Article

Using Incremental Ensemble Learning Techniques to Design Portable Intrusion Detection for Computationally Constraint Systems

Promise Ricardo Agbedanu [1,‡]*, Richard Musabe [2,‡], James Rwigema [1,‡] and Ignace Gatare [3,‡]

1 African Center of Excellence in Internet of Things, University of Rwanda, Kigali, Rwanda
2 Rwanda Polytechnic, Kigali, Rwanda
2 College of Science and Technology, University of Rwanda, Kigali, Rwanda
* Correspondence: ricardopromise@gmail.com
‡ These authors contributed equally to this work.

Abstract: Computers have evolved over the years and as the evolution continues, we have been ushered into an era where high-speed internet has made it possible for devices in our homes, hospital, energy and industry to communicate with each other. This era is what is known as the Internet of Things (IoT). IoT has several benefits in the health, energy, transportation and agriculture sectors of a country's economy. These enormous benefits coupled with the computational constraint of IoT devices which makes it difficult to deploy enhanced security protocols on them make IoT devices a target of cyber-attacks. One approach that has been used in traditional computing over the years to fight cyber-attacks is Intrusion Detection System (IDS). However, it is practically impossible to deploy IDS meant for traditional computers in IoT environments because of the computational constraint of these devices. In this regard, this study proposes a lightweight IDS for IoT devices using an incremental ensemble learning technique. We used Gaussian Naive Bayes and Hoeffding tree to build our incremental ensemble model. The model was then evaluated on the TON IoT dataset. Our proposed model was compared with other state-of-the-art methods proposed and evaluated using the same dataset. The experimental results show that the proposed model achieved an average accuracy of 99.98%. We also evaluated the memory consumption of our model which showed that our model achieved a lightweight model status of 650.11KB as the highest memory consumption and 122.38KB as the lowest memory consumption.

Keywords: Internet of Things;Incremental Machine Learning; Intrusion Detection System; Online Machine Learning; Cyber-Security, Ensemble Learning

1. Introduction

As the evolution of computing technology continues, the ability of things such as fridges, air-conditioners, medical equipment, and meters among others to communicate has become a reality due to fast communication technologies. A paradigm popularly known as the Internet of Things (IoT) has not only become a household term with smart homes, but it also has numerous uses in energy, agriculture, manufacturing, healthcare, and transportation. There is no doubt that the IoT has many benefits, which is why the number of IoT devices is growing at an exponential rate. The number of IoT devices is estimated to reach 30.9 billion by 2025, according to [1]. The numerous benefits of the IoT ecosystem make it attractive to cyber-attacks. An attack statistic presented by SAM Seamless Network shows that over 1 billion IoT-based attacks happened in 2021 [2]. Although methodologies such as encryption and secured architecture are progressively being deployed to ensure that IoT devices are secured, the computational constraint of these devices makes it difficult to implement these security measures to their fullest potential. Another approach to securing these devices from cyber-attacks is to detect these attacks before they are exploited by an

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 September 2022 doi:10.20944/preprints202209.0058.v2

attacker. Intrusion Detection Systems have been around for more than four decades with the development of these IDSs focused on traditional computing systems [3]. They have been among the primary methodologies used to protect computer networks. [4] defines intrusion detection as the process of detecting activities perpetrated against computer systems by intruders. Over the past forty years, a lot of breakthroughs have been made in the area of intrusion detection. One of the biggest breakthroughs in this area is the use of machine learning in detecting intrusions. However, with all these breakthroughs it is practically impossible to deploy traditional computing-based IDS methods in the internet of things. This situation has been created because of the computational constraints posed by IoT devices. This has led to several studies being carried out to design IDSs that can be deployed in IoT systems without significantly affecting the computational resources of these devices. Several approaches have been proposed in designing lightweight IDSs for IoT environments, but these studies fail to either report how these lightweight IDSs are achieved or how much computational resource these proposed approaches consume. For example, [5–9] proposed various techniques that are supposed to translate into lightweight IDSs but these works either failed to report how these methods translate into lightweight IDS or how much computational resources these proposed methods consume. In this study, we propose a novel lightweight intrusion detection system using an incremental machine learning approach. The main contributions of this study are as follows

- Using an incremental machine learning approach to design a lightweight IoT intrusion detection system
- Measuring the memory consumption of our proposed model.
- The study uses an incremental ensemble approach to achieve improved accuracy.
- The study evaluates the proposed IDS model on an IoT dataset.

The remainder of the paper is structured as follows. Section 2 discusses the study's background. Section 3 focuses on works relevant to our study, whereas sections 4, 5, and 6 focus on the proposed model, experimental evaluation, and conclusion, respectively.

2. Background

2.1. Intrusion Detection System

An intrusion detection system (IDS) is a security device that detects illegal access to data within a networked or computer-based environment in order to threaten the integrity, availability, or confidentiality of the computing device [10,11]. The objective of an IDS continuously monitors network traffic and flag any activity that violates the normal usage of the system [12]. According to [13], typically, an IDS consists of sensors, an analysis engine and some reporting system. Intrusion detection systems can be classified either on how they are deployed or how they detect illegal activities. From a deployment perspective, an IDS can either be classified as distributed, centralized or a hybrid. on the other hand, an IDS can be classified as a signature-based, anomaly-based, specification-based or hybrid. [14] signature-based detection is the set of pre-defined rules such as the sequence of bytes in network traffic that are pre-loaded to trigger an alert when a matched sequence is detected. On the other hand, anomaly detection records the normal behaviour of a network and then compares them with the current behaviour of the system. The authors also explained the specification-based detection method as an approach that uses input specifications that are designed manually. Finally, hybrid detection methods deploy a combination of signature-based, anomaly-based and specification-based detection methods with the aim of improving accuracy and reducing false positive rates.

2.2. Ensemble Learning

According to [15], ensemble learning is a machine learning technique that focuses on combining the strengths of different machine learning algorithms into a single algorithm. The primary goal of ensemble learning is to improve accuracy by leveraging the strengths of the ensemble learners [16].

 doi:10.20944/preprints202209.0058.v2

There are instances where traditional machine learning models do not achieve high accuracy, [17] Several ensemble-based techniques have been developed over time, but the most popular are bagging, boosting, stacking generalization, and expert mixture [16].

In the preceding paragraphs, we briefly explain the three categories of ensemble learning.

2.2.1. Bagging-based Learning

Bagging, short for bootstrap aggregation, is an algorithm that is best suited for problems with a small training dataset. Given a training set S with a cardinality n, the bagging algorithm trains several independent classifiers T. Each of these classifiers are trained using a percentage of N [16] sampling. Linear classifiers such as linear SVM, decision stumps, and single-layer perceptrons are excellent candidates for bagging [16]. Classifiers are trained and then combined using simple majority voting in bagging.

Bagging, an abbreviation for bootstrap aggregation, is a method that works well with issues that have a limited training dataset. The bagging algorithm learns several independent classifiers T given a training set S with a cardinality of n. Each of these classifiers is trained using a proportion of N citezhang2012ensemble sampling. Linear classifiers like linear SVM, decision stumps, and single-layer perceptrons are great candidates for bagging [16].. In bagging, classifiers are trained and then concatenated using simple majority voting.

2.2.2. Boosting-based Learning

An iterative approach can be used to generate a strong classifier from a set of weak classifiers. Although boosting also combines a large number of weak learners through simple majority voting, there is one significant difference between boosting and bagging. Every instance in bagging has an equal chance of being in each dataset used in training. In boosting, on the other hand, the dataset used to train each subsequent model focuses on instances misclassified by the previous model. At any given time, a boosting designed for a binary class problem generates a set of three weak classifiers. The first learning classifier is trained on a random subset of the training data available. A different subset of the original training dataset is used to train the second learning classifier [18].

2.2.3. Stack Generalization

Non-trainable combiners are used in both bagging and boosting methods. The combination weights in non-trainable combiners are determined after the classifiers have been trained. The combination rule used in non-trainable combiners does not allow determining which member classifier learned from which partition of the feature space [16]. Trainable combiners can be used to solve this problem. Individual ensemble members can be combined using a separate classifier in stacked generalization.

2.2.4. Mixture of Experts

A sampling technique is used to train an ensemble of classifiers in a mixture of experts. The classifiers are then combined using a weighted combination rule [19]. Furthermore, a mixture of experts can encompass the selection of algorithms, with each classifier trained to become an expert in a different aspect of the feature space. Individual classifiers are usually not weak classifiers since they are trained to become experts.

2.3. Online/Incremental Machine Learning

Online machine learning is also known as incremental machine learning is increasingly becoming popular in the area of real-time data streams. [20] online algorithms instantaneously build machine learning models after seeing a small portion of the data. This leads to the inability to undo less optimal decisions that were earlier made because the data used will no longer be available for the algorithm. The concept of machine learning models being able to acquire knowledge from continuous data without accessing the original data has been applied to domains like intelligent robots, auto-driving and unmanned aerial vehicles

 doi:10.20944/preprints202209.0058.v2

[21–23]. According to [24] as reported by [20], in data stream models infinite stream of data arrives continuously and these streams of data are to be processed by systems that have resource constraints. The main restriction of data stream models is that memory of these systems are usually small and can only hold a minimal portion of the data stream. When it comes to data stream models, only a minimal subset of the data can be kept for instant data analysis [25]. Figure 1 shows an online machine learning model using an offline dataset whiles figure 2 shows the same model using streams of network traffic.

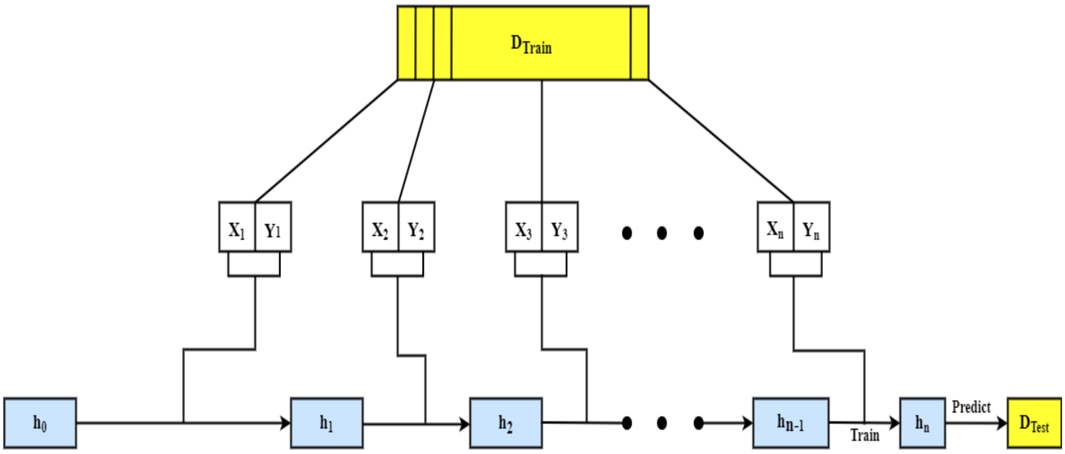

Figure 1. Online machine learning using Offline dataset.

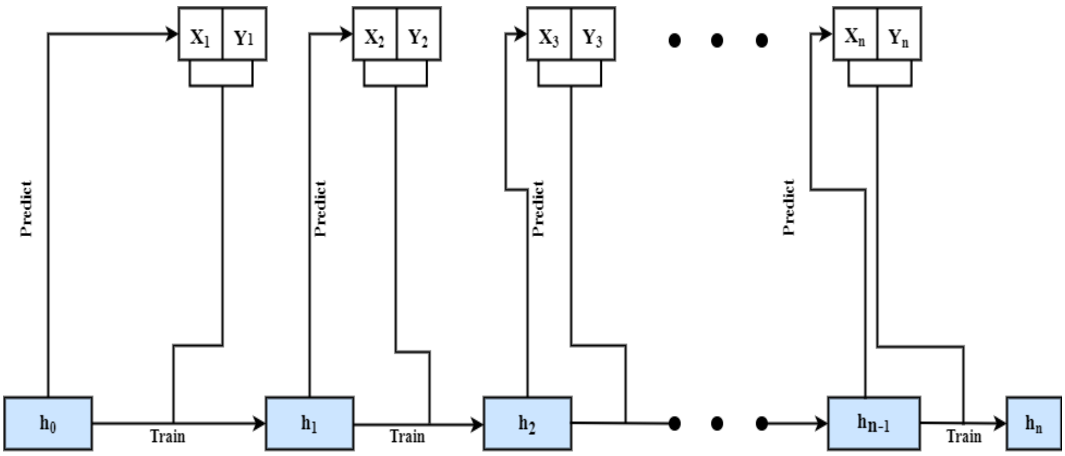

Figure 2. Online machine learning using data stream.

3. Related Work

Throughout this section, we will look at some studies that are relevant to our work. Yang et al [26] proposed an ensemble framework for intrusion detection systems (IDSs) in IoT environments, with a primary focus on idea drift adaptability. To manage concept drift in IoT anomaly detection, the suggested framework employs a technique known as performance-weighted probability averaging ensemble. When compared to other cutting-edge approaches, the suggested framework performed better. Even though the author's proposed method took less time to run than the other methods they looked at for their study, they did not look into how the proposed model affected other computing parameters, such as memory.

Jan et al [5] used a supervised Support Vector Machine (SVM) to detect IoT adversarial attacks. The authors utilized only the packet arrival rate to the sensor node to design the proposed IDS. The accuracy of the proposed IDS showed better performance compared to other models like neural networks, KNN and decision trees. One of the drawbacks

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 September 2022 doi:10.20944/preprints202209.0058.v2

of this study is that it only considered DDoS attacks Additionally, the authors failed to report how the proposed approach leads to a lightweight IDS. Parameters such as memory consumption and model running time were not reported.

In a similar study, [27] proposed a lightweight IDS for IoT ecosystems using a Deep Belief Network and Genetic Algorithm. According to the study, the proposed system was more accurate than other methods that were looked at for the study. However, the study failed to report how the proposed approach translates to a lightweight model. Moreover, the dataset used for the experimental validation is not an IoT-based dataset. Like in other studies, parameters that are supposed to prove the lightweight status of the proposed method were not considered in the study.

Roy et al [7] also designed a lightweight IDS for IoT systems by using a set of optimization techniques. They used multicollinearity, sampling, and dimensionality reduction to reduce the training data, which resulted in a shorter training time. Like other earlier related works considered in this section, although their proposed approach reduces the training time of the model, the study did not report how much memory the model consumes.

Zhao et al. also [8], which suggested a network intrusion detection method for IoT devices utilising a lightweight neural network. To minimise the dimensionality of features, the authors employed a principal component analysis approach. The proposed method was tested using the UNSW-NB15 and Bot-IoT datasets. Despite the fact that the authors determined that both the ultralight feature extraction network and principal component analysis contributed to the suggested model's lightweight performance, they did not report on the computational complexity of their proposed method.

In order to make a lightweight IDS for IoT systems, [9] said that they used a mix of feature selection techniques on different datasets to make a lightweight IDS algorithm for IoT traffic. However, two of the datasets used to evaluate their proposed lightweight IDS were non-IoT related. Also, the authors didn't talk about how their proposed model would affect the computing power in their experimental environment.

Latif et al [6] reported using a Dense Random Neural Network to develop a lightweight intrusion detection system for IoT environments. The proposed model was evaluated on the ToN-IoT dataset, and the results show a detection accuracy of 99.14% for binary class classification and 99.05% for multiclass classifications. However, Latif et al. did not report on the computational complexity of their proposed model. A parameter is required to measure the lightness of the proposed model. Pan et al [28] also suggested a lightweight intelligent intrusion detection system (IDS) architecture for wireless sensor networks. To create their model, the authors used KNN and the sine cosine technique. The authors reported that combining the above techniques improves classification accuracy and also reduces false alarms. However, the authors failed to report how the lightweight model was achieved or what parameters were used to determine the lightweight status of the model.

Reis et al. [29] created an IDS for cyber-physical systems using incremental support vector machines. In their study, a one-class support vector machine was applied to each sensor to retrieve abnormal behaviors. As an output of the proposed incremental machine learning model, these anomalies are orchestrated. Although the model proposed by Reis et al. achieved an accuracy higher than 95%, the study didn't go into detail about how the proposed method would affect the computational resources of cyber-physical systems.

To reduce the computation overhead [30], introduced a privacy-preserving pipeline-based intrusion detection for distributed incremental learning that selects unique features using an innovative extraction technique. Current incremental learning techniques are computationally expensive. The distributed intrusion detection method is used to distribute the load across IoT and edge devices. Theoretical analysis and experiments show that state-of-the-art techniques require less space and time. The study, however, reported on time complexity but not on space complexity. Furthermore, the experimental validation dataset is not an IoT-based dataset.

 doi:10.20944/preprints202209.0058.v2

4. Proposed Model

We detail the design and conceptual implementation of our suggested approach in this section. The suggested model is based on a machine learning technique, ensuring the creation of a lightweight IDS model suitable for the internet of things environment. The proposed model employs incremental machine learning and data streaming ensemble learning approaches to create a lightweight intrusion detection system for the internet of things environment. The proposed model processes network data generated in IoT environments as data streams. After each iteration, the model is updated. Figure 3 depicts our proposed model.

1. Pre-processing: At this stage, the dataset used to train our proposed model is cleaned. The data pre-processing approach used in this study includes imputing missing data values and transforming and selecting features that are important to train our machine learning model. We employed one-hot encoding as one of the techniques to pre-process our data. A single hot encoding transformer will encode all of the features that are provided to it. If a list or set is supplied, this transformer will encode each item in the list or set. By composing it with compose.Select in River, you can apply it to a subset of features.
2. Model training: In this study, we proposed a novel online stacking ensemble machine learning technique using Gaussian Naive Bayes and Hoeffding Tree Classifier. We chose these two machine learning models to build our ensemble learning because we wanted to achieve the following three objectives
    - Design a model that consumes a minimal computational resource (lightweight)
    - Building a fast model
    - A model that achieves a high accuracy

   Gaussian Naive Bayes and Hoeffding Tree Classifier are used as the base classifiers of our proposed model whiles Hoeffding Tree is used as the meta classifier of the proposed model. Each observation of the dataset is read as a stream and is then used to train the base and meta classifier. Each base classifier predicts each stream of data, that is, $X_i Y_i$ which becomes feature input to the meta classifier. The meta classifier (HT) then uses the outputs of the base classifiers to make a better prediction. We chose Hoeffding trees because they learn patterns in data without continuously storing data samples for future reprocessing. This makes them particularly suitable for use on embedded devices. Similarly, Gaussian NB is quick and flexible, and it produces highly reliable results. It works well with large amounts of data and requires little training time. It also improves grading performance by removing insignificant specifications [31,32].
3. Model evaluation: The final stage of the model is the model evaluation stage. The proposed model's accuracy, precision, recall, F1, model training time, and memory consumption are all evaluated.

We chose incremental or online machine learning to develop our framework in this work for the following reasons.

1. Network traffic is generated in blocks as a data stream. By using incremental learning on network traffic, models can predict the nature of traffic without having to be trained on large datasets.
2. The computational constraints of IoT devices make loading an entire training dataset into main memory difficult and impractical. Even if the entire training data can fit into the main memory of an IoT device, the device's computational power will be drastically reduced.
3. Because of the sophisticated nature of cyber-attacks, new data is constantly available. Retraining the model on the entire dataset will be time-consuming and computationally expensive. Because models in online machine learning are trained with data streams, they can easily learn from new data examples without consuming a lot of computational power.

 doi:10.20944/preprints202209.0058.v2

4. Real-time network traffic is generated. To prevent intruders from gaining unauthorized access to devices, this traffic must be analyzed in real-time. In real-time environments, online machine learning has proven to be an effective learning method.
5. Traffic flow is dynamic and constantly changing. Changes in network traffic can have an impact on the predictive performance of machine learning models, which is referred to as concept drift in machine learning. Models should be able to self-adapt to changes in the relationship between input and output data to handle concept drifts.

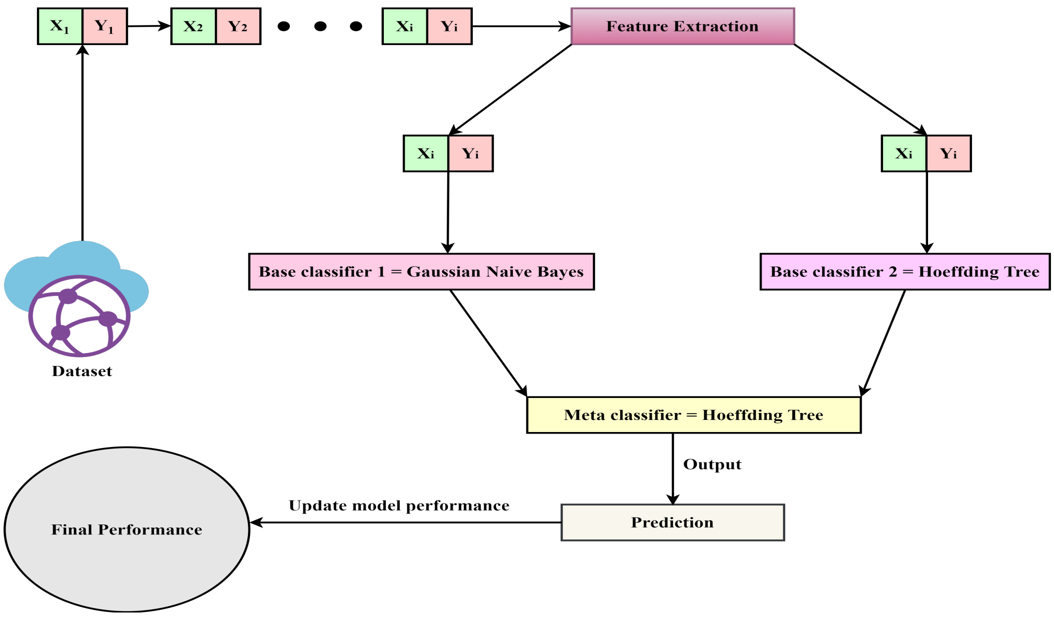

Figure 3. Our proposed model

4.1. Gaussian Naive Bayes

According to [33], the Naive Bayes algorithm is a typical illustration of how generative hypotheses and parameter guesses can facilitate learning. Consider the problem of predicting a label y ∈ {0,1} from a vector of characteristics **X** = $(x_1,..., x_d)$, where each x_i is in the range of {0,1}. The optimal classifier of Bayes is given below

$$h_{Bayes}(\mathbf{X}) = argmaxP[Y = y|X = x], y \in \{0, 1\} \tag{1}$$

We need 2^d parameters to define the probability function P[Y = y | X = x], each of which relates to P[Y = 1 | X = x] for a given value of y ∈ $\{0,1\}^d$. This means that when the number of features increases, so does the number of instances necessary. In the Naive Bayes technique, we make the generative assumption that, given the label, the features are independent of one another. To put it another way,

$$P[Y = y|X = x] = \prod_{i=1}^{d} P[Y = y|X = x] \tag{2}$$

The Bayes optimum classifier can be reduced further using this assumption and the Bayes rule:

$$h_{Bayes}(\mathbf{X}) = argmaxP[Y = y] \prod_{i=1}^{d} P[X_i = x_i|Y = y] \tag{3}$$

That is, the set of parameters to estimate has been reduced to 2d +1. In this situation, the generative assumption we made considerably decreased the number of parameters we needed to learn. When the maximum likelihood principle is used to figure out the parameters, the resulting classification model is called the Naive Bayes classifier.

One typical technique to handle continuous attributes in Naive Bayes classification is to use Gaussian distributions to express the probabilities of the features based on the classes. As a result, every attribute is represented as Xi N(, 2) by a Gaussian probability density function (PDF), [34] as reported by [35].

$$X_i \sim N(\mu, \sigma^2) \tag{4}$$

The Gaussian PDF is shaped like a bell and is defined by the equation below where μ is the mean and σ^2 is the variance.

$$N(\mu, \sigma^2)(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{5}$$

4.2. Hoeffding Tree (HT)

Hulten et al [36] are the first to propose Hoeffding trees. The Hoeffding tree algorithm is a fundamental algorithm for stream data classification. It is an induction of a decision tree algorithm that could learn from enormous data streams if the distributed generating examples remain constant over time. It creates decision trees that are similar to the standard batch learning method. Asymptotically, Hoeffding trees as well as decision trees are connected. The HT technique is based on the basic premise that a modest sample size can frequently be sufficient to identify an optimal splitting feature. The key point to understand here is that classic batch learning algorithms produce decision trees based on attribute splitting. The HT method is mathematically verified to use the Hoeffding bound. To comprehend the significance of the Hoeffding bound, a few assumptions must be made. Let's say we get N separate samples of a random variable r with a range of R, where r is a measure of attribute selection. In the case of Hoeffding trees, r is information gain, and if we calculate the mean value of r_{mean} for this sample, the Hoeffding limit indicates that the true mean of r is at least 1-δ. The primary benefits of the HT algorithm are as follows:

1. it is incremental in nature
2. it achieves high accuracy with small sample size.
3. scans on the same data are never performed.

However, Hoeffding Tree has a few disadvantages. The main disadvantage is that HT cannot handle concept drift because the node cannot be changed once it is created. [37] described how to deal with concept drift using classifiers. The algorithm devotes a significant amount of time to attributes with nearly identical splitting quality. Furthermore, memory utilization can be further optimized.

$$[!h] \in = \sqrt{\frac{R^2 1n \frac{1}{\delta}}{2n}} \tag{6}$$

The algorithm for Hoeffding tree is shown in algorithm 1.

5. Experimental Evaluation

5.1. EXPERIMENTAL ENVIRONMENT

The proposed method was implemented using Python 3.8 with River as our framework for online machine learning. The proposed method was implemented on a MacBook Pro running on an M1 chip with 16 GB of RAM. The TON-IoT dataset was used to evaluate the proposed framework. There are several incremental or online streaming libraries that provide machine functionalities. Some of these libraries are Creme, scikit-multiflow and River. In this study, we choose to build our incremental learning models using River. [38] is a merger of creme and Scikit-multiflow. River is a library that allows continual learning by handling dynamic data streams. We chose River because it includes data transformation methods, learning algorithms, and optimization algorithms. Its distinct data structure lends itself well to settings involving streaming data and web applications.

 doi:10.20944/preprints202209.0058.v2

Algorithm 1: Hoeffding Tree Algorithm [36]

Input: S is a sequence of examples,
X is a set of discrete attributes,
G(.) is a split evaluation function,
δ is one minus the desired probability of choosing the correct attribute at any given node

Output: HT is a decision tree

procedure HOEFFDINGTREE(S, X, G, δ)
    Let HT be a tree with a single leaf l_1 (the root)
    Let X_1 = X ∪ {X_θ }
    Let $\bar{G}_1(X_\theta)$ be the $\bar{G}$ obtained by predicting the most frequent class in S
    for *each class* Y_k **do**
        for *each value* X_{ij} *of each attribute* $X_i \in$ **X do**
            Let n_{ijk} (l_1) = 0
        end
    end
    for *each example* (X, Y_k) *in S* **do**
        Sort (x, y) into a leaf l using HT
        for *each* X_{ij} *in X ssuch that* $X_i \in$ **X do**
            Increment n_{ijk} (l)
        end
    end
    Label l with the majority class among the examples seen so far at l
    if *the examples seen so far at l are not all of the same class* **then**
    end
    Compute $\bar{G}_l$ (X_i) for each attribute $X_i \in X_l - X_\theta$ using the counts n_{ijk}(l)
    Let X_a be the attribute with highest $\bar{G}_l$.
    Let X_b be the attribute with second-highest $\bar{G}_l$.
    Compute ∈ using Equation 1
    if $\bar{G}_l$ $(X_a)-$ $\bar{G}_l$ $(X_b) > \in$ *and* $X_a \neq X_\theta$, **then**
    end
    Replace l by an internal node that splits on X_a.
    For each branch of the split
    Add a new leaf l_m and let X_m = X − {X_a}.
    Let $G_m(X_\theta)$ be the G obtained by predicting the most frequent class at l
    for *each class* Y_k *and each value* X_{ij} *of each attribute* $X_i \in X_m - \{X_\theta\}$ **do**
        Let $n_{ijk}(lm)$ = 0
    end
    return HT
end procedure

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 September 2022 doi:10.20944/preprints202209.0058.v2

5.2. DATASET

According to [39], the TON-IoT dataset was built by the Cyber Range and IoT Las at the University of South Wales. The dataset has nine (9) types of cyber-attacks. These are Denial of Service (DoS), Distributed Denial of Service (DDoS), ransomware, backdoor, data injection, scanning, Cross-site Scripting (XSS), password cracking and Man-in-The-Middle (MiTM). The generated data were from seven IoT and IIoT devices namely, fridge, motion light, garage door, GPS tracker, thermostat and weather. The fridge dataset has a total of 587076 records, the motion light dataset has 452262 records, the garage door has 591446 records, the GPS tracker produced 595686 records whiles the thermostat and weather produced 442228 and 650242 records respectively. The statistics of the dataset used is shown in tables 1 and 2.

5.3. EVALUATION METRICS

True positive (V_P): Positive intrusion that is both expected and confirmed.

False positive (U_P): An intrusion that was expected to be positive but ended up turning out to be negative.

True negative (V_N): The intrusion is expected to be negative and confirmed to be negative.

False negative (U_N): The intrusion was expected to be negative, but it turned out to be positive.

5.3.1. Accuracy

A model's overall accuracy can be measured by the number of correctly predicted events made by the given model. The formula below computes the total accuracy of the model.

$$Accuracy = \frac{V_P + V_N}{V_P + V_N + U_P + U_N}$$

5.3.2. Precision

Precision is found by dividing the total number of positive detections by the number of positive detections that were correctly identified as positive.

$$Precision = \frac{V_P}{V_P + U_P}$$

5.3.3. Recall

The recall is defined as the ratio of true positive detections to the number of real abnormal samples.

$$Recall = \frac{V_P}{V_P + U_N}$$

5.3.4. F1 Score

The F1 score is the average of precision and recall. The F1-score is determined as the weighted average of precision and recall, taking both the U_P and U_N into consideration.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * V_P}{2 * V_P + U_P + U_N}$$

5.3.5. Memory

The computational constraint of IoT devices makes it difficult and sometimes impossible to run IDS meant for traditional computers on these IoT devices. This calls for developing models that consume minimal memory (lightweight models).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 September 2022 doi:10.20944/preprints202209.0058.v2

Table 1. Statistics of TON IoT dataset [39]

Fridge IoT dataset	
Type of attack	**No of rows**
Backdoor	35568
DDoS	10233
Injection	7079
Normal	500827
Password	28425
Ransomware	2902
XSS	2042
GPS tracker IoT dataset	
Backdoor	35571
DDoS	10226
Injection	6904
Normal	513849
Password	513849
Ransomware	2833
Scanning	550
XSS	577
Motion light IoT dataset	
Backdoor	28209
DDoS	8121
Injection	5595
Normal	388328
Password	17521
Ransomware	2264
Scanning	1775
XSS	449
Weather IoT dataset	
Backdoor	35641
DDoS	15182
Injection	9726
Normal	559718
Password	25715
Ransomware	2865
Scanning	529
XSS	866
Garage IoT dataset	
Backdoor	35568
DDoS	10230
Injection	6331
Normal	515443
Password	19287
Ransomware	2902
Scanning	529
XSS	1156

5.3.6. Model Running Time

In this evaluation metric, we measure the total time it takes for the proposed model to run.

5.4. RESULTS

In this section, we present the results of the proposed and compared them with other state-of-the-art techniques. To begin with, this study compares the results of the proposed

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 September 2022 doi:10.20944/preprints202209.0058.v2

Table 2. Statistics of TON IoT dataset continuation [39]

Modus IoT dataset	
Backdoor	40035
Injection	7079
Normal	405904
Password	24269
Scanning	529
XSS	577
Thermostat IoT dataset	
Backdoor	35568
DDoS	10230
Injection	6331
Normal	515443
Password	19287
Ransomware	2902
Scanning	529
XSS	1156

model with other state-of-the-art IDS proposed and evaluated with the TON IoT dataset. We decided to limit the state-of-the-art to studied that used ToN IoT dataset because we wanted to eliminate biases. In comparing these studies, we considered the category of the TON IoT dataset used in each study, the method proposed by each of the works under consideration, the highest accuracy recorded and whether the study records the time used to build the model as well as the amount of memory the model consumes. The comparison of our approach with other state of art IDS for IoT systems is presented in table 3. Where the authors fail to report on a parameter, we indicate is as non-available (N/A). From table 3, it could be observed that out of the five states-of-the-art IDSs considered in the study none of them reports on the model or the memory consumption of their proposed technique. Although [40,41] both report 100% accuracy, our proposed model outperforms the methods proposed in those studies for the following reasons:

1. The accuracy reported in our study is the average accuracy of our proposed model whereas [40,41] reports total accuracy.
2. Our study focused on multi-class classification whereas [40,41] focused on binary class classification.

In table 4 the accuracy, time, and memory consumption of the models used to build our incremental ensemble technique are compared with our proposed model. Although the time and memory consumption of the individual models are lower than our proposed model, we wanted to propose a model that achieves a trade-off between accuracy, time, and memory consumption. Our proposed model ended up achieving a higher accuracy without significantly increasing the time and memory consumption. The time and memory consumption of the proposed model shows it can run on devices that are computationally constrained without significantly impacting negatively on the computational resources of these devices.

Figure 4 below shows the output of our proposed model in terms of the time taken to build the model. The results show that our proposed model recorded the least training time of 59 seconds on the thermostat dataset and the highest training time of 114 seconds on the weather IoT dataset. The concept of incremental learning allows our model to learn one stream of data at a time. Therefore, the time used to train the model on a stream of data will be the total observations in the dataset divided by the total model training time. This makes our model very fast irrespective of the size of the dataset.

When tested on the modus dataset, our proposed model achieved a superior average accuracy of 96.81%, with precision, recall, and F1 scores of 97.23%, 96.81% and 96.92% respectively. The Hoeffding tree had an average accuracy of 92.96%, with precision, recall,

 doi:10.20944/preprints202209.0058.v2

Table 3. Using the ToN IoT dataset, we compared our proposed model to state-of-the-art models that had been tested using the same dataset

Study	Year of the study	Method used	Highest accuracy	Model training time (S)	Memory consumption (KB)
[6]	2021	Dense Random Neural Network	99.14%	N/A	N/A
[40]	2022	Optimized decision tree	100	N/A	N/A
[41]	2022	Ensemble based voting	100	N/A	N/A
[42]	2022	Graph Neural Network	97.87	N/A	N/A
[43]	2021	Synthetic minority oversampling technique	99.0	N/A	N/A
Our proposed model	2022	Stack-based Incremental ensemble (HT and Gaussian NB)	99.98	71	122.38

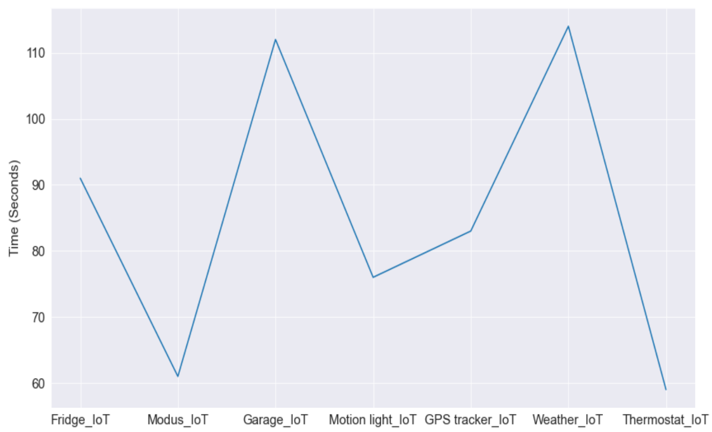

Figure 4. Model training time of our model using the TON IoT dataset.

 doi:10.20944/preprints202209.0058.v2

Table 4. Comparing the accuracy (Acc), model time consumption (Time) and memory usage of the base classifiers against our model on the different datasets

Fridge IoT dataset								
Gaussian NB			**Hoeffding Tree**			**Our proposed model**		
Acc	**Time (s)**	**Memory (KB)**	**Acc**	**Time (s)**	**Memory (KB)**	**Acc**	**Time (s)**	**Memory (KB)**
85.31%	29.9	15.85	98.68%	20.7	532.71	99.98%	104	650.11
Modus IoT dataset								
Gaussian NB			**Hoeffding Tree**			**Our proposed model**		
Acc	**Time (s)**	**Memory (KB)**	**Acc**	**Time (s)**	**Memory (KB)**	**Acc**	**Time (s)**	**Memory (KB)**
77.60%	19.2	19.2	92.36%	14	124.58	96.81%	75	495.25
Garage IoT dataset								
Gaussian NB			**Hoeffding Tree**			**Our proposed model**		
Acc	**Time (s)**	**Memory (KB)**	**Acc**	**Time (s)**	**Memory (KB)**	**Acc**	**Time (s)**	**Memory (KB)**
85.96%	37	27.05	95.70%	29.5	75.85	99.96%	131	394.95
Motion light IoT dataset								
Gaussian NB			**Hoeffding Tree**			**Our proposed model**		
Acc	**Time (s)**	**Memory (KB)**	**Acc**	**Time (s)**	**Memory (KB)**	**Acc**	**Time (s)**	**Memory (KB)**
85.86%	22.9	20.6	92.06%	15.5	33.56	99.98%	79	219.58
GPS Tracker IoT dataset								
Gaussian NB			**Hoeffding Tree**			**Our proposed model**		
Acc	**Time (s)**	**Memory (KB)**	**Acc**	**Time (s)**	**Memory (KB)**	**Acc**	**Time (s)**	**Memory (KB)**
85.20%	28.3	10.56	98.29%	18.7	120.36	99.97%	97	281.94
Weather IoT dataset								
Gaussian NB			**Hoeffding Tree**			**Our proposed model**		
Acc	**Time (s)**	**Memory (KB)**	**Acc**	**Time (s)**	**Memory (KB)**	**Acc**	**Time (s)**	**Memory (KB)**
86.08%	33.4	20.58	98.37%	23.7	314.91	99.93%	116	627.81
Thermostat IoT dataset								
Gaussian NB			**Hoeffding Tree**			**Our proposed model**		
Acc	**Time (s)**	**Memory (KB)**	**Acc**	**Time (s)**	**Memory (KB)**	**Acc**	**Time (s)**	**Memory (KB)**
87.27%	19.9	6.85	99.12%	13.8	55.07	99.94%	71	122.38

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 September 2022 doi:10.20944/preprints202209.0058.v2

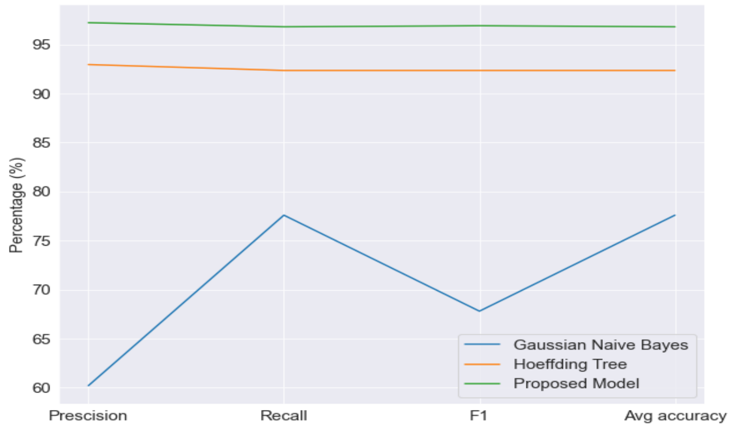

Figure 5. Accuracy of Gaussian NB, HT and proposed model on Modus dataset.

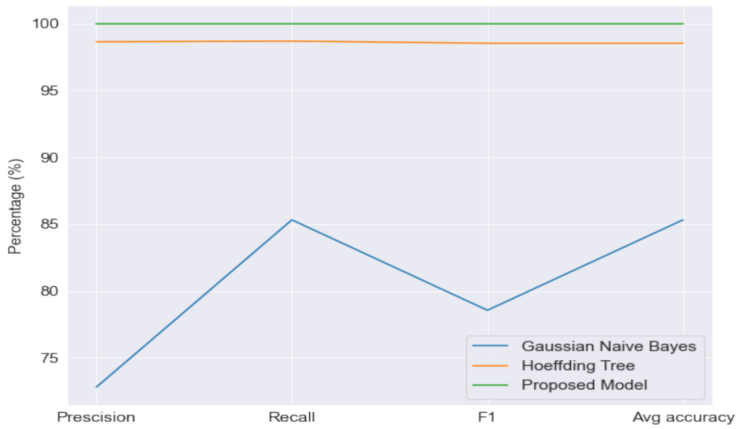

Figure 6. Accuracy of Gaussian NB, HT and proposed model on Fridge dataset.

and F1 scores of 92.36%, 92.36%, and 92.36%, respectively. Using the GPS IoT dataset, the Gaussian NB had an average accuracy of 77.60%, with precision, recall, and F1 scores of 60.21%, 77.60%, and 67.81%, respectively. Figure 5 shows the results of our model when evaluated using the modus IoT dataset.

Similarly, our proposed model has a superior average accuracy of 99.98% when it was evaluated using the fridge IoT dataset. The proposed model also recorded the same value for precision, recall and F1 score using the same dataset. Hoeffding tree algorithm recorded 98.63%, 98.68%, 98.52% and 98.52% for precision, recall, F1 score and average accuracy respectively. Gaussian NB recorded the least average accuracy. Recording an average accuracy of 85.31%. Gaussian NB also recorded the least values for precision, recall and F1 scores with 72.8%, 85.31% and 78.55% respectively. Figure 6 illustrates the outcomes of our model when tested against the fridge IoT dataset.

The experimental findings show that when tested using the motion IoT dataset, our proposed method performed better. The proposed ensemble model achieved an average accuracy of 99.98% with precision and recall of the same value whiles recording 99.97% for F1 score. The same dataset revealed that the Hoeffding tree recorded an average accuracy of 92.06% whiles recording a precision, recall and F1 score of 88.56%, 92.06% and 89.64% respectively. The average accuracy, precision, recall and F1 score recorded by Gaussian NB

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 September 2022 doi:10.20944/preprints202209.0058.v2

Figure 7. Accuracy of Gaussian NB, HT and proposed model on Motion dataset.

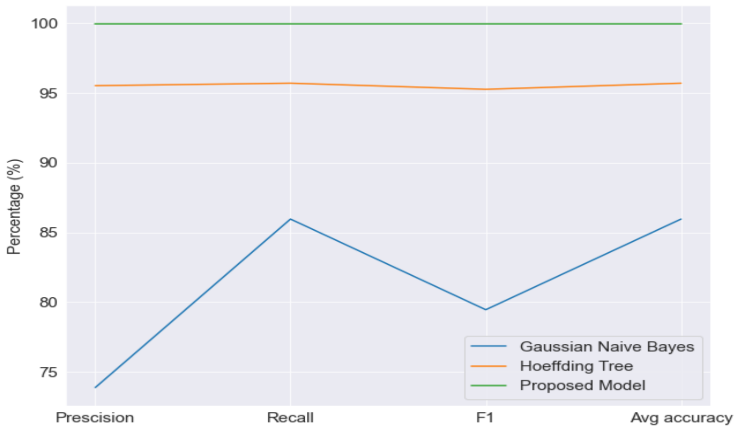

Figure 8. Accuracy of Gaussian NB, HT and proposed model on Garage dataset.

is 85.86%, 73.73%, 85.86% and 79.33% respectively. Figure 7 depicts the results of our model when tested against the motion IoT dataset.

When tested on the garage IoT dataset, our proposed ensemble model again had the highest precision, recall, F1 score, and average accuracy. Our proposed model recorded an average accuracy of 99.96% with the same value being recorded for precision, recall and F1 score. Hoeffding tree on the other hand recorded a precision, recall, F1 score and average accuracy of 95.52%, 95.70%, 95.26% and 95.70% respectively. Gaussian Naive Bayes recorded a precision, recall, F1 score and average accuracy of 73.88%, 85.96%, 79.46% and 85.96 respectively when it was evaluated using the garage IoT dataset. Figure 8 shows the results of our model when tested against the garage IoT dataset.

Evaluating our proposed model on the GPS tracker dataset, our model achieved a superior average accuracy of 99.97% with precision, recall and F1 score of 99.97% each. Hoeffding tree recorded an average accuracy of 98.29% whiles recording a precision, recall and F1 score of 98.36%, 98.29% and 98.08% respectively. Evaluating the Gaussian NB using the GPS IoT dataset revealed an average accuracy of 85.20% with precision, recall and F1 score of 88.26%, 85.20% and 82.02% respectively. Figure 9 shows the results of our model when evaluated using the GPS tracker IoT dataset.

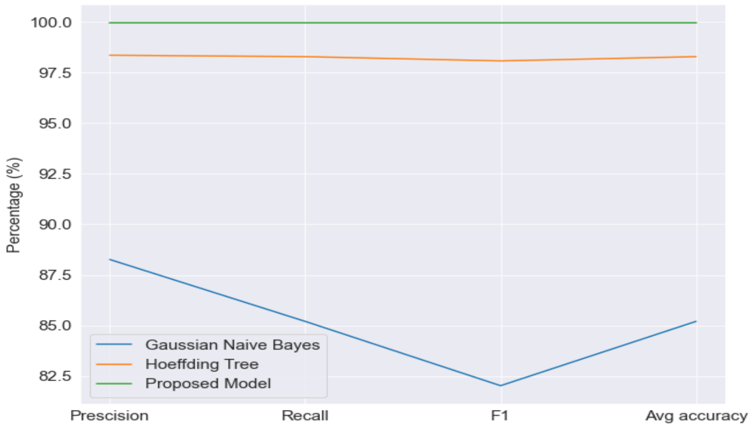

Figure 9. Accuracy of Gaussian NB, HT and proposed model on GPS Tracker dataset.

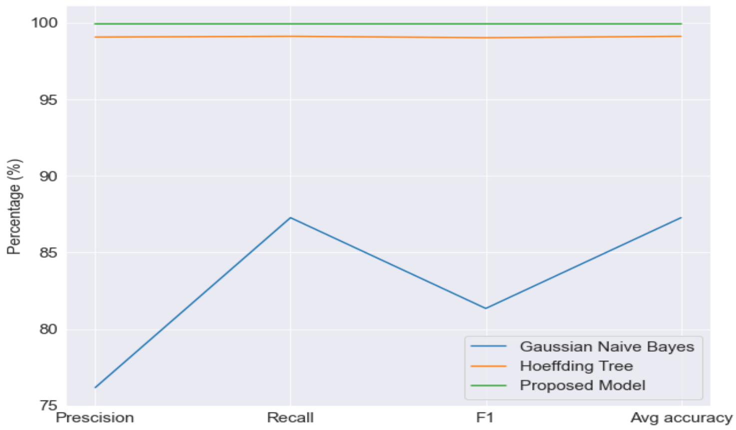

Figure 10. Accuracy of Gaussian NB, HT and proposed model on the Thermostat dataset.

The experimental result shows that when our proposed model is evaluated using the thermostat IoT dataset, the model achieved an average accuracy of 99.94% with a precision, recall and F1 score of 99.94% for each of them respectively. Gaussian NB showed an average accuracy of 87.27% whiles recording a precision, recall and F1 score of 76.17%, 87.27% and 81.34% respectively. Hoeffding tree showed an average accuracy of 99.12% with a precision, recall and F1 score of 99.07%, 99.12% and 99.03% respectively. Figure 10 shows the results of our model when tested against the thermostat IoT dataset.

We also evaluated our model on the weather IoT dataset which is one of the datasets found in the TON IoT dataset. The results show that Gaussian NB recorded a precision, recall, F1 and average accuracy of 80.02%, 86.08%, 76.65% and 86.08% respectively. On the other hand, the Hoeffding tree recorded a precision, recall, F1 and average accuracy of 98.47%, 98.37%, 98.30% and 98.37% respectively. However, our proposed ensemble technique recorded an average accuracy of 99.93% with a precision, recall and F1 score of 99.93% respectively. Figure 11 illustrates the outcomes of our model when tested against the weather IoT dataset.

The Fridge IoT dataset recorded the highest consumption of 650.11KB whiles the weather IoT dataset recorded the lowest memory consumption of 122.38KB. The results of the memory consumption of the model proposed in this study show that even at the

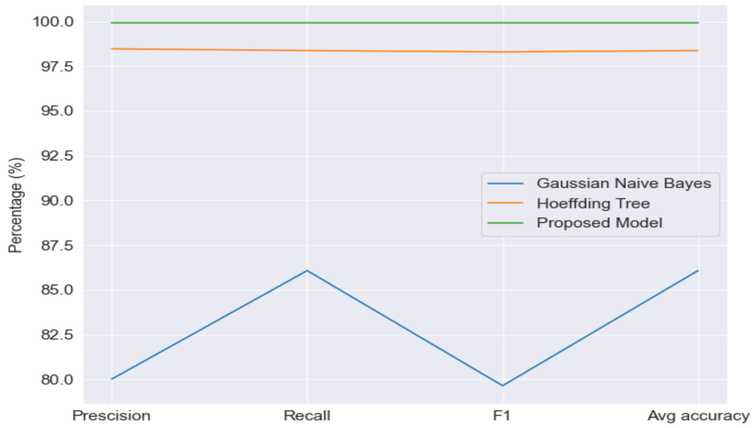

Figure 11. Accuracy of Gaussian NB, HT and proposed model on Weather dataset.

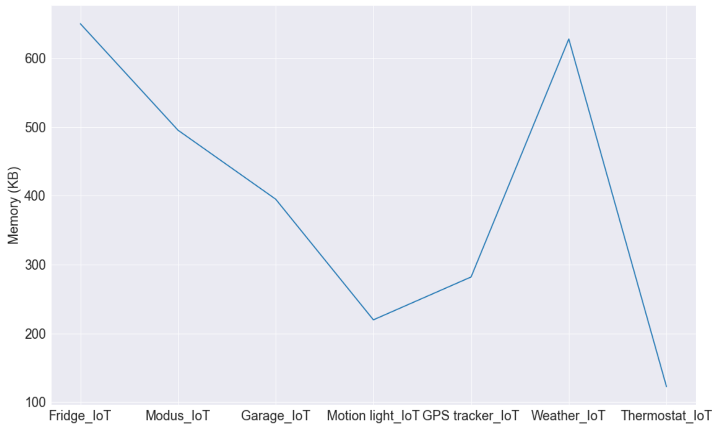

Figure 12. Memory consumption of our proposed model on various sub-datasets of TON IoT dataset

highest memory consumption the proposed IDS achieves a lightweight status and can potentially run on IoT devices without significantly affecting the available memory of these devices. The memory consumption of our proposed model is shown in figure 12.

The precision, recall and F1 score of Gaussian Naive Bayes, Hoeffding tree and our proposed model on different attack categories is shown in tables 5 and 6 below.

6. Conclusion

The security of the Internet of Things ecosystems is increasingly gaining great importance as a result of its numerous applications. The security of IoT systems has gone beyond encryption, authentication, and secured architecture. Recently, a lot of security-based research in IoT systems has been focused on detecting attacks and anomalies in network traffic. However, because of the computational constraints of IoT devices, IDS developed for traditional computing systems cannot be deployed in IoT environments. It is therefore expedient to design lightweight IDS that can be deployed on IoT devices. In this view, we used the incremental machine learning technique to design a lightweight IDS for IoT systems using incremental ensemble machine learning algorithms. Our proposed model was evaluated using the TON IoT dataset. The results show that our proposed model achieved a high average accuracy rate of 99.98%. The experimental results show the highest

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 September 2022 doi:10.20944/preprints202209.0058.v2

Table 5. Comparing the Precision (P), Recall (R) and F1 score of each model on each attack category

Fridge IoT dataset									
	Gaussian NB			Hoeffding Tree			Our proposed model		
Attack category	**P (%)**	**R (%)**	**F1 (%)**	**P (%)**	**R (%)**	**F1 (%)**	**P (%)**	**R (%)**	**F1 (%)**
Backdoor	0.00	0.00	0.00	98.73	99.38	99.06	99.97	99.76	99.86
DDoS	0.00	0.00	0.00	88.65	95.58	91.99	99.98	99.95	99.97
Injection	0.00	0.00	0.00	75.37	83.26	79.12	99.87	99.97	99.92
Normal	85.31	100.00	92.07	99.96	100.00	99.98	99.98	100.00	99.99
Password	0.00	0.00	0.00	90.09	92.99	91.52	99.97	99.97	99.97
Ransomware	0.00	0.00	0.00	43.83	27.29	33.64	99.59	99.69	99.64
XSS	0.00	0.00	0.00	100.00	11.31	20.33	100.00	99.41	99.71
Modus IoT dataset									
Backdoor	0.00	0.00	0.00	83.16	71.65	76.97	97.35	87.40	92.10
Injection	0.00	0.00	0.00	41.83	18.16	25.33	75.70	69.80	72.63
Normal	77.60	100.00	87.39	99.97	100.00	99.99	99.99	99.98	99.99
Password	0.00	0.00	0.00	46.42	68.78	55.43	71.61	89.00	79.36
Scanning	0.00	0.00	0.00	47.65	63.14	54.31	86.60	71.64	79.62
XSS	0.00	0.00	0.00	14.29	0.20	0.40	17.51	25.10	20.63
Garage IoT dataset									
Backdoor	0.00	0.00	0.00	99.55	94.32	96.86	99.67	99.91	99.79
DDoS	0.00	0.00	0.00	35.38	98.51	52.06	99.65	98.99	99.32
Injection	0.00	0.00	0.00	0.00	0.00	0.00	99.81	99.78	99.79
Normal	85.96	100.00	92.45	99.97	100.00	99.99	100.00	100.00	100.00
Password	0.00	0.00	0.00	66.51	47.23	55.23	99.85	99.88	99.87
Ransomware	0.00	0.00	0.00	0.00	0.00	0.00	99.90	99.83	99.86
Scanning	0.00	0.00	0.00	0.00	0.00	0.00	99.81	98.68	99.24
XSS	0.00	0.00	0.00	0.00	0.00	0.00	100.00	99.13	99.57
Motion light IoT dataset									
Backdoor	0.00	0.00	0.00	43.94	99.28	60.92	99.97	99.70	99.83
DDoS	0.00	0.00	0.00	0.00	0.00	0.00	99.96	99.94	99.95
Injection	0.00	0.00	0.00	0.00	0.00	0.00	99.93	99.95	99.94
Normal	85.86	100.00	92.39	99.95	100.00	99.97	99.98	100.00	99.99
Password	0.00	0.00	0.00	0.00	0.00	0.00	99.98	99.98	99.98
Ransomware	0.00	0.00	0.00	0.00	0.00	0.00	99.82	99.82	99.82
Scanning	0.00	0.00	0.00	0.00	0.00	0.00	99.77	99.77	99.77
XSS	0.00	0.00	0.00	0.00	0.00	0.00	100.00	99.11	99.55
GPS Tracker IoT dataset									
Backdoor	95.05	7.50	13.90	97.87	99.47	98.66	99.97	99.73	99.61
DDoS	98.77	8.65	15.91	69.66	92.50	79.47	99.87	99.97	99.92
Injection	16.77	46.00	24.58	79.02	79.78	79.40	99.84	99.88	99.86
Normal	88.42	96.57	92.31	99.96	100.00	99.98	99.98	100.00	99.99
Password	100.00	11.11	20.00	85.27	83.39	84.32	99.96	99.94	99.95
Ransomware	20.08	58.84	29.94	99.54	7.70	14.29	99.12	99.61	99.37
Scanning	100.00	0.18	0.36	100.00	20.36	33.84	100.00	99.64	99.82
XSS	10.92	13.52	12.08	0.00	0.00	0.00	100.00	95.84	97.88

memory consumption at 650.11KB and the lowest memory consumption at 122.38KB. The experimental result shows that our approach has led to the design of an IDS that has a high accuracy rate and a lightweight model that can potentially run on IoT devices. In the future, we plan to evaluate our approach to other IoT-based datasets and deploy our model on an IoT device to evaluate parameters such as CPU usage, memory, and energy consumption. Additionally, future work could consider exploring how concept drifts in these datasets could be handled.

Table 6. Comparing the Precision (P), Recall (R) and F1 score of each model on each attack category continuation

	Gaussian NB			Hoeffding Tree			Our proposed model		
Attack category	**P (%)**	**R (%)**	**F1 (%)**	**P (%)**	**R (%)**	**F1 (%)**	**P (%)**	**R (%)**	**F1 (%)**
Weather IoT dataset									
Backdoor	100.00	0.05	0.10	96.42	99.48	97.92	99.89	99.98	99.94
DDoS	0.00	0.00	0.00	83.59	93.53	88.28	99.89	99.84	99.87
Injection	0.00	0.00	0.00	69.59	89.18	78.18	97.39	99.88	98.62
Normal	86.08	100.00	92.52	99.97	100.00	99.98	100.00	100.00	100.00
Password	0.00	0.00	0.00	89.57	78.11	83.45	99.93	98.87	99.45
Ransomware	100.00	0.31	0.63	86.95	34.66	49.56	96.71	99.55	98.11
Scanning	0.00	0.00	0.00	100.00	38.56	55.66	95.19	86.01	90.37
XSS	0.00	0.00	0.00	100.00	39.84	56.98	100.00	94.11	96.97
Thermostat IoT dataset									
Backdoor	0.00	0.00	0.00	97.48	99.43	98.44	99.93	99.75	99.84
Injection	0.00	0.00	0.00	88.55	93.47	90.94	99.11	99.80%	99.45
Normal	87.27	100.00	93.20	99.95	100.00	99.97	99.98	100.00	99.99
Password	0.00	0.00	0.00	85.37	84.64	85.00	99.52	98.99	99.26
Ransomware	0.00	0.00	0.00	72.39	44.70	55.27	99.55	98.23	98.89
Scanning	0.00	0.00	0.00	0.00	0.00	0.00	69.33	85.25	76.47
XSS	0.00	0.00	0.00	100.00	1.34	2.64	100.00	94.65	97.25

Author Contributions: Contributions: Conceptualization, P.R.A., R.M. and J.R.; methodology, P.R.A. and J.R.; investigation, P.R.A., R.M. and I.G.; resources, P.R.A.; writing—original draft preparation, P.R.A.; writing—review and editing, R.M., I.G. and J.R.; supervision, R.M., I.G. and J.R.; project administration, P.R.A. and R.M.; funding acquisition, P.R.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Regional Scholarship and Innovation Fund (RSIF) and Google PhD Fellowship

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: [https://research.unsw.edu.au/projects/toniot-datasets].

Conflicts of Interest: The authors declare no conflict of interest

Abbreviations

The following abbreviations are used in this manuscript:

IDS	Intrusion Detection System
IoT	Internet of Things
ML	Machine Learning
HT	Hoeffding Tree
NB	Naive Bayes
DDoS	Distributed Denial of Service
XSS	Cross Site Scripting

References

1. Statista. • Global IoT and non-IoT connections 2010-2025 | Statista.
2. SAM Seamless Network. 2021 IoT Security Landscape - SAM Seamless Network.
3. Anderson, J.P. Computer security threat monitoring and surveillance. *Technical Report, James P. Anderson Company* **1980**.
4. Vacca, J.R. *Computer and information security handbook*; Newnes, 2012.
5. Jan, S.U.; Ahmed, S.; Shakhov, V.; Koo, I. Toward a lightweight intrusion detection system for the internet of things. *IEEE Access* **2019**, *7*, 42450–42471.

6. Latif, S.; e Huma, Z.; Jamal, S.S.; Ahmed, F.; Ahmad, J.; Zahid, A.; Dashtipour, K.; Aftab, M.U.; Ahmad, M.; Abbasi, Q.H. Intrusion detection framework for the internet of things using a dense random neural network. *IEEE Transactions on Industrial Informatics* **2021**.
7. Roy, S.; Li, J.; Choi, B.J.; Bai, Y. A lightweight supervised intrusion detection mechanism for IoT networks. *Future Generation Computer Systems* **2022**, *127*, 276–285.
8. Zhao, R.; Gui, G.; Xue, Z.; Yin, J.; Ohtsuki, T.; Adebisi, B.; Gacanin, H. A novel intrusion detection method based on lightweight neural network for internet of things. *IEEE Internet of Things Journal* **2021**.
9. Diwan, T.D.; Choubey, S.; Hota, H.; Goyal, S.; Jamal, S.S.; Shukla, P.K.; Tiwari, B. Feature entropy estimation (FEE) for malicious IoT traffic and detection using machine learning. *Mobile Information Systems* **2021**, *2021*.
10. Mukkamala, S.; Janoski, G.; Sung, A. Intrusion detection using neural networks and support vector machines. In Proceedings of the Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290). IEEE, 2002, Vol. 2, pp. 1702–1707.
11. Garcia-Teodoro, P.; Diaz-Verdejo, J.; Maciá-Fernández, G.; Vázquez, E. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security* **2009**, *28*, 18–28.
12. Denning, D.E. An intrusion-detection model. *IEEE Transactions on software engineering* **1987**, pp. 222–232.
13. Zarpelão, B.B.; Miani, R.S.; Kawakani, C.T.; de Alvarenga, S.C. A survey of intrusion detection in Internet of Things. *Journal of Network and Computer Applications* **2017**, *84*, 25–37.
14. Chaabouni, N.; Mosbah, M.; Zemmari, A.; Sauvignac, C.; Faruki, P. Network intrusion detection for IoT security based on learning techniques. *IEEE Communications Surveys & Tutorials* **2019**, *21*, 2671–2701.
15. Otoum, S.; Kantarci, B.; Mouftah, H.T. A novel ensemble method for advanced intrusion detection in wireless sensor networks. In Proceedings of the Icc 2020-2020 ieee international conference on communications (icc). IEEE, 2020, pp. 1–6.
16. Zhang, C.; Ma, Y. *Ensemble machine learning: methods and applications*; Springer, 2012.
17. Khoei, T.T.; Aissou, G.; Hu, W.C.; Kaabouch, N. Ensemble learning methods for anomaly intrusion detection system in smart grid. In Proceedings of the 2021 IEEE International Conference on Electro Information Technology (EIT). IEEE, 2021, pp. 129–135.
18. Schapire, R.E. The strength of weak learnability. *Machine learning* **1990**, *5*, 197–227.
19. Jacobs, R.A.; Jordan, M.I.; Nowlan, S.J.; Hinton, G.E. Adaptive mixtures of local experts. *Neural computation* **1991**, *3*, 79–87.
20. Benczúr, A.A.; Kocsis, L.; Pálovics, R. Online machine learning in big data streams. *arXiv preprint arXiv:1802.05872* **2018**.
21. Khannoussi, A.; Olteanu, A.L.; Labreuche, C.; Narayan, P.; Dezan, C.; Diguet, J.P.; Petit-Frère, J.; Meyer, P. Integrating operators' preferences into decisions of unmanned aerial vehicles: multi-layer decision engine and incremental preference elicitation. In Proceedings of the International Conference on Algorithmic Decision Theory. Springer, 2019, pp. 49–64.
22. Mozaffari, A.; Vajedi, M.; Azad, N.L. A robust safety-oriented autonomous cruise control scheme for electric vehicles based on model predictive control and online sequential extreme learning machine with a hyper-level fault tolerance-based supervisor. *Neurocomputing* **2015**, *151*, 845–856.
23. Feng, F.; Chan, R.H.; Shi, X.; Zhang, Y.; She, Q. Challenges in task incremental learning for assistive robotics. *IEEE Access* **2019**, *8*, 3434–3441.
24. Muthukrishnan, S.; et al. Data streams: Algorithms and applications. *Foundations and Trends® in Theoretical Computer Science* **2005**, *1*, 117–236.
25. Henzinger, M.R.; Raghavan, P.; Rajagopalan, S. Computing on data streams. *External memory algorithms* **1998**, *50*, 107–118.
26. Yang, L.; Manias, D.M.; Shami, A. PWPAE: An Ensemble Framework for Concept Drift Adaptation in IoT Data Streams. In Proceedings of the 2021 IEEE Global Communications Conference (GLOBECOM). IEEE, 2021, pp. 01–06.
27. Shakhov, V.; Jan, S.U.; Ahmed, S.; Koo, I. On Lightweight method for intrusions detection in the Internet of Things. In Proceedings of the 2019 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom). IEEE, 2019, pp. 1–5.
28. Pan, J.S.; Fan, F.; Chu, S.C.; Zhao, H.Q.; Liu, G.Y. A Lightweight Intelligent Intrusion Detection Model for Wireless Sensor Networks. *Security and Communication Networks* **2021**, *2021*.
29. Reis, L.H.A.; Murillo Piedrahita, A.; Rueda, S.; Fernandes, N.C.; Medeiros, D.S.; de Amorim, M.D.; Mattos, D.M. Unsupervised and incremental learning orchestration for cyber-physical security. *Transactions on emerging telecommunications technologies* **2020**, *31*, e4011.
30. Tabassum, A.; Erbad, A.; Mohamed, A.; Guizani, M. Privacy-preserving distributed IDS using incremental learning for IoT health systems. *IEEE Access* **2021**, *9*, 14271–14283.
31. Jadhav, S.D.; Channe, H. Comparative study of K-NN, naive Bayes and decision tree classification techniques. *International Journal of Science and Research (IJSR)* **2016**, *5*, 1842–1845.
32. McCallum, A.; Nigam, K.; et al. A comparison of event models for naive bayes text classification. In Proceedings of the AAAI-98 workshop on learning for text categorization. Citeseer, 1998, Vol. 752, pp. 41–48.
33. Shalev-Shwartz, S.; Ben-David, S. *Understanding machine learning: From theory to algorithms*; Cambridge university press, 2014.
34. Mitchell, T.M.; Mitchell, T.M. *Machine learning*; Vol. 1, McGraw-hill New York, 1997.
35. Bustamante, C.; Garrido, L.; Soto, R. Comparing fuzzy naive bayes and gaussian naive bayes for decision making in robocup 3d. In Proceedings of the Mexican International Conference on Artificial Intelligence. Springer, 2006, pp. 237–247.
36. Hulten, G.; Spencer, L.; Domingos, P. Mining time-changing data streams. In Proceedings of the Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, 2001, pp. 97–106.

37. Wang, H.; Fan, W.; Yu, P.S.; Han, J. Mining concept-drifting data streams using ensemble classifiers. In Proceedings of the Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003, pp. 226–235.
38. Montiel, J.; Halford, M.; Mastelini, S.M.; Bolmier, G.; Sourty, R.; Vaysse, R.; Zouitine, A.; Gomes, H.M.; Read, J.; Abdessalem, T.; et al. River: machine learning for streaming data in Python **2021**.
39. Alsaedi, A.; Moustafa, N.; Tari, Z.; Mahmood, A.; Anwar, A. TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. *IEEE Access* **2020**, *8*, 165130–165150.
40. Abu Al-Haija, Q.; Al Badawi, A.; Bojja, G.R. Boost-Defence for resilient IoT networks: A head-to-toe approach. *Expert Systems* **2022**, p. e12934.
41. Khan, M.A.; Khan Khattk, M.A.; Latif, S.; Shah, A.A.; Ur Rehman, M.; Boulila, W.; Driss, M.; Ahmad, J. Voting classifier-based intrusion detection for iot networks. In *Advances on Smart and Soft Computing*; Springer, 2022; pp. 313–328.
42. Lo, W.W.; Layeghy, S.; Sarhan, M.; Gallagher, M.; Portmann, M. E-GraphSAGE: A Graph Neural Network based Intrusion Detection System for IoT. In Proceedings of the NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2022, pp. 1–9.
43. Gad, A.R.; Nashat, A.A.; Barkat, T.M. Intrusion detection system using machine learning for vehicular ad hoc networks based on ToN-IoT dataset. *IEEE Access* **2021**, *9*, 142206–142217.