





Article

Empirical Analysis of Data Streaming and Batch Learning Models for Network Intrusion Detection

Kayode S. Adewole ¹, Taofeekat T. Salau-Ibrahim ², Agbotiname Lucky Imoize ^{3,4,*}, Idowu Dauda Oladipo ¹, Muyideen AbdulRaheem ¹, Joseph Bamidele Awotunde ¹, Abdullateef O. Balogun ^{1,5}, Rafiu Mope Isiaka ⁶ and Taye Oladele Aro ⁷

¹ Department of Computer Science, Faculty of Information and Communication Sciences, University of Ilorin, Ilorin 240003, Nigeria

² Department of Computer Science, Al-Hikmah University Ilorin, Ilorin 240281, Nigeria

³ Department of Electrical and Electronics Engineering, Faculty of Engineering, University of Lagos, Akoka, Lagos 100213, Nigeria

⁴ Department of Electrical Engineering and Information Technology, Institute of Digital Communication, Ruhr University, 44801 Bochum, Germany

⁵ Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Bandar Seri Iskandar 32610, Perak, Malaysia

⁶ Department of Computer Science, Kwara State University, Malete 241103, Nigeria

⁷ Department of Computer Science, Confluence University of Science and Technology, Osara 264103, Nigeria

* Correspondence: aimoize@unilag.edu.ng



Citation: Adewole, K.S.; Salau-Ibrahim, T.T.; Imoize, A.L.; Oladipo, I.D.; AbdulRaheem, M.; Awotunde, J.B.; Balogun, A.O.; Isiaka, R.M.; Aro, T.O. Empirical Analysis of Data Streaming and Batch Learning Models for Network Intrusion Detection. *Electronics* **2022**, *11*, 3109. <https://doi.org/10.3390/electronics11193109>

Academic Editors: Nurul I. Sarkar and Juan-Carlos Cano

Received: 9 September 2022

Accepted: 26 September 2022

Published: 28 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Abstract: Network intrusion, such as denial of service, probing attacks, and phishing, comprises some of the complex threats that have put the online community at risk. The increase in the number of these attacks has given rise to a serious interest in the research community to curb the menace. One of the research efforts is to have an intrusion detection mechanism in place. Batch learning and data streaming are approaches used for processing the huge amount of data required for proper intrusion detection. Batch learning, despite its advantages, has been faulted for poor scalability due to the constant re-training of new training instances. Hence, this paper seeks to conduct a comparative study using selected batch learning and data streaming algorithms. The batch learning and data streaming algorithms considered are J48, projective adaptive resonance theory (PART), Hoeffding tree (HT) and OzaBagAdwin (OBA). Furthermore, binary and multiclass classification problems are considered for the tested algorithms. Experimental results show that data streaming algorithms achieved considerably higher performance in binary classification problems when compared with batch learning algorithms. Specifically, binary classification produced J48 (94.73), PART (92.83), HT (98.38), and OBA (99.67), and multiclass classification produced J48 (87.66), PART (87.05), HT (71.98), OBA (82.80) based on accuracy. Hence, the use of data streaming algorithms to solve the scalability issue and allow real-time detection of network intrusion is highly recommended.

Keywords: data streaming; batch learning; network intrusion; projective adaptive resonance theory; J48; Hoeffding tree; OzaBagAdwin

1. Introduction

The continuous growth of technological innovations in the networking infrastructure, particularly cloud services and the Internet of Things (IoT), has spawned new networking infrastructure-based companies. It has caused the creation of enormous amounts of data that need to be meaningfully processed [1,2]. This growth has also given rise to malicious attacks on data or information passed across networked systems and devices. That is, hackers are paying attention to network devices due to the sensitive data available on these devices [3]. For instance, based on the Symantec Internet Security Threat Report, there have been approximately more than 5 billion zero-day attacks [4]. Additionally, Data Breach Statistics disclosed that more than 8 billion data have been taken or unauthorizedly

accessed by cyber criminals [5]. Based on this premise, the protection of these devices and the development of sophisticated cybersecurity solutions is imperative, as failure to do so may be deleterious. One of the various technologies used in cybersecurity is intrusion detection systems (IDS) [6].

IDS can be a software package or a hardware device that detects on network devices to preserve system security. IDS' primary goal is to identify potential security threats on vital networks and systems, and whatever information they track as well as notifying all violations committed by any unauthorized personnel or internal attacker [7]. In other words, IDS aims to detect various types of malicious threats early, which a conventional firewall cannot do. Concerning data sources utilized to identify anomalous activities, IDS can be divided into host-based and network-based IDS classes [8]. HIDS examines data originating from the host system and can identify attacks that do not require network traffic. Conversely, NIDS tracks network traffic retrieved from a network and can analyze many systems or devices connected to a network. In addition, NIDS can track external malicious behaviors that may be triggered by an external threat at an early stage before the risks propagate to other attached systems [9].

Consequently, due to the amount of data travelling through the networks, NIDS have a restricted capacity to analyze all data adequately [10]. To mitigate this drawback, machine learning (ML) models often referred to as ML-based IDS have been introduced by researchers and cybersecurity analysts as a viable solution. However, studies have shown that the performance of ML models is chiefly dependent on the nature and quality of data used for developing such ML methods. In the context of network systems, the generated data are either produced in batches or as a continuous flow of data (data stream). As for the data generated in batches, batch learning models are deployed, which separate or classify the data based on their respective class labels. That is, batch learning models (BLM) require the use of training and testing datasets separately. Nonetheless, BLMs suffer from inadaptability to unknown attacks, scalability issues with huge data as well as constant retraining of the model. In the case of the data stream, data streaming models (DSM) that support instance learning based on a specified time window are utilized. DSM can handle a large amount of data as well as addressing the evolving nature of intrusion data [11]. This study is motivated by the need to address the growing nature of intrusion incidents which demands scalable approaches with comparable performance [11,12].

Justify by this motivation, it is imperative to comparatively analyze the performance and efficacy of BLM and DSM for intrusion detection. Specifically, an extensive empirical comparative performance analysis is conducted on selected BLM (decision tree (J48) and projective adaptive resonance theory (PART)) and DSM (Hoeffding tree (HT) and OzaBagAdwin (OBA)) methods with diverse computational characteristics. The investigated ML models' computational characteristics consist of tree-based classifiers (J48 and HT), a rule induction-based (PART) classifier, and ensemble (OBA) methods. The selected ML algorithms represent the state-of-the-art methods, which are widely used in the two domains. The findings from this study will validate the effectiveness and suitability of the investigated ML models in ML-based IDS based on binary and multi-class classification processes.

Therefore, this study seeks to answer the following research questions:

- i. How effective are the selected BLM implementations for detecting network intrusion detection?
- ii. How effective are the selected DSM implementations for detecting network intrusion detection?
- iii. Can DSM achieve comparable performance compared with BLM?
- iv. How effective is the performance of the experimented BLM and DSM compared with existing methods?

The following are the major contributions of this paper:

- Extension of the state-of-the-art studies in the domain of network intrusion detection by investigating the performance of BLM and DSM algorithms.

- Investigation of the performance of selected BLM and DSM algorithms based on binary and multi-class classification tasks.
- Extensive analysis to ascertain the effectiveness of the selected BLM and DSM algorithms.

The sections that follow are organized as follows: Section 2 focuses on studies that employed the use of batch learning and data streaming algorithms, and Section 3 discusses the methodological framework, batch learning and data streaming algorithms. Section 4 focuses on the dataset description and evaluation metrics employed in this study, while Section 5 compares the two methodologies and summarizes the findings of the experiments conducted. Finally, Section 6 presents the conclusion of the study and offers opportunities for future work.

2. Related Work

The use of digital technology has grown significantly in our everyday lives, making it necessary to protect data and systems from intrusions. Intruders have devised several novel intrusion strategies to gain access to data and information in transit and stores [13]. An intruder is often defined as a system, program, or human that tries and succeeds in infringing on or performing an activity on a computer network that is not legitimately authorized. There are internal and external intruders. The first refers to a set of people who have legal access to a system and are attempting an illegal action. The second set is those who do not have access to the system and attempt to encroach. To decide which packets are trusted and which are anomaly based, the collected packets are evaluated with the trusted database of known signatures by the IDS. These systems are also linked to the firewall, which monitors data traffic inside and between networks.

Unlawful conduct or traffic on a system or network might be considered an intrusion. A monitoring instrument or software program that monitors a network for harmful events or policy violations and generates reports is known as an intrusion detection system. Intrusion detection systems (IDS), according to the authors in [14], are viable solutions for issues of cybersecurity; however, they come with implementation obstacles. Anomaly-based IDS typically have a high rate of false positives (FP) and demand a lot of computing power.

To identify and guard against such malicious assaults, in [14], the authors presented a unique two-stage intelligent IDS. The two-stage architecture presented is based on machine learning techniques. The IDS utilizes the K-means clustering algorithm as the first stage of attack identification, and the second stage involves the use of the supervised learning algorithm to categorize such assaults and reduce the number of false positives. The application of this technique results in a computationally effective IDS that can detect and categorize threats with high accuracy and low false positives.

In IDS, researchers have used ML-based approaches to detect the fundamental distinctions between normal and anomalous data automatically and with great accuracy. It can identify unknown threats. To identify and describe ML-based and DL-based IDS, the authors of [15] suggest an IDS ontology in which data objects serve as the primary criterion for identification. The survey first defines the idea of IDSs and their classifications. Then, the machine learning techniques, metrics, and benchmark datasets that are often employed in IDSs are used. The taxonomy system is used as a baseline and for addressing major IDS concerns using ML- and DL-based approaches.

The authors in [16] offered a new approach for a NIDS that uses a genetic algorithm (GA)-based exhaustive search to select an improved feature subset, and fuzzy C-means clustering (FCM). FCM was used to compute additional features for the improved feature subset used by the proposed hybrid algorithm. By combining the GA with five-fold cross-validation to choose the CNN model structure, the method finds the bagging classifier and the CNN model as effective extractors. It validates the performance with the five-fold cross-validation, and the classifier receives the feature subsets generated via the deep learning CNN model. The high-quality feature set produced by the three-layered feature

construction using the GA, FCM, and CNN extractors significantly enhances the final detecting accuracy, further to a hybridized CNN and bagging BG learning methods.

In reference [17], the authors suggested network IDS based on predictive rule mining and GA. During evolution, the innovative notion of updating the crossover and mutation rates is used to keep a healthy balance between exploitation and exploration. To cover a wider variety of attacks, a different training set was employed. On the knowledge discovery process at the network security laboratory and data mining baseline dataset, the suggested technique is implemented. Using accuracy and detection rate measures, the results assess the efficacy of the network intrusion detection model. Both wired and wireless networks can use this approach.

ML is frequently utilized in the development of firewalls and IDSs to discover new types of assaults due to its proven outcomes. IDSs can identify intruders by analyzing network traffic going through them using machine learning techniques. In [18], the authors provided a unique fitness function as well as parameter adjustment for GA-based feature selection. The current study creates an improved GA-based feature selection approach that is evaluated on three benchmark network traffic datasets. A comparison of the typical feature selection approaches is also made. The results demonstrate that the technique improves accuracy.

Furthermore, the authors in [19] proposed a machine learning IDS with a GA for feature selection. The suggested system's effectiveness was assessed using the NSL-KDD dataset. In addition, the researcher used decision trees (DT), support vector machines (SVM), random forests (RF), extra-trees (ET), extreme gradient boosting (XGB), and naive Bayes (NB) in the modeling process. The results showed that employing the GA classifier improves the performance of the classifiers. Furthermore, the new ML techniques produced better outcomes compared to other techniques.

The authors in [20] presented NB-SVM as a new and effective IDS that combines SVM and NB feature embedding, which is a new data quality enhancement technique that focuses on using a deep learning approach and a bi-directional long short-term memory (LSTM) based IDS model. In experiments, the KDDCUP-99 and UNSW-NB15 datasets are utilized to test the developed system. For both the KDDCUP-99 and UNSW-NB15 datasets, the model utilizing bi-directional LSTM produced remarkable results with high accuracy. The experiment was replicated by changing the network's activation functions. SoftMax and ReLU produced outstanding results for both datasets. The results were compared to those obtained using state-of-the-art techniques. Similarly, the authors in [21] proposed IDS that is capable of reducing malicious events by combining three machine learning models DT, NB and ANN to form an ensemble classifier using AdaBoost. New statistical features were studied, and the experimental results on UNSW-NB15 and NIMS botnets datasets produced promising results.

One of the challenges with existing IDS solutions is performance degrading over time, as new attacks are surfacing. To cope with this challenge, researchers have proposed the model update method which allows incremental model updates to address the network traffic's changing behavior over time. Based on this, the authors of [22] proposed an improved IDS system based on data streaming that maintains classification accuracy and improves false-positive rates by up to 12% while rejecting only 8% of the instances without a model update. If regular model updates are performed, the proposed solution can improve the classification accuracy by up to 6% while rejecting only 2% of network events.

Reference [11] presents the performance of Hoeffding tree (HT), NB, and Ozabag DSM algorithms for website phishing detection. The authors established that Ozabag outperformed the other two algorithms in terms of accuracy, kappa and kappa temp evaluation metrics. The authors also noted that the training time of Ozabag algorithm is more than NB and HT on a dataset with large samples.

The authors of [23] proposed AB-HT, an algorithm based on incremental AdaBoost and Hoeffding tree. The algorithm performs ensemble incremental learning for IDSs. Without having to retrain the model using previous training data, the AB-HT algorithm can

identify new intrusions. As a result, it might lower the number of computational resources required for training the algorithm and keeping the system's effectiveness. According to the experimental findings, the suggested model required less time to train over time than the AdaBoost-DT model. Additionally, the AB-HT model could achieve a F1-score that is 18% better on average when compared with the HT and HATT techniques. The results demonstrate that the AB-HT model is promising for IDS research.

Similarly, the authors in [24] proposed a model to identify an intruder using the SVM classifier, focused on enhancing classifier performance when trained to recognize signs of unidentified attacks. Additionally, the study established and investigated the most popular classifiers utilized in the deployment of IDSs (Bayes and KNN). The study showed a better improvement when compared with some existing work in the areas of IDSs. ThunderSecure is a novel unsupervised IDS for 100G research networks that was proposed by authors in [25]. ThunderSecure uses multi-core processors and GPUs to construct an effective packet analysis and detection network. It takes real-time network data streams, extracts statistical and temporal information from them and sends them to an anomaly detection network of one class. Results revealed that ThunderSecure can distinguish between science data traffic, even after training. This made it almost guaranteed that unusual flows on the stream segment would be detected.

However, the major challenge associated with the batch learning models for network IDS is characterized by their lack of adaptability when new attacks are launched by intruders. Batch learning models need to be retrained on the new threat signatures to perform effectively and retain their initial generalization features. In addition, batch learning models are computationally expensive when exposed to a large number of datasets for training. Moreover, intrusion data are characterized by their continual and evolving nature, which requires a scalable system to address. Hence, this paper provides a comprehensive investigative study to compare the performance of BLM and DSM for network intrusion detection.

3. Materials and Methods

3.1. Proposed Framework

Figure 1 shows the proposed framework in this study for network intrusion detection. The network intrusion dataset is collected from a public repository which is subjected to both batch and stream learning models. Two algorithms are chosen for each of the learning paradigms. For BLM, J48 and PART algorithms are selected. Meanwhile, for DSM, Hoeffding tree (HT) and OzaBagAdwin (OBA) are selected. The results from the individual model are compared as well as across the two models (that is, both batch and streaming methods). The outcome of this comparison provides great insight into the performance of the models proposed in this study.

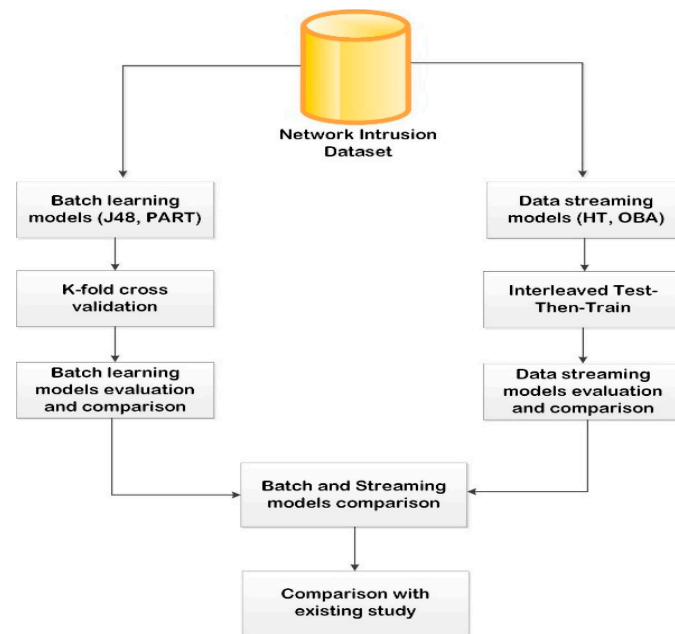


Figure 1. The experimental ML-IDS framework.

3.2. Batch Learning Models (BLM)

The process of training ML-based models in batches when the data are gathered over time is known as batch learning model (BLM), also referred to as offline learning [26]. The models are periodically trained in batches using the collected data. The model requires a substantial amount of time due to the volume of acquired data, and resources such as CPU, RAM, disk space, disk I/O, network I/O, etc. Only periodically, dependent on how well models developed using fresh data perform, models for batch learning are used in production [27,28].

The models must be retrained using the new dataset if BLM needs to learn about new data, and based on many factors, such as model effectiveness, and substituted with the model already in production [29,30]. Additionally, the entire batch learning procedure can be automated but has the drawback of taking a considerable amount of time as well as materials to retrain the model.

The system cannot learn gradually in batch learning; it must be trained to utilize all available data [31]. This process occurs offline because of the time and computing power required. The system is initially trained, subsequently going into operation, where it operates without having to learn anything new; it simply applies what it has learnt. Consequently, the entire process of training, testing, and launching an ML system is extremely simple to automate, as a result. Even batch learning systems may adjust to new circumstances [32].

In BLM settings, the model update is necessary over time to improve performance by simply updating the data and retraining a new version of the system. This technique is straightforward and frequently works well, although training with the entire amount of data can take several hours. If the system must respond to rapidly changing data, such as intrusion data, then a more reactive solution is required [33,34].

Furthermore, training on the entire collection of data necessitates a significant amount of computer resources. It may even be impossible to employ a BLM if the amount of data is enormous [14,35]. In all of these circumstances, though, using algorithms that can learn incrementally is a superior alternative [36,37].

3.2.1. Decision Tree (J48)

J48 is a decision tree-based algorithm that determines how selected attributes perform based on many instances. Among its characteristics that make it suitable for the present

work are the efficient error recession method, ease of calculating missing values and its use in a batch learning environment [38]. Algorithm 1 gives the details description of J48 classifier.

Algorithm 1: J48 Algorithm

```

1: Generate a node N as the root node;
2: If (T and C are in the same category)
   {
   leaf node = N;
   N is mark as class C;
   return N;
   }
3: For j = 1 to n
   { Compute Information gain(Ai);}
4: Assign ta as the testing attribute;
5: N.ta = attribute with the highest information gain;
6: if (N.ta = continuous)
   { compute threshold value; }
7: For (each T' in the T split)
8:     if (empty(T') = true)
   { assign child of N as leaf node;}
9:     else
   {child of N = dtree (T')}
10: compute the node N classification error rate;
11: return N;

```

3.2.2. Projective Adaptive Resonance Theory (PART)

PART is a rule induction based algorithm that infers rules using a partial decision tree technique [39]. The feature of PART that distinguish it from RIPPER and C4.5 is that it does not require the performance of a global optimization strategy to make correct rules. PART was used for batch learning in this study. Algorithm 2 gives the details description of PART classifier.

Algorithm 2: PART Algorithm

Inputs: Dataset S,

A1-> dimensions of input vectors,

A2 -> expected maximum clusters allowable at each clustering level

Initial parameters ρ_0 , ρ , σ , α , θ and e .

Output: RuleSet -> set of rules

1: Let $\rho = \rho_0$.

2: L1: WHILE (not stopping condition i.e., stable clusters not yet formed)

 FOREACH input vector in S do

 Compute k_{ij} for all A1 nodes V_i and committed A2 nodes V_j . If all A2 nodes are non-committed, go to L2

 Compute T_j for all committed A2 nodes V_j .

 L2: Select the best A2 node V_j . If no A2 node can be picked, add the input data into outlier O and then proceed with L1

 If the best is a committed node, calculate r_j , else goto L3

 If $r_j \geq \rho$, goto L3, else reset the best V_j and goto L2

 L3: Set the winner V_j as the committed and update the bottom-up and top-down weights for winner node V_j .

 ENDFOR

 FOREACH cluster C_j in A/2, compute the dimension associated with set D_j . Let $S = C_j$

$\rho = \rho + \rho h$, then, goto L1.

 For the outlier O, let $S = 0$, goto L1

 ENDWHILE

3.3. Data Stream Model (DSM)

The advancement of technology has created a tremendous volume of the data stream, bringing the task of deciphering the underlying patterns to the attention of academics and industry. To gain such vital knowledge, it is necessary to mine stream data in real time. Data stream mining is the practice of mining data in real time by storing, processing, and extracting useful knowledge that can aid decision making and the comprehension of extraordinary occurrences [40].

A data stream is an ongoing, dynamic sequence of data that often enters a system for archiving or processing. Think of a satellite-mounted remote sensor that is continuously gathering data. The data are enormous (terabytes in size), temporally organized, rapidly changing, and theoretically limitless. In the realm of data streams, these qualities pose significant challenges. Traditional OLAP and data mining approaches often necessitate several data scans, making them impractical for stream data applications [41].

Relational databases, for example, simple and structured data sets, transactional databases, and data warehouses, are suited for data mining techniques. Data grow rapidly in many complex forms, data that are semi-structured and unstructured, for example, spatial and temporal data, as well as multimedia and hypertext data, owing to the continuing and rapid advancement of sophisticated database systems, data collection technologies, and the World Wide Web. As a result, mining such complicated data becomes a crucial duty in the data mining world. Various methods have been presented in recent years to tackle the constraints of storing and analyzing fast and continuous streams of data [42].

Upon the advent of Big Data, in addition to the increase in data volume, nonetheless, the idea of data velocity also increases. We cannot assume that many potential real-world difficulties will involve dealing with a static collection of instances. Instead, they could arrive in waves, resulting in an infinite, vast and expanding dataset. It will grow in size over time, with new instances arriving in batches or one at a time. Data streams are a type of problem that presents numerous new hurdles to data mining approaches. The ability to continuously update the classification model with new data is essential, working within the temporal limits set by the frequency of occurrences, and coping with memory constraints [43].

In the future, the IoT, a vast network of physical devices that goes beyond traditional computer networks, will generate massive amounts of Big Data streams in real time. The ability to gain insights concealed in the large and expanding seas of data available is critical to the fulfilment of IoT [44]. The general process for the data mining stream is shown in Figure 2.

There are striking features of data stream mining. First, because of the high volume of data in the stream, it is impossible to store it for future analysis. The huge data stream contains a variety of data types, including texts, photos, and other types of data. Second, the fast pace at which these huge data are generated necessitates a high level of data mining efficiency. The arrival of a big data stream is dynamic and changes over time. As a result, issues such as measurement or calculation model flaws may have an impact. Third, massive streams of data are generated from a variety of heterogeneously scattered sources. As a result, mining requires sophisticated algorithms. The huge volume of stream data that allows processing and computing in one pass poses significant challenges [42]. Highlights of some distinguishing features of traditional data mining with data stream mining are shown in Table 1.

3.3.1. Hoeffding Tree (HT)

The Hoeffding tree (HT), also referred to as streaming decision tree induction, was derived from the Hoeffding bound used in tree induction. What distinguishes HT from other decision tree-based algorithms is that it gives a considerable level of confidence in the appropriate attribute to split the tree. Based on Hoeffding's bound, a confidence interval for the entropy estimation is expressed in Equation (1), where ϵ is the estimated value, n is the number of samples accumulated at the node, R is the random variable's scope,

and δ is the desired change such that the approximation is not within its estimated value. Algorithm 3 shows the procedures involved in HT. The characteristics of HT necessitate its selection as one of the algorithms investigated in this study. Algorithm 3 gives a detail description of the HT classifier.

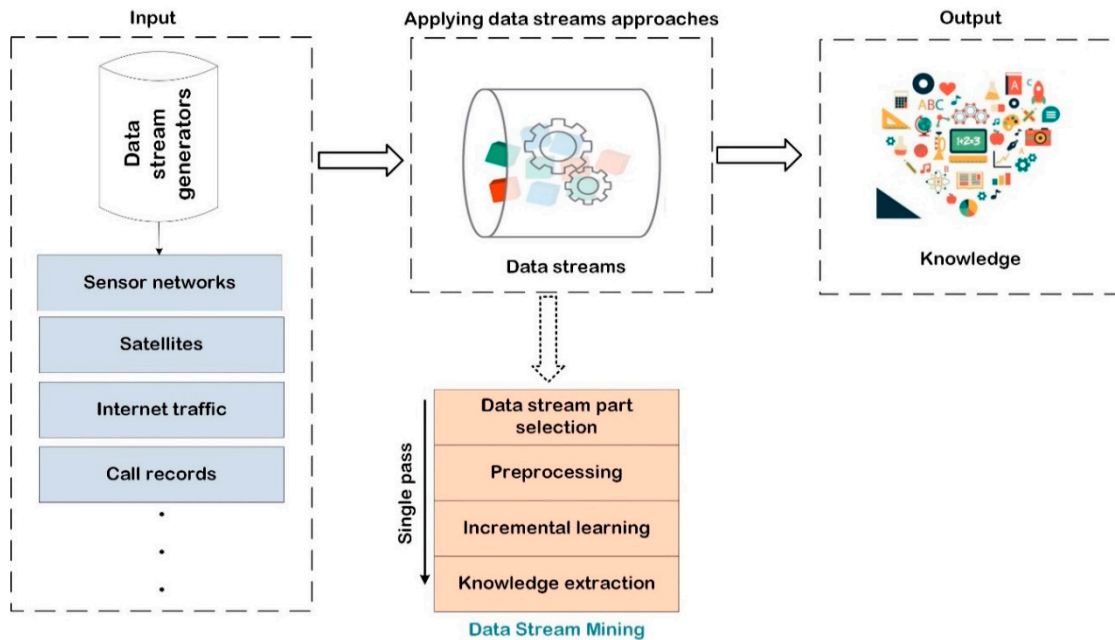


Figure 2. General data stream mining process.

Table 1. Differences between stream data mining and traditional data mining.

Feature	Data Stream Mining	Traditional Data Mining
Processing	Real-time samples	Offline every record
Storage	Not feasible	Feasible
Volume	Infinite	Finite
Data Generation	Dynamic	Static
Time	Only one pass	More time to access data
Data Type	Heterogeneous	Homogeneous
Result	Approximation	Accurate

Algorithm 3: HT Algorithm

Input: A stream of labeled data, confidence parameter δ

Output: Tree Model

- 1: Assign HT as a tree(root) having single leaf
- 2: Initialization: initialize n_{ijk} as counter at the root
- 3: for each data(a,b) in Stream do
- 4: HTGrow((a,b), HT, δ)

HTGrow((a,b), HT, δ)

- 1: apply sorting to (a,b) as leaf l based on HT
- 2: perform an update on n_{ijk} counter at leaf l
- 3: if data found at l so far do not belong to the same class
- 4: then calculate G for every attribute
- 5: if $G(\text{best-attribute}) - G(\text{second-best}) > \epsilon = \sqrt{\frac{R^2 \ln 1/\delta}{2n}}$
- 6: then on best attribute perform split-leaf
- 7: for every branch do
- 8: start a new leaf and initialize counts

3.3.2. OzaBagAdwin (OBA)

OzabagAdwin (OBA) is a machine learning algorithm based on ensemble learning in evolving data streams. Its striking features are its higher performance compared to a single classifier, and its ability to perform addition, removal and update on base classifiers whenever there is a drift. Algorithm 4 shows the OBA algorithm, where S denotes the data stream, M represents the number of base models, y represents the set of class labels, and x represents the features vector of instances. Algorithm 4 gives the details description of Ozabag classifier.

Algorithm 4: Ozabag Algorithm

Input: S (Stream data), M (Number of Bases

Model), Y (set of class labels), X (features vector)

Output: class label predicted based on majority vote

1: Initialization: setup M number of base classifier: h_1, h_2, \dots, h_M ;

2: while CanNext(S) do

3: $(x, y) = \text{NextInstance}(S)$;

4: for $m = 1, \dots, M$ do

5: $w \leftarrow \text{Poisson}(1)$;

6: train h_m with an instance using weight w ;

7: return overall class label based on the majority vote.

4. Experimental Setup

4.1. Dataset Description

The dataset used in this study is UNSW-NB15 [21], which is publicly available online. The dataset is labeled with 49 features. The instances are labelled as either normal or attack. Additionally, the dataset has another label that comprises the different categories of network attacks. This label was used to develop the multiclass models for the selected algorithms. The attack categories in this label are DoS, backdoor, reconnaissance, analysis, fuzzers, exploits, worm, generic, and shellcode.

4.2. Evaluation Metrics

In this section, the evaluation methods in terms of data feeding to the algorithms as well the performance of the algorithm are addressed. Hold-out and interleaved test-then-train methods were used for handling train and test datasets in a data streaming environment. However, interleaved test-then-train is adopted for this study because the accuracy can be incrementally updated. The performance metrics adopted are accuracy, kappa, ram hours, time, and memory.

Hold-out and cross-validation are some of the methods used for handling train and test datasets in a batch learning environment [44]. Cross-validation uses parameter k . This technique divides the whole dataset randomly based on k -fold. One-fold is used for training while other folds are used for testing the learning algorithm. In this study, 5-fold cross-validation was adopted because model overfitting is greatly avoided [45,46]. The performance metrics adopted are accuracy, F-measure, sensitivity, precision, false-positive and receiver operating characteristics (ROC). They are calculated using the following equations.

True positive is indicated by the number of intrusions that were successfully classified (TP), while the amount of correctly identified normal network traffic implies a true negative (TN). False positive (FP) is the number of true network traffic mistakenly labeled as intrusion, and false negative (FN) indicates how often intrusions were mistaken for normal network traffic. The observable and anticipated concurrence probabilities, P_o and P_e , are used in the Kappa statistic. The metrics are defined as follows:

1. *Accuracy:* This shows the percentage of classifications the model correctly predicts. It is computed utilizing

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

2. *Sensitivity*: This computes the percentage of actual positive cases that were predicted to be positive or (true positive). Another name for sensitivity is recall. Thus, it refers to the model's accuracy in detecting network intrusions in this context. Sensitivity is calculated mathematically by

$$True\ positive\ rate\ or\ Sensitivity = \frac{TP}{TP + FN} \quad (2)$$

3. *Precision*: This indicator shows the percentage of data occurrences that the model correctly predicts to be normal. It is calculated by dividing the total number of true positives by the sum of all true positives and false positives. Mathematically, it can be calculated by

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

4. *F-measure*: This is the precision and sensitivity weighted harmonic mean. The formula is as follows:

$$F - measure = \frac{2PR}{P + R} \quad (4)$$

5. *False-positive rate*: This gauges the percentage of lawful traffic that was incorrectly categorized as a network assault. The indicator is determined by

$$False\ positive\ Rate = \frac{FP}{FP + TN} \quad (5)$$

6. *Kappa statistics*: The observed and anticipated accuracy are correlated by the kappa statistic.

$$Kappa = \frac{P_o - P_e}{1 - P_e} \quad (6)$$

5. Results and Discussion

This section explains the results of the streaming and batch learning algorithms after being subjected to the network intrusion dataset. The results are two-fold: first for binary classification, and second for multiclass classification.

5.1. DSM Experimental Results Based on Binary Class Classification

For the streaming algorithms binary classification, OBA achieved 99.67% accuracy as compared to HT, which achieved a lower accuracy value of 98.38%. Similarly, the Kappa statistics for OBA was 99.33% while HT achieved 96.73%. This shows that within the streaming environment, OBA achieved considerably higher accuracy and kappa statistics for network intrusion detection based on a binary class classification problem. The time taken for OBA compared to HT is higher. The results of the experiments are depicted in Table 2.

Table 2. Hoeffding tree (HT) and OzaBagAdwin (OBA) binary classification results.

Metrics	HT Mean Value	OBA Mean Value
Accuracy	98.38	99.67
Kappa	96.73	99.33
Ram Hours	0.00	0.00
Time	0.91	4.08
Memory	0.00	0.00

5.2. DSM Experimental Results Based on Multi-Class Classification

For the experiment that considered multiclass classification, OBA achieved 82.80%, 76.14% and 12.86 sec for accuracy, kappa statistics and time respectively. HT on the other achieved 71.98%, 62.27% and 1.80 sec for accuracy, kappa statistics and time, respectively. This result shows that OBA outperforms HT in terms of accuracy and kappa statistics. However, the time taken for OBA to complete the classification is greater compared to HT, which is similar to the findings for binary classification problems. The results of the experiments are depicted in Table 3.

Table 3. Hoeffding tree (HT) and OzaBagAdwin (OBA) multiclass classification results.

Metrics	HT Mean Value	OBA Mean Value
Accuracy	71.98	82.80
Kappa	62.27	76.14
Ram Hours	0.00	0.00
Time	1.80	12.86
Memory	0.00	0.00

5.3. BLM Experimental Results Based on Binary Class Classification

For the batch learning algorithms binary classification, J48 and PART achieved 94.73% and 92.83% accuracy, respectively. This shows that within the batch learning environment, J48 outperformed PART in detecting network intrusions. The false-positive rate, which is also important in determining the strength of the intrusion detection system, was 5.60% and 7.10% for J48 and PART, respectively. This result also confirmed the superiority of J48 for detecting network intrusion for the binary classification problem when compared with the PART algorithm. The results of the experiments are depicted in Table 4.

Table 4. J48 and PART algorithms binary classification results.

Metrics	J48 Mean Value	PART Mean Value
Accuracy	94.73	92.83
Sensitivity	94.70	92.80
Precision	94.70	0.93
F-measure	0.947	0.93
ROC	0.982	0.98
False Positive Rate	0.056	0.071

5.4. BLM Experimental Results Based on Multi-Class Classification

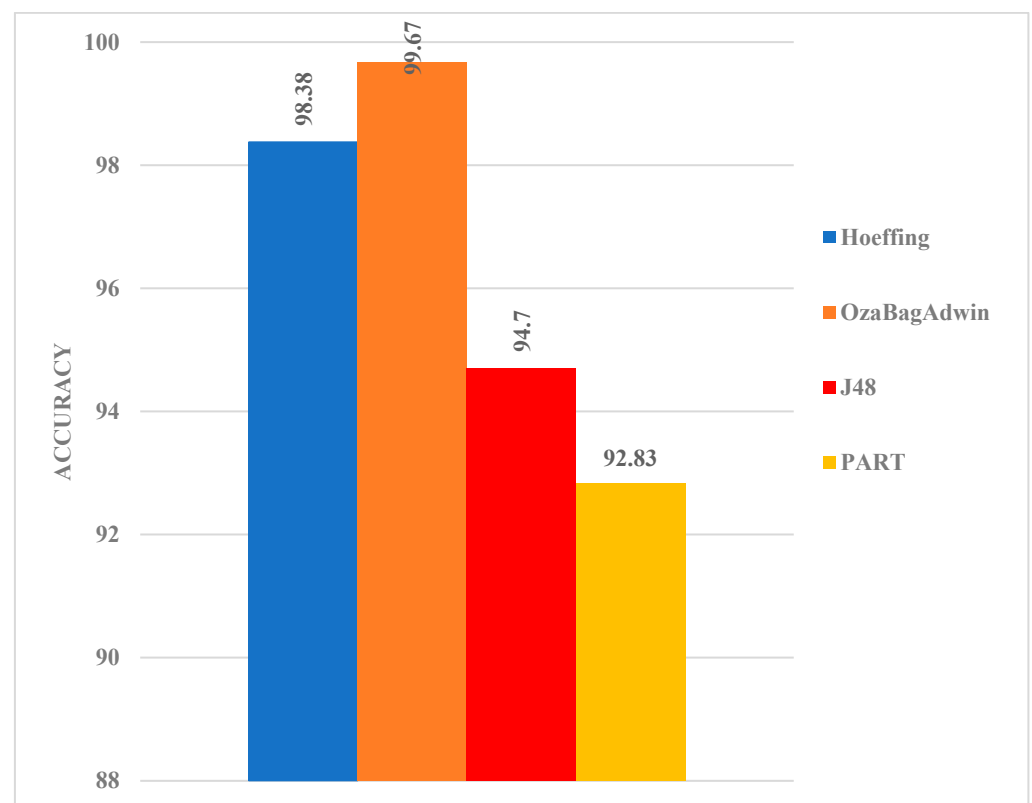
For the multiclass classification in the batch learning environment, J48 achieved accuracy, sensitivity, precision, f-measure, and ROC 87.66%, 87.70%, 88.00%, 87.00%, and 97.00%, respectively. However, PART achieved 87.05%, 87.10%, 87.00%, 87.00%, and 97.00%, respectively, for the same metrics. This result shows that J48 still achieved higher results based on the evaluation metrics considered in this study. However, both algorithms achieved the same false positive rate of 2.4%. The results of the experiments are shown in Table 5.

Table 5. J48 and PART algorithms multiclass classification results.

Metrics	J48 Mean Value	PART Mean Value
Accuracy	87.66	87.05
Sensitivity	87.70	87.10
Precision	0.88	0.87
F-measure	0.87	0.87
ROC	0.97	0.97
False Positive Rate	0.024	0.024

5.5. Performance Comparison of DSM and BLM Based on Binary Class Classification

After conducting a series of experiments based on binary classification, the accuracy obtained for the two streaming algorithms outperformed those of the batch learning algorithms. HT and OBA had higher accuracy of 98.38% and 99.67% when compared with J48 and PART with an accuracy of 94.70% and 92.83% (Figure 3). This result reveals that while batch learning algorithms have been mostly used for network intrusion detection in the case of binary classification problems, data streaming algorithms can serve as a replacement for batch learning algorithms in this scenario. This will enable the model to inherit all of the characteristics of the data streaming algorithm and thus, provide a scalable model that addresses the inherent problem of batch learning settings.

**Figure 3.** Model comparison based on the binary class classification.

5.6. Performance Comparison of DSM and BLM Based on Multi-Class Classification

After conducting a series of experiments based on multiclass classification (Figure 4), the accuracy obtained for the two streaming algorithms was HT (71.98%) and OBA (82.80%). For the batch learning algorithms, the accuracy was J48 (87.66%) and PART (87.05%). It is worth noting that the two batch learning algorithms achieved better accuracy when compared with the two streaming algorithms for the multiclass classification problem. This is a slight deviation from the result obtained for binary classification. The size of the

training samples for each of the classes may account for this variation. Nevertheless, the results of the two data streaming algorithms are still close to the batch learning algorithms.

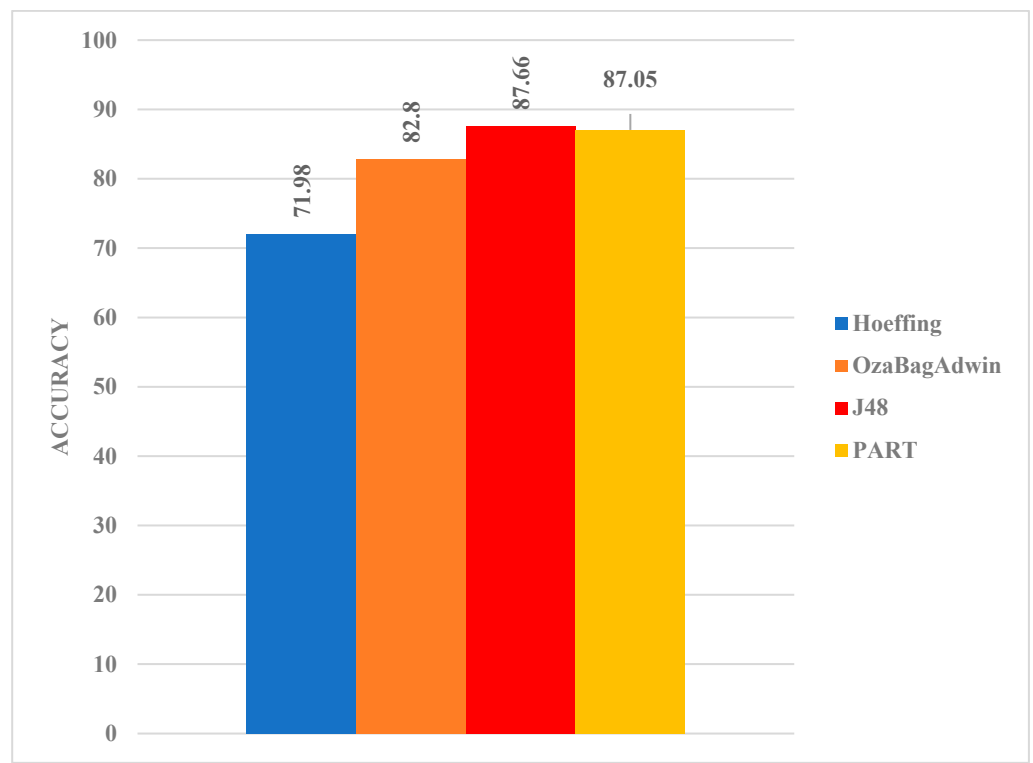


Figure 4. Model comparison based on the multiclass classification task.

5.7. Performance Comparison of DSM and BLM with Existing Studies

Comparing this work with existing studies that utilized the same dataset is vital. This assesses the performance of the proposed algorithms for network intrusion detection. Table 6 shows the baseline comparison of the proposed approach with the existing study. The data streaming algorithms for binary classification outperformed other algorithms, including the one proposed in the existing study. Although PART has a close result with J48 for multiclass classification, the result obtained with J48 shows superiority.

Table 6. Baseline comparison with existing work.

Approaches	Classification	Model	Accuracy (%)
BLM	Binary	J48	94.73
		PART	92.83
	Multiclass	J48	87.66
		PART	87.05
DSM	Binary	HT	98.38
		OBA	99.67
	Multiclass	HT	71.98
		OBA	82.80
[20]	Binary	Decision Tree	95.32
		Naïve Bayes	91.17
		Artificial Neural Network	92.54

5.8. Performance Comparison of DSM and BLM based on Statistical Test

To further compare the performance of DSM and BLM algorithms based on the results obtained in the previous section, we conducted a t-test statistical significance analysis. T-test is an inferential statistic method utilized to determine if there is a significant difference

between the means of two groups as well as how the two groups are related. With this test, we want to ascertain if there is a significant difference between the performance of DSM and BLM algorithms in terms of the binary and multiclass classification results. As shown in Table 7, the p -value (two-tail) result confirmed that there is a significant difference between the two settings in terms of the binary classification task (p -value < 0.05). However, as shown in Table 8, there is no significant difference between the two settings based on the multiclass classification results of the algorithms (p -value > 0.05). That is, the performance of both settings for multiclass classification task is similar.

Table 7. Statistical comparison of DSM and BLM based on binary classification results.

<i>t</i> -Test: Two-Sample Assuming Unequal Variances		
	DSM	BLM
Mean	99.025	93.78
Variance	0.83205	1.805
Hypothesized Mean Difference	0	
df	2	
t Stat	4.567739292	
p ($T \leq t$) one-tail	0.02236856	
t Critical one-tail	2.91998558	
p ($T \leq t$) two-tail	0.044737121	
t Critical two-tail	4.30265273	

Table 8. Statistical comparison of DSM and BLM based on multiclass classification results.

<i>t</i> -Test: Two-Sample Assuming Unequal Variances		
	DSM	BLM
Mean	77.39	87.355
Variance	58.5362	0.18605
Hypothesized Mean Difference	0	
df	1	
t Stat	−1.839039075	
p ($T \leq t$) one-tail	0.158531543	
t Critical one-tail	6.313751515	
p ($T \leq t$) two-tail	0.317063086	
t Critical two-tail	12.70620474	

5.9. Findings Based on Research Questions

To answer the research question raised in the introduction section, the following conclusions were drawn based on the experiments conducted.

RQ1: How effective are the selected batch learning algorithm implementations for detecting network intrusion detection?

J48 and PART algorithms were the batch learning algorithm considered. The experimental results showed that J48 produced significant results when compared with PART. This superior performance was observed across the binary and multiclass classification. This may account for the reason why tree-based algorithms, among the well-known algorithm categories, have been widely used for intrusion detection.

RQ2: How effective are the selected data streaming algorithm implementations for detecting network intrusion detection?

HT and OBA were considered for the data streaming algorithm. The result of the experiment revealed that for binary classification, OBA achieved better results when compared with HT. Additionally, for multiclass classification, OBA achieved better results compared to HT. Nevertheless, the training time of the OBA algorithm was greater than those of NB and HT on the dataset with large samples.

RQ3: Can data streaming algorithms achieve comparable performance compared with batch learning algorithms?

The data streaming algorithms achieved better results when compared with the batch learning algorithms for binary classification problems. Although J48 produced the highest result for multiclass classification, the data streaming algorithms also produced comparable results when compared with batch learning algorithms for multiclass classification problems. The results of the t-test statistic conducted further supported the findings. The t-test statistic results showed that there is a significant difference between the performance of DSM and BLM algorithms for binary classification task. However, there was no significant difference between the performance of DSM and BLM for multiclass classification task. Nevertheless, the performance of the selected data streaming algorithms can be improved for multiclass classification tasks to achieve better results on large evolving data streams with intrusion characteristics based on the evaluation metrics considered in this study.

RQ4: How effective is the performance of the two sets of algorithms compared with existing methods?

To empirically answer this research question, the study compares the results of the batch and stream learning algorithms proposed in this paper with the existing study [20]. The work of [20] was chosen because the dataset used in our study was collected by the authors. Since the same dataset was used in the two studies, this provides a fair comparison in terms of the performances of the algorithms. As shown in the previous section, the proposed data streaming algorithms for binary class classification outperformed all the three algorithms in the existing study. Likewise, the J48 algorithm outperformed naïve Bayes and artificial neural networks.

6. Conclusions and Future Work

In this study, an investigative study to ascertain the performance of both data streaming and batch learning algorithms for network intrusion detection was conducted. To aid this comparison, this research studied two algorithms in each learning method. Particularly, the HT and OBA algorithms were selected from the data streaming settings as well as J48 and PART algorithms from the batch learning settings. Both binary and multiclass classification tasks were investigated, and the result of the binary classification for network intrusion favored data streaming algorithms. This result was further supported based on the t-test statistical analysis. With a slight improvement in the evaluation metrics used, batch learning algorithms showed superiority over data streaming algorithms for multiclass classification problems. Nevertheless, the results of the data streaming algorithms for multiclass classification are still close to the batch learning settings, which was also supported by the t-test statistical analysis. Future work can investigate the performance of these two settings by considering more batch learning and data streaming algorithms, such as deep learning, random forest, Hoeffding option tree, and Hoeffding adaptive tree as well as different datasets to extend the findings in this study. Although there is no significant difference in the performance of the two settings in terms of multiclass classification results, nevertheless, the performance of data streaming algorithms for multiclass classification tasks can be improved in future work based on the evaluation metrics considered.

Author Contributions: The manuscript was written through the contributions of all authors. Conceptualization, K.S.A. and T.T.S.-I.; methodology, K.S.A., J.B.A., R.M.I. and I.D.O.; software, A.O.B., T.O.A. and K.S.A.; validation, M.A., A.L.I. and I.D.O.; formal analysis, A.L.I.; investigation, J.B.A.; resources, K.S.A.; data curation, A.O.B.; writing—original draft preparation, T.T.S.-I. and R.M.I.; writing—review and editing, I.D.O., M.A., T.O.A. and R.M.I.; visualization, A.L.I.; supervision, J.B.A.; project administration, K.S.A., T.T.S.-I. and A.L.I.; funding acquisition, J.B.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The dataset used is publicly available at <https://research.unsw.edu.au/projects/unsw-nb15-dataset> (accessed on: 21 August 2021).

Acknowledgments: The work of Agbotiname Lucky Imoize is supported in part by the Nigerian Petroleum Technology Development Fund (PTDF) and in part by the German Academic Exchange Service (DAAD) through the Nigerian-German Postgraduate Program under grant 57473408.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Balogun, O.A.; Mojeed, H.A.; Adewole, K.S.; Akintola, A.G.; Salihu, S.A.; Bajeh, A.O.; Jimoh, R.G. Optimized Decision Forest for Website Phishing Detection. In Proceedings of the Computational Methods in Systems and Software, Online, 1 October 2021; Springer: Cham, Switzerland, 2021; pp. 568–582.
2. Balogun, O.A.; Adewole, K.S.; Bajeh, A.O.; Jimoh, R.G. Cascade Generalization Based Functional Tree for Website Phishing Detection. In Proceedings of the International Conference on Advances in Cyber Security, Penang, Malaysia, 24–25 August 2021; Springer: Singapore, 2021; pp. 288–306.
3. Balogun, A.O.; Adewole, K.S.; Raheem, M.O.; Akande, O.N.; Usman-Hamza, F.E.; Mabayoje, M.A.; Akintola, A.G.; Asaju-Gbolagade, A.W.; Jimoh, M.K.; Jimoh, R.G.; et al. Improving the phishing website detection using empirical analysis of Function Tree and its variants. *Heliyon* **2021**, *7*, e07437. [[CrossRef](#)] [[PubMed](#)]
4. Awotunde, J.B.; Chakraborty, C.; Adeniyi, A.E. Intrusion Detection in Industrial Internet of Things Network-Based on Deep Learning Model with Rule-Based Feature Selection. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 7154587. [[CrossRef](#)]
5. Utzerath, J.; Dennis, R. Numbers and statistics: Data and cyber breaches under the General Data Protection Regulation. *Int. Cybersecur. Law Rev.* **2021**, *2*, 339–348. [[CrossRef](#)]
6. Elijah, A.V.; Abdullah, A.; Jhanjhi, N.; Supramaniam, M. Ensemble and Deep-Learning Methods for Two-Class and Multi-Attack Anomaly Intrusion Detection: An Empirical Study. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 520–528. [[CrossRef](#)]
7. Balogun, A.O.; Jimoh, R.G. Anomaly intrusion detection using an hybrid of decision tree and K-nearest neighbor. *J. Adv. Sci. Res. Appl.* **2015**, *2*, 67–74.
8. Atli, B.G.; Miche, Y.; Kalliola, A.; Oliver, I.; Holtmanns, S.; Lendasse, A. Anomaly-Based Intrusion Detection Using Extreme Learning Machine and Aggregation of Network Traffic Statistics in Probability Space. *Cogn. Comput.* **2018**, *10*, 848–863. [[CrossRef](#)]
9. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20. [[CrossRef](#)]
10. Salau-Ibrahim, T.T.; Jimoh, R.G. Negative Selection Algorithm Based Intrusion Detection Model. In Proceedings of the 20th IEEE Mediterranean Electrotechnical Conference, MELECON 2020-Proceedings, Palermo, Italy, 16–18 June 2020; pp. 202–206.
11. Adewole, K.S.; Raheem, M.O.; Abdulaheem, M.; Oladipo, I.D.; Balogun, A.O.; Baker, O.F. Malicious URLs Detection Using Data Streaming Algorithms. *J. Teknol. Dan Sist. Komput.* **2021**, *9*, 224–229. [[CrossRef](#)]
12. Kasongo, S.M.; Sun, Y. Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset. *J. Big Data* **2020**, *7*, 105. [[CrossRef](#)]
13. Jiang, K.; Wang, W.; Wang, A.; Wu, H. Network Intrusion Detection Combined Hybrid Sampling with Deep Hierarchical Network. *IEEE Access* **2020**, *8*, 32464–32476. [[CrossRef](#)]
14. Kaja, N.; Shaout, A.; Ma, D. An intelligent intrusion detection system. *Appl. Intell.* **2019**, *49*, 3235–3247. [[CrossRef](#)]
15. Liu, H.; Lang, B. Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. *Appl. Sci.* **2019**, *9*, 4396. [[CrossRef](#)]
16. Nguyen, M.T.; Kim, K. Genetic convolutional neural network for intrusion detection systems. *Future Gener. Comput. Syst.* **2020**, *113*, 418–427. [[CrossRef](#)]
17. Sinha, U.; Gupta, A.; Sharma, D.K.; Goel, A.; Gupta, D. Network Intrusion Detection Using Genetic Algorithm and Predictive Rule Mining. In *Cognitive Informatics and Soft Computing*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 143–156.
18. Tao, Z.; Huiling, L.; Wenwen, W.; Xia, Y. GA-SVM based feature selection and parameter optimization in hospitalization expense modeling. *Appl. Soft Comput.* **2019**, *75*, 323–332. [[CrossRef](#)]
19. Gu, J.; Lu, S. An effective intrusion detection approach using SVM with naïve Bayes feature embedding. *Comput. Secur.* **2021**, *103*, 102158. [[CrossRef](#)]
20. Moustafa, N.; Turnbull, B.; Choo, K.-K.R. An Ensemble Intrusion Detection Technique Based on Proposed Statistical Flow Features for Protecting Network Traffic of Internet of Things. *IEEE Internet Things J.* **2018**, *6*, 4815–4830. [[CrossRef](#)]
21. Radiuk, P.M. Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets. *Inf. Technol. Manag. Sci.* **2018**, *20*, 20–24. [[CrossRef](#)]
22. Horchulhack, P.; Viegas, E.K.; Santin, A.O. Toward feasible machine learning model updates in network-based intrusion detection. *Comput. Netw.* **2022**, *202*, 108618. [[CrossRef](#)]
23. Data, M.; Aritsugi, M. AB-HT: An Ensemble Incremental Learning Algorithm for Network Intrusion Detection Systems. In Proceedings of the 2022 International Conference on Data Science and Its Applications (ICoDSA), Bandung, Indonesia, 6–7 July 2022; pp. 47–52.
24. Saeed, M.M. A real-time adaptive network intrusion detection for streaming data: A hybrid approach. *Neural Comput. Appl.* **2022**, *34*, 6227–6240. [[CrossRef](#)]

25. Gong, Q.; DeMar, P.; Altunay, M. ThunderSecure: Deploying real-time intrusion detection for 100G research networks by leveraging stream-based features and one-class classification network. *Int. J. Inf. Secur.* **2022**, *21*, 799–812. [[CrossRef](#)]
26. Fekri, M.N.; Patel, H.; Grolinger, K.; Sharma, V. Deep learning for load forecasting with smart meter data: Online Adaptive Recurrent Neural Network. *Appl. Energy* **2021**, *282*, 116177. [[CrossRef](#)]
27. Wang, K.; Gopaluni, R.B.; Chen, J.; Song, Z. Deep Learning of Complex Batch Process Data and Its Application on Quality Prediction. *IEEE Trans. Ind. Inform.* **2018**, *16*, 7233–7242. [[CrossRef](#)]
28. Sarwar, S.S.; Ankit, A.; Roy, K. Incremental Learning in Deep Convolutional Neural Networks Using Partial Network Sharing. *IEEE Access* **2019**, *8*, 4615–4628. [[CrossRef](#)]
29. Yadav, S.S.; Jadhav, S.M. Deep convolutional neural network based medical image classification for disease diagnosis. *J. Big Data* **2019**, *6*, 113. [[CrossRef](#)]
30. Parisi, G.I.; Kemker, R.; Part, J.L.; Kanan, C.; Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Networks* **2019**, *113*, 54–71. [[CrossRef](#)]
31. Schelter, S.; Biessmann, F.; Januschowski, T.; Salinas, D.; Seufert, S.; Szarvas, G. *On Challenges in Machine Learning Model Management*; Bulletin of the IEEE Computer Society Technical Committee on Data Engineering; IEEE Explorer; Pennsylvania State University: State College, PA, USA, 2018.
32. Mnih, V.; Badia, A.P.; Mirza, L.; Graves, A.; Harley, T.; Lillicrap, T.P.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the 33rd International Conference on Machine Learning, PMLR, New York, NY, USA, 19–24 June 2016; pp. 1928–1937.
33. Stan, T.; Thompson, Z.T.; Voorhees, P.W. Optimizing convolutional neural networks to perform semantic segmentation on large materials imaging datasets: X-ray tomography and serial sectioning. *Mater. Charact.* **2020**, *160*, 110119. [[CrossRef](#)]
34. Tran, H.T.N.; Ang, K.S.; Chevrier, M.; Zhang, X.; Lee, N.Y.S.; Goh, M.; Chen, J. A benchmark of batch-effect correction methods for single-cell RNA sequencing data. *Genome Biol.* **2020**, *21*, 12. [[CrossRef](#)]
35. Garbuio, M.; Gheno, G. An Algorithm for Designing Value Propositions in the IoT Space: Addressing the Challenges of Selecting the Initial Class in Reference Class Forecasting. *IEEE Trans. Eng. Manag.* **2021**, 1–12. [[CrossRef](#)]
36. Montero-Manso, P.; Hyndman, R.J. Principles and algorithms for forecasting groups of time series: Locality and globality. *Int. J. Forecast.* **2021**, *37*, 1632–1653. [[CrossRef](#)]
37. Benzer, S.; Garabaghi, F.H.; Benzer, R.; Mehr, H.D. Investigation of some machine learning algorithms in fish age classification. *Fish. Res.* **2022**, *245*, 106151. [[CrossRef](#)]
38. Adewole, K.S.; Akintola, A.G.; Salihu, S.A.; Faruk, N.; Jimoh, R.G. Hybrid Rule-Based Model for Phishing URLs Detection. *Lect. Notes Inst. Comput. Sci. Soc.-Inform. Telecommun. Eng.* **2019**, *285*, 119–135.
39. Rutkowski, L.; Jaworski, M.; Duda, P. Basic Concepts of Data Stream Mining. In *Stream Data Mining: Algorithms and Their Probabilistic Properties*; Springer: Cham, Switzerland, 2020; pp. 13–33.
40. Kholghi, M.; Keyvanpour, M. An analytical framework for data stream mining techniques based on challenges and requirements. *arXiv* **2011**, arXiv:1105.1950.
41. Gaber, M.M.; Gama, J.; Krishnaswamy, S.; Gomes, J.B.; Stahl, F. Data stream mining in ubiquitous environments: State-of-the-art and current directions. *WIREs Data Min. Knowl. Discov.* **2014**, *4*, 116–138. [[CrossRef](#)]
42. Ramírez-Gallego, S.; Krawczyk, B.; García, S.; Woźniak Michał and Herrera, F. A survey on data preprocessing. *Neurocomputing* **2017**, *239*, 39–57. [[CrossRef](#)]
43. Morales, G.D.F.; Bifet, A.; Khan, L.; Gama, J.; Fan, W. IoT big data stream mining. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Online, 13–17 August 2016; pp. 2119–2120.
44. Adewole, K.S.; Raheem, M.O.; Abikoye, O.C.; Ajiboye, A.R.; Oladele, T.O.; Jimoh, M.K.; Aremu, D.R. Malicious Uniform Resource Locator Detection Using Wolf Optimization Algorithm and Random Forest Classifier. In *Machine Learning and Data Mining for Emerging Trend in Cyber Dynamics*; Springer: Cham, Swizarland, 2021; pp. 177–196.
45. Balogun, A.O.; Basri, S.; Mahamad, S.; Capretz, L.F.; Imam, A.A.; Almomani, M.A.; Adeyemo, V.E.; Kumar, G. A Novel Rank Aggregation-Based Hybrid Multifilter Wrapper Feature Selection Method in Software Defect Prediction. *Comput. Intell. Neurosci.* **2021**, *2021*, 5069016. [[CrossRef](#)] [[PubMed](#)]
46. Alsariera, Y.A.; Balogun, A.O.; Adeyemo, V.E.; Tarawneh, O.H.; Mojeed, H.A. Intelligent Tree-Based Ensemble Approaches for Phishing Website Detection. *J. Eng. Sci. Technol.* **2022**, *17*, 0563–0582.