
<th>Reference</th>
<th>Feature Selection Method</th>
<th>Dataset</th>
<th>Classification Method</th>
<th>Accuracy (%)</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
<th>F1-Score (%)</th>
<th>Memory Usage (MB)</th>
<th>Energy Usage (J)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CPU DRAM</td>
<td></td>
</tr>
<tr>
<td>Jung and Chen (2019)</td>
<td>-</td>
<td>UNSW-NB15</td>
<td>SVM</td>
<td>85.99</td>
<td>50.99 (avg)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Srivastava et al. (2019)</td>
<td>Variance Threshold</td>
<td>UNSW-NB15</td>
<td>DT</td>
<td>86.15</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MLR</td>
<td>76.18</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MNB</td>
<td>71.25</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RF</td>
<td>85</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yang et al. (2019)</td>
<td>Information Gain</td>
<td>NSL-KDD</td>
<td>NB</td>
<td>70.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>KNN</td>
<td>75.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SVM</td>
<td>74.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RF</td>
<td>76.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Khammassi and Krichen (2017)</td>
<td>GA-LR</td>
<td>UNSW-NB15</td>
<td>DT</td>
<td>81.42</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Moostafa and Slay (2016)</td>
<td>GR PCC</td>
<td>UNSW-NB15</td>
<td>NB</td>
<td>82.07</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DT</td>
<td>85.56</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ANN</td>
<td>81.34</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LR</td>
<td>83.15</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EM</td>
<td>78.47</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aksu &amp; Aydin (2018)</td>
<td>-</td>
<td>CICIDS 2017</td>
<td>SVM</td>
<td>69.79</td>
<td>80</td>
<td>70</td>
<td>65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Yousefi-azar et al. (2017)</td>
<td>Auto-encoder</td>
<td>NSL-KDD</td>
<td>GNB</td>
<td>83.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>KNN</td>
<td>75</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SVM</td>
<td>74.73</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>XGBOOST</td>
<td>78</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Taher et al. (2019)</td>
<td>Chi Square</td>
<td>NSL-KDD</td>
<td>SVM</td>
<td>82.34</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>This study</td>
<td>PCC</td>
<td>UNSW-NB15</td>
<td>RF</td>
<td>86.18</td>
<td>86.91</td>
<td>86.18</td>
<td>85.2</td>
<td>17.486</td>
<td>665.646</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NB</td>
<td>61.08</td>
<td>85.29</td>
<td>61.8</td>
<td>62.96</td>
<td>20.256</td>
<td>667.49</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GBDT</td>
<td>86.24</td>
<td>87.23</td>
<td>86.24</td>
<td>83.29</td>
<td>17.357</td>
<td>665.641</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LR</td>
<td>82.21</td>
<td>79.42</td>
<td>82.21</td>
<td>79.25</td>
<td>55.737</td>
<td>665.643</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>KNN</td>
<td>85.32</td>
<td>84.51</td>
<td>85.81</td>
<td>85.32</td>
<td>17.437</td>
<td>2676.628</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SVM</td>
<td>83.82</td>
<td>85.81</td>
<td>83.82</td>
<td>80.53</td>
<td>17.667</td>
<td>665.73</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DT</td>
<td>80.66</td>
<td>80.66</td>
<td>80.66</td>
<td>78.19</td>
<td>511.121</td>
<td>1159.499</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Proposed Ensembler</td>
<td>86.8</td>
<td>88</td>
<td>86.8</td>
<td>86.40</td>
<td>19.251</td>
<td>2679.476</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CICIDS 2017</td>
<td>RF</td>
<td>80.79</td>
<td>79.86</td>
<td>80.78</td>
<td>79.3</td>
<td>17.372</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NB</td>
<td>28.1</td>
<td>68.88</td>
<td>28.1</td>
<td>30.84</td>
<td>17.311</td>
<td>665.641</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GBDT</td>
<td>79.48</td>
<td>79.87</td>
<td>79.48</td>
<td>76.96</td>
<td>17.322</td>
<td>665.641</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LR</td>
<td>66.72</td>
<td>62.94</td>
<td>66.72</td>
<td>64.53</td>
<td>17.316</td>
<td>665.641</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>KNN</td>
<td>79.47</td>
<td>78.72</td>
<td>79.47</td>
<td>78.93</td>
<td>17.154</td>
<td>665.478</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SVM</td>
<td>75.4</td>
<td>80.73</td>
<td>75.4</td>
<td>72.9</td>
<td>17.313</td>
<td>665.639</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DT</td>
<td>73.56</td>
<td>76.47</td>
<td>73.56</td>
<td>71.87</td>
<td>17.311</td>
<td>665.638</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Proposed Ensembler</td>
<td>80.81</td>
<td>80.29</td>
<td>80.81</td>
<td>79.78</td>
<td>17.342</td>
<td>665.641</td>
</tr>
</tbody>
</table>

continued on following page
PCC, the Decision Tree used the most current memory, the proposed ensemble approach used the greatest peak memory while KNN used the least current memory. Overall, all the models utilize less current RAM on average when using the PCC technique with CICIDS 2017 dataset.

Generally, the proposed ensemble technique provided the highest performance, but it is computationally intensive to deploy on a low-memory system because the model required a significant amount of memory and energy. Moreover, although Naive Bayes consumed the least amount of energy and memory, it had the worst performance of all the models. Among all models, SVM used the most energy. These findings offer practical guidelines to machine learning experts when choosing the optimum ML model with higher performance accuracy and the most efficient utilization of system resources to avoid bottlenecks on low-powered and memory computing devices such as IoT appliances. Theoretically, this study provided a method for measuring energy and memory consumption of ML techniques for intrusion detection. Though there are other overhead metrics, this current study investigated only energy and memory usage. Future work will investigate latency and throughput in similar experimental settings. Latency could cause network overhead and make detection of malicious activities slow and throughput might lengthen the waiting time of the network traffic.
REFERENCES


Dhalla, H. K. (2020). A Performance Analysis of Native JSON Parsers in Java, Python, MS.NET Core, JavaScript, and PHP. 2020 16th International Conference on Network and Service Management(CNSM), (pp. 1-5). IEEE.


