applied sciences

MDPI

Article

AA-HMM: An Anti-Adversarial Hidden Markov Model for Network-Based Intrusion Detection

Chongya Song *, Alexander Pons and Kang Yen

Department of Electrical and Computer Engineering, Florida International University, Miami, FL 33174, USA; apons@fiu.edu (A.P.); yenk@fiu.edu (K.Y.)

* Correspondence: ysong024@fiu.edu; Tel.: +1-786-537-7990

Received: 13 November 2018; Accepted: 26 November 2018; Published: 28 November 2018

Featured Application: This algorithm may be applied to light-weight/personal security products to provide adaptive intrusion detection capability with low latency.

Abstract: In the field of network intrusion, malware usually evades anomaly detection by disguising malicious behavior as legitimate access. Therefore, detecting these attacks from network traffic has become a challenge in this an adversarial setting. In this paper, an enhanced Hidden Markov Model, called the Anti-Adversarial Hidden Markov Model (AA-HMM), is proposed to effectively detect evasion pattern, using the Dynamic Window and Threshold techniques to achieve adaptive, anti-adversarial, and online-learning abilities. In addition, a concept called Pattern Entropy is defined and acts as the foundation of AA-HMM. We evaluate the effectiveness of our approach employing two well-known benchmark data sets, NSL-KDD and CTU-13, in terms of the common performance metrics and the algorithm's adaptation and anti-adversary abilities.

Keywords: network intrusion detection; adversarial setting; Anti-Adversarial Hidden Markov Model (AA-HMM); evasion patterns; dynamic window (DW); threshold (TH); pattern entropy (PE); adaptability

1. Introduction

In the practical deployment of Network Intrusion Detection System (NIDS) in the industry, there is an imbalance phenomenon—almost only signature-based detectors are being used, which scan characteristic byte sequences of the network traffic [1]. This situation is somewhat striking, especially when considering that Machine Learning (ML) has successfully been implemented in many other areas of computer science, often resulting in large-scale deployments in the commercial world. Examples from these domains include product recommendation systems (such as those used by Amazon [2] and Netflix [3]), optical character recognition systems [4], natural language translation [5], and spam detection [6], which is closer to the NIDS scenario [1].

One of the important reasons causing this imbalance phenomenon is that ML-based NIDS is working in an adversarial environment, which makes detection tasks challenging due to the presence of adaptive and intelligent adversaries who can carefully manipulate the attacking payload to evade detection. These evasion attacks undermine the underlying assumption of ML—the stationarity (the same distribution) of data for training and testing [7]. As a larger number of novel on-line services are emerging, the patterns of legitimate behaviors have become diversified, which in turn blur the boundary between normal and anomaly patterns [8]. Consequently, the existing ML-based NIDS cannot attain the required the industry-level performance attributed to the aforementioned two reasons.

Technically, the ML algorithms currently being studied and created for NIDS can be classified into four categories [9,10]: (1) ensemble-based, (2) clustering-based, (3) deep-learning, and (4) hybrid. However, each one has its disadvantages. (1) The performance of ensemble-based classifiers is highly unpredictable on unseen samples due to the high and non-eliminable correlations among the base classifiers [11,12]. (2) Since clustering-based models are unsupervised classifiers, their performance is not reliable as they classify samples without learning knowledge from the true labels. In many cases, a clustering algorithm is usually used as a component of a newly proposed classifier instead of being used as an individual classifier, such as in [13], who created a new model by combining the advantages of a clustering and ensemble algorithms. In practice, a model that solely relies on the ensemble or clustering technique should not be the best choice for NIDS, because deploying such a model would have a higher possibility of causing unexpected and serious damage to an organization due to its highly erratic generalization abilities on future packets. (3) Although deep-learning classifiers outperform the aforementioned two types of classifiers in terms of detection rate and stability, deploying a deep-learning model in practice may largely reduce network throughput because it usually suffers from the issue of high latency [14]. The efficiency problem might be resolved by quantum computers in the future [15], but deep-learning should not be considered the best choice for NIDS in terms of efficiency based on the current circumstances. (4) There are many enhanced models [16–23] classified as hybrid-type, which perform better by either combining the existing approaches in other domains (e.g., fuzzy logic) or creating novel mechanisms (e.g., feedback variables) specifically for NIDS (i.e., our AA-HMM), based on a specific shallow algorithm (e.g., decision tree) [14]. The temporary disadvantage of hybrid classifiers is that they need to be tested on various data sets to verify their stability. However, this long-term evaluation process is necessary for all newly proposed algorithms. In conclusion, it is necessary to design a new accurate and efficient hybrid-type algorithm for anomaly-based NIDS in practice.

The remainder of this paper is structured as follows. Section 2 presents the motivation provided by the four requirements for an applicable NIDS, based on the analysis of the disadvantages of the existing algorithms in Section 1. Section 3 introduces five fundamental concepts and terminologies for the designed AA-HMM feedback mechanism. Section 4 illustrates the AA-HMM architecture, as well as the principles and approaches of achieving adaptive and anti-adversarial capabilities with the aid of the five basic concepts in Section 3. Section 5 demonstrates AA-HMM at the implementation/coding level: (1) variables, (2) ensemble procedure, (3) pseudocode, and (4) workflow. Section 6 discusses some appropriate evaluation metrics and defines a new evaluation metric called Efficiency Matrix (EM), specific for intrusion detection problems, which reflects the security level of a NIDS in practice. Section 7 evaluates the AA-HMM on the NSL-KDD and CTU-13 data sets when adopting default settings and setting the initial matrices (transition and emission) with balanced parameters (Section 7.2.1), so our primary goal was to demonstrate the designed mechanisms of adaptive and anti-adversarial through interpreting the trajectories of created variables such as Dynamic Window.

2. Motivation

According to the analysis of the disadvantages of the existing ML algorithms with respect to practical deployment (see Section 1), we determined four requirements that are necessary for an applicable NIDS: (1) high detection rate (or low bias) on the trained patterns, (2) online-learning ability for tackling the unseen (including evasion) attacks and patterns, (3) high stability (or low variance) for ensuring the expected performance can be achieved while avoiding the possible severe damage (caused by the erratic performance) in practice, and (4) high efficiency for avoiding the NIDS to be the bottleneck of the network throughput. Consequently, we thought that a qualified base algorithm for a hybrid algorithm should not be a clustering or deep-learning algorithm, as they violate requirements (1) and (4), respectively. Moreover, in order to enable a common algorithm (e.g., decision tree, support vector machine (SVM), etc.) with online learning ability, we should rely on an unsupervised learning procedure to learn knowledge from the features. Typically, there are three options: (1) creating a

new unsupervised procedure and combining it with the base algorithm, (2) utilizing an existing unsupervised algorithm (e.g., K-Means) and combining it with the base algorithm, and (3) invoking the unsupervised learning procedure (i.e., Baum-Welch) that comes with the algorithm (i.e., the HMM) in a novel approach. Option (3) is better than options (1) and (2) in terms of performance and reliability because such procedures were specifically designed for the corresponding base classifiers and their performance has been verified by researchers for many years.

Overall, the Hidden Markov Model (HMM) is one of the algorithms that perfectly meets all the four aforementioned requirements for NIDS. Both theoretical and empirical results have shown that the HMM is capable of representing probability distributions corresponding to complex real-world phenomena in terms of simple and compact models [24]. The HMM is superior due to the strong online learning ability, which is driven by a reliable unsupervised learning procedure called Baum-Welch (BW) [25]. This has been verified by the success of HMM in various practical applications, where it has become the predominant methodology for designing Automatic Speech Recognition systems (ASR) [26]. Likewise, the HMM has been successfully applied to other fields, such as signature verification, communication and control, bioinformatics, computer vision, and network security. A growing number of HMM-based NIDS have been developed in recent years, which have been applied either to misuse detection to model a predefined set of attacks, or in anomaly detection to model normal behavior patterns, such as in [16]. Most importantly, the HMM-based applications in anomaly and misuse detection have emerged in both the main categories of Intrusion Detection System (IDS): (1) host-based IDS (HIDS) in [17] and (2) network-based IDS (NIDS) in [18,19,27]. The HMM has recently begun to emerge in applications of Wireless IDS (WIDS) [20]. Given the most recent investigation in [28], the HMM is the algorithm that requires the least time of adoption for building NIDS. Therefore, adopting the HMM is not only a good choice in terms of the performance, but also a major progress in the ML-based NIDS research due to its unique qualities.

The performance of HMM will not be improved unless we invoke the BW in a novel way, so we needed to design a feedback mechanism as a metric (represented as a set of variables on the implementation level) to determine the status of the underlying traffic pattern, so that we could update the model to a local optimal state and enhance its performance through invoking the BW based on the pattern status. Consequently, Sections 3–6 provide demonstrations of concrete solutions and the corresponding deductions of designing such an effective feedback mechanism (a set of variables), which can be divided into six challenges:

(1) What are the feedback variables (Section 3)?
(2) Why these feedback variables are selected (Sections 3 and 4)?
(3) How can these feedback variables be captured (Section 4)?
(4) How can these feedback variables be measured/quantified (Section 4)?
(5) How can these feedback variables be used (Sections 4 and 5)?
(6) How can the BW be invoked based on these feedback variables to improve the performance of the algorithm (refer to Sections 4 and 5)?

3. Foundation

This section provides a detailed description of the AA-HMM in terms of foundational concepts. It defines and introduces five principle concepts that are necessary to fully understand the underlying approach of the AA-HMM: (1) Pattern Entropy (PE), (2) PE Reduction (PERD), (3) Window Width (WW), (4) Local Optimal Window Width (LOWW), and (5) Dynamic Window (DW). Among these concepts, (1) and (2) are the basis of (3) and (4). In addition, (5) is established based on (3) and (4), which acts as the core feedback mechanism that enables the model to adaptively adjust according to dynamic network patterns.

3.1. Pattern Entropy

The fundamental concept of the proposed algorithm is the underlying PE. It is a metric for quantifying the entropy/complexity of a sequence of network data samples (packets or flows). Considering the dynamic and diverse nature of network traffic, an appropriate definition should include the numbers of attacks and attackers, but also consider the sequential and diverse information of samples.

Assuming a labeled intrusion data set, the total number of anomaly and normal samples (packets or flows) are X and Y, respectively; the total number of attackers and legitimate users are A and B, respectively; and the types of anomaly and normal samples are represented as P and Q, respectively. Then, the entropy of this data set or its pattern PE can be defined as:

$$\mathrm{PE}(\text{entire data set}) = (\mathrm{AP})^{\mathrm{X}} + (\mathrm{BQ})^{\mathrm{Y}} \tag{1}$$

Since every anomaly sample might be launched by any attacker, A represents the entropy of every malicious sample in terms of the possible attackers. Furthermore, every attacker may launch any type of attack; hence, P represents the entropy of every attacker in terms of the possible attacks. Therefore, the term AP should be interpreted as the total entropy of every malicious sample and the term $(\mathrm{AP})^{\mathrm{X}}$ is defined as the PE of malicious pattern due to the presence of X malicious samples. After applying the same method to calculate the PE of normal pattern $(\mathrm{BQ})^{\mathrm{Y}}$, the total PE of the entire data set is expressed by adding the two PEs together. Note, normalization should not be applied because the PE is intended to reflect the complexity variation resulting from factors such as the length of sections, the number of malicious samples, etc.

3.2. PE Reduction

Given the definition of PE, if we only calculated the entropy of a subset, the value would be much lower than the original or entire samples. For instance, assume a data set with evenly distributed normal and anomaly samples. If we only calculated the PE on any 1/n subset, the variables X, Y, A, B, P, and Q would be reduced to X/n, Y/n, A/n, B/n, P/n, and Q/n, respectively. As a result, the PE of this subset should be calculated as:

$$\mathrm{PE}\left(\text{any } \frac{1}{\mathrm{n}} \text{ subset}\right) = \left(\left(\frac{\mathrm{A}}{\mathrm{n}}\right)\left(\frac{\mathrm{P}}{\mathrm{n}}\right)\right)^{\frac{\mathrm{X}}{\mathrm{n}}} + \left(\left(\frac{\mathrm{B}}{\mathrm{n}}\right)\left(\frac{\mathrm{Q}}{\mathrm{n}}\right)\right)^{\frac{\mathrm{Y}}{\mathrm{n}}} = \sqrt[\mathrm{n}]{\frac{(\mathrm{AP})^{\mathrm{X}}}{\mathrm{n}^{2\mathrm{X}}}} + \sqrt[\mathrm{n}]{\frac{(\mathrm{BQ})^{\mathrm{Y}}}{\mathrm{n}^{2\mathrm{Y}}}} \tag{2}$$

Since the term $(\mathrm{AP})^{\mathrm{X}}$ would considerably decrease after being divided by $\mathrm{n}^{2\mathrm{X}}$ and rooted by n, the first term in Equation (2) would be much smaller than that in Equation (1). The same comparison is also applicable to the second terms of the two equations. We concluded that, for any data set, the shorter the subset, the lower the PE.

3.3. Window Width

Since an HMM can be employed to predict samples section-by-section (e.g., classifying the first 20 samples, then the next 20 samples, until predicting all the samples), Window Width (WW) is defined as the number of samples in each section. Inferred from the definition of PE, shorter sections have lower PE and are easier to accurately predict. Therefore, splitting the entire data set into shorter subsets and then predicting them in order would effectively enhance the detection rate of sample sequences.

3.4. Local Optimal Window Width

To determine the correlation between the WW (length of section) and the corresponding accuracy, we performed extensive experiments on a variety of data sets (e.g., NSL-KDD, CTU-13, etc.): (1) building and testing nearly 20 HMMs (refer to [24,25] for the principle of HMM) with the same initial configuration on the same data sets; (2) each model sets up a unique WW—if a model's WW is

25 and a data set has 1000 samples, then this model will split the data set into 40 sections, each one containing 25 samples, and making predictions on the 40 sections (windows) in order. (3) The range of the tested WWs was 25 to 450 (steps of 25). Figure 1 provides a representation of the relationship between these models with different WWs and their corresponding accuracies.

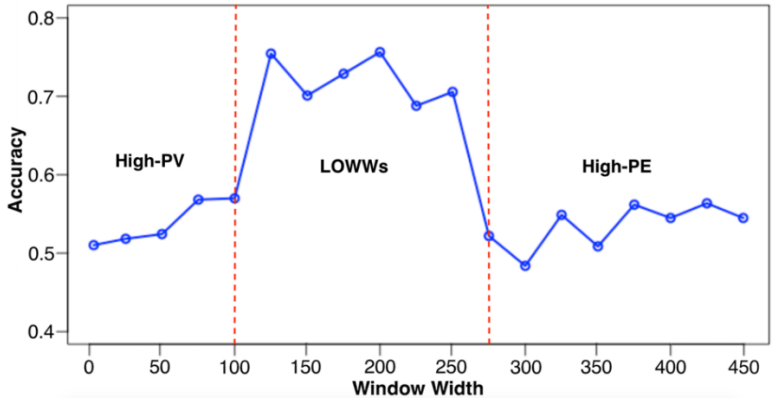

Figure 1. Correlation between Window Width (WW) and accuracy.

For the previous inference regarding the PE, longer WWs usually suffer from higher PEs and are harder to accurately predict, so the models with wide WWs (from 275 to 450) have low accuracies. Although the models with narrow WWs (from 10 to 100) are also inaccurate, this phenomenon does not contradict the inference of PE, but results from another vital factor called Pattern Variation (PV), which is understood as the samples' differences in terms of type and distribution between different windows. Since the HMMs interpret samples as hidden states and observations, the PV should be expressed as the differences between hidden states, as well as the differences between observations, in terms of type and distribution between adjacent windows. Therefore, a higher PV would produce a lower detection rate because it is difficult for a model with a fixed configuration to perform well on all windows (patterns). Consequently, the WWs that result in relatively high accuracies are called Local Optimal Window Widths (LOWWs). As Figure 1 shows, the LOWWs for this experiment were 125, 150, 175, 200, 225, and 250 samples in length. So, the next task for improving performance was to search for one of the LOWWs and set it as the model's WW.

3.5. Dynamic Window

Continuing with the above experiment result and further searching for the best model (WW = 125 samples, total number of windows = total number of samples divided by WW = 181, overall accuracy = 72.88%), we discovered that its performance varies from one window to another, as shown in Figure 2 (a similar result was found for all models). The windows (dots in clusters 1, 2, and 3) with low accuracies may need to be combined with adjacent windows to form wider windows or split into multiple windows, so that the patterns of newly created windows are more suitable for the current model's configuration. Furthermore, to maximize the performance, the WW should be treated as a variable that is always and continuously subject to change. Therefore, we designed a mechanism called the Dynamic Window (DW), which smartly searches and sets every WW as the LOWW according to the underlying pattern.

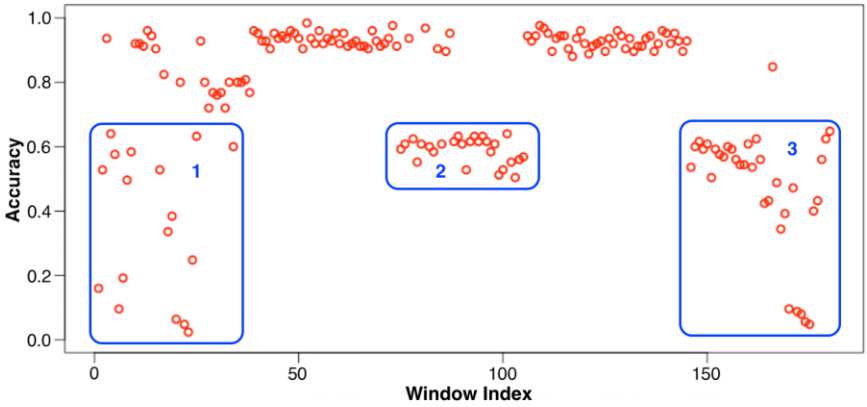

Figure 2. Correlation between accuracies of Dynamic Window (DW) with a fixed Window Width (WW = 125) and window indices.

4. Methodology

The DW is the key component of the AA-HMM; it utilizes two pairs of variables for adjusting the WW to LOWW based on the underlying pattern: (1) Model Difference (MD) and Difference Trend (DT); (2) Threshold (TH) and Threshold Controller (THC). The first variable pair enables the adaptability of the AA-HMM, whereas the second variable pair provides the model's anti-adversarial capabilities. We establish the overall conceptual and logical contribution of the previous concepts and these variable pairs in the following section.

4.1. Architecture

The architecture of the AA-HHM is shown in Figure 3, which was constructed upon five basic five: PE, PV, WW, LOWW, and DW. Given the study of [7], the detection ability or security level of an anomaly-based NIDS can be divided into three levels (from lowest to highest): (1) strong knowledge acquisition and decent generalization abilities, (2) strong generalization and decent adaptive abilities, and (3) strong adaptive and anti-adversarial abilities. The AA-HMM consists of three layers corresponding to these three security levels:

1. In order to achieve the lowest security level, the regular HMM was adopted as the base algorithm to learn the traffic pattern and make predictions.
2. To attain the second security level, a pair of feedback variables, called Model Difference (MD) and Difference Trend (DT), were designed to improve the adaptability of base HMM.
3. To achieve the top security level, the variable pair called Threshold (TH) and Threshold Controller (THC) were integrated to realize the required anti-adversary ability.

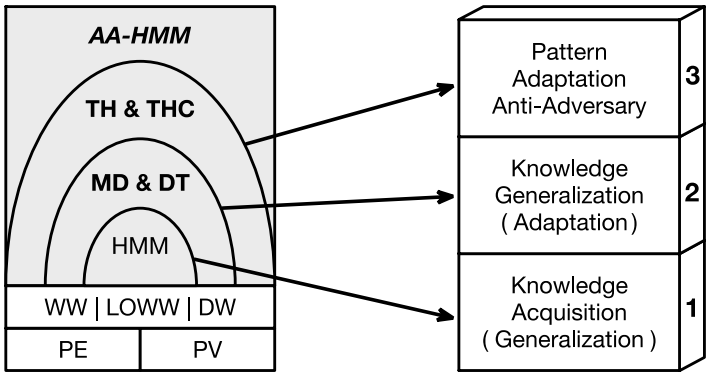

Figure 3. Anti-Adversarial Hidden Markov Model's (AA-HHM) overview architecture.

4.2. Adaptive Mechanism: MD and DT

It is important to comprehend the cause of rendering the low detection rate before designing any enhancement mechanism. As the upper part of Figure 4 shows, if the accuracy decreases, it indicates that the model's parameters are not well tuned to fit the pattern being predicated. Since the Baum-Welch (BW) procedure is responsible for updating the model, the decreased accuracy demonstrates that BW is not capable of updating the transition and emission matrices appropriately, which indicates that the underlying pattern is intensively fluctuating. An intensive fluctuating pattern usually means a stronger randomness of all the samples, which causes two negative factors for a window-based model (predict samples section-by-section, refer to Section 3.3.): (1) high PV where the type and distribution of samples between windows are changing frequently and (2) high PE where windows include more types of samples as the types of samples are changing frequently. As such, the two negative factors, high PE and PV (refer to Section 1 Introduction and Section 2 Motivation) are presented here as possible consequences of intensive pattern fluctuation.

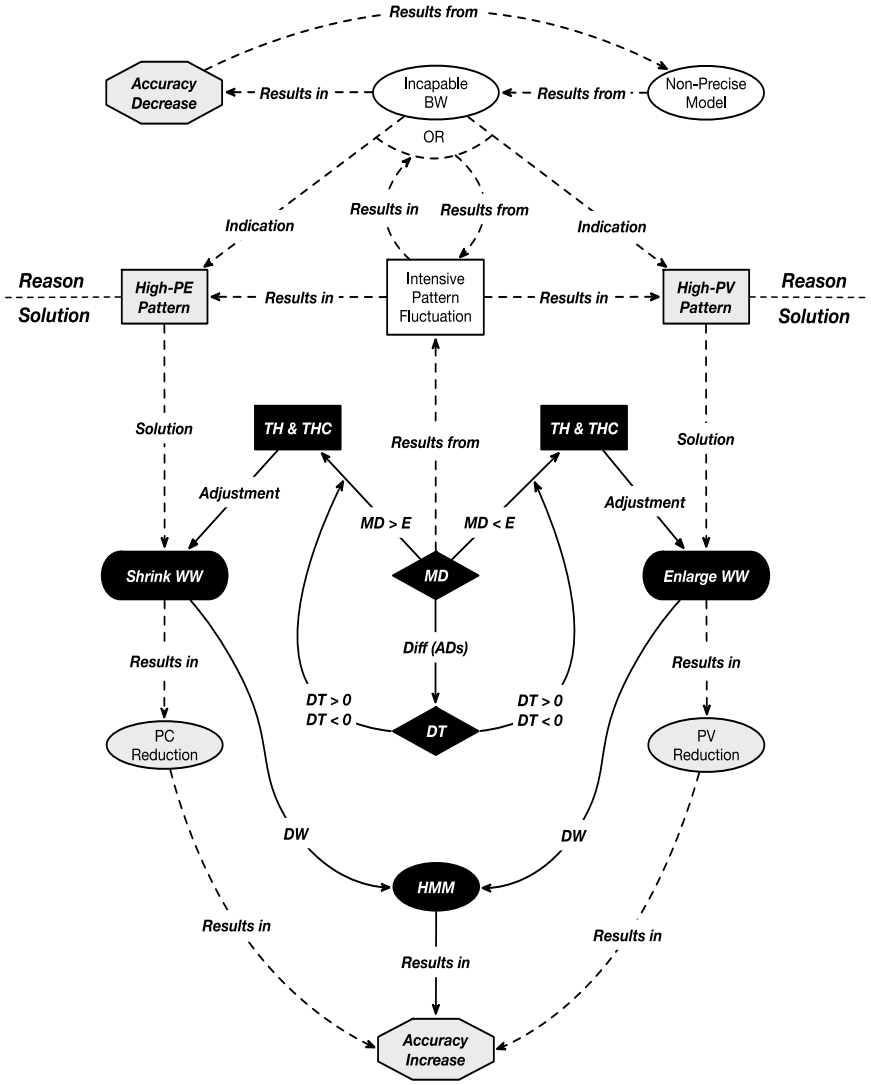

Figure 4. The logic chain of AA-HMM. * Shapes with black background are components of AA-HMM, Solid lines represent the interactions between components and dashed lines are relationship between concepts.

As stated, both negative factors are a result of the same phenomenon—an intensive pattern fluctuation that would require the BW to significantly update the model's parameters. Since BW is

limited in its ability to update the model to a local optimal state against intensive fluctuated pattern, the model's accuracy is reduced and can be detected using the feedback variable MD, which is defined as the quantitative model difference between adjacent windows:

$$\mathrm{MD}(\ \mathrm{M_n},\ \mathrm{M_{n+1}}) \ = \mathrm{Diff}[\ \mathrm{M_n}(\mathrm{W_n}),\ \mathrm{M_{n+1}}(\mathrm{W_{n+1}})\ ] \tag{3}$$

where $\mathrm{W_x}$ is the identity of a window and $\mathrm{M_y}$ represents a specific model y, then $\mathrm{M_y}(\mathrm{W_x})$ is model y's parameters (transition and emission matrices) after updating upon the pattern in window x. Therefore, the entire term $\mathrm{Diff}[\ \mathrm{M_n}(\mathrm{W_n}),\ \mathrm{M_{n+1}}(\mathrm{W_{n+1}})]$ is the difference between the two models in terms of parameters/configuration, which can be formally defined as, "a vector of differences calculated from consecutive transition and emission matrices in each iteration of the Baum-Welch procedure, which is calculated by summing the $\mathrm{L_2}-\mathrm{Norm}$ distances between consecutive transition and emission matrices" [29]. If the MD is larger than a certain pre-defined value E, it indicates that the predicted pattern is intensively fluctuating and the updated model would not be in a good state, which produces low accuracy. Therefore, in order to improve the accuracy, we needed to reduce the PE of the next window through reducing the WW, so that the BW could update the model in a more accurate state (Figure 4). If the MD is smaller than E, the accuracy is being maintained at a high level, indicating that the PE must be at a very low level. So, slightly enlarging the WW would not only keep the WW within the range of LOWWs (Figure 1), but also reduce the PV of the next window, which results in improved accuracy (see Section 2 Motivation). Furthermore, the amount of WW adjustment is based on the difference in magnitude between the MD and E, where the greater the difference, the greater the adjustment. As such, to consider the tendency factor, the adaptive variable DT is defined as the differences between adjacent MDs:

$$\begin{aligned} &\mathrm{DT_n}(\ \mathrm{MD_n},\ \mathrm{MD_{n+1}}\ ) = \ \mathrm{Diff}\ (\ \mathrm{MD_n}, \mathrm{MD_{n+1}}\ ) \\ &= \ \mathrm{Diff}\ [\ \mathrm{MD_n}(\ \mathrm{M_n},\ \mathrm{M_{n+1}}\ ), \mathrm{MD}_{n+1}(\ \mathrm{M_{n+1}}, \mathrm{M_{n+2}}\ )\ ] \\ &= \mathrm{Diff}\left\{ \begin{array}{c} \mathrm{Diff}\ [\ \mathrm{M_n}(\mathrm{W_n}),\ \mathrm{M_{n+1}}(\mathrm{W_{n+1}})\ ] \\ \mathrm{Diff}\ [\ \mathrm{M_{n+1}}(\mathrm{W_{n+1}}),\ \mathrm{M_{n+2}}(\mathrm{W_{n+2}})] \end{array} \right\} \end{aligned} \tag{4}$$

As Equation (4) shows, the DT is the difference between two MDs—the difference in the parameter' (Figure 5), which reflects the changing trend of the model's parameters between adjacent windows and acts as a calibrating metric for the WW adjustment. We define four sets of operations derived from the combinations of MD and DT:

- When MD > E, based on the difference magnitude between MD and E, the WW should be decreased to reduce the PE, then:
    - If DT > 0, based on the difference magnitude between the DT and 0, the WW should be decreased again because the DT indicates that the PE of recent windows has continued to increase.
    - If DT < 0, based on the difference in magnitude between the DT and 0, the WW should be increased because the DT indicates that the PE of recent windows has kept decreasing.
- When MD < E, based on the difference in magnitude between MD and E, the WW should be increased to reduce the PV, then:
    - If the DT > 0, based on the difference in magnitude between DT and 0, the WW should be increased again because the DT indicates that the PV of recent windows has kept increasing.
    - If DT < 0, based on the difference magnitude between the DT and 0, the WW should be decreased because the DT indicates that the PV of recent windows has kept decreasing.

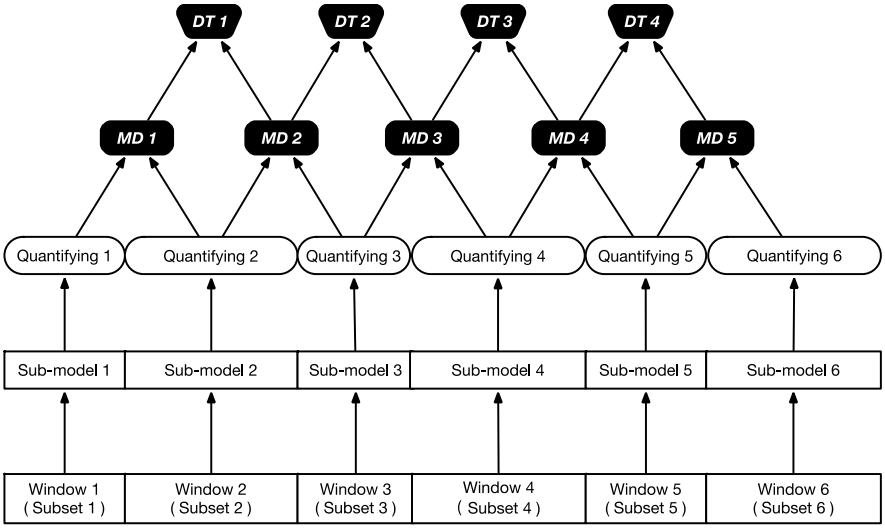

Figure 5. Adaptive mechanism—Model Difference (MD) and Difference Trend (DT).

The variables MD and DT are indicators and reveal the underlying pattern information (Figure 4), which is used with the four operations to adjust the DW and WW based on the patterns, resulting in the ability of the model to correlate with the pattern of the samples. The pattern information is successfully extracted, stored, and utilized by the MD and DT. An overview of the architecture (assuming that there are six windows in total) of the adaptability of the approach is depicted in Figure 5.

4.3. Anti-Adversarial (AA) Mechanism: TH and THC

The AA-HMM can resolve three types of evasion attacks. (1) For any kind of evasion technique that is not specific to the ML-based NIDS, selecting a base algorithm with a strong generalization ability is always a sensible decision because the evasion difficulty is improved once adopted. As a probability-based algorithm, the HMM is inherently capable of identifying samples that have never seen before. (2) Optimal Evasion (OE) is one of the most recent types of evasion attacks that specifically targets ML-based NIDS. OE creates malicious samples by minimally manipulating the initial attack until it successfully evades detection [30,31]. For example, OE may successfully evade detection by only modifying the value of a numeric feature from 10,000 (can be detected) to 10,001 (cannot be detected). Since this malicious sample is extremely close to its initial sample (normal version), an anomaly-based model may misclassify it as normal. To counteract this type of evasion, we simply discretized all the numeric features and aggregated nominal features into three to five bins, so that any manipulation of a sample would be amplified to a level that could be detected by the model. In addition, this pre-processing approach can improve the HMM's general accuracy against all types of samples (Section 7 Experiment). (3) Some sophisticated attackers may evade detection by sending some manipulated samples to the feedback-enabled NIDS, which gradually train the detector to a state that is not capable of identifying any attack launched later. If you defend against this type of attack by disabling the feedback component, the accuracy would be largely decreased due to the misclassifications of all other types of samples (including normal ones).

Accordingly, to enable the model to identify the intensions of attackers, we defined a variable called Threshold (TH), which represents the range of DW. The logic chain of TH (the relationship between DW and TH) is shown in Figure 6.

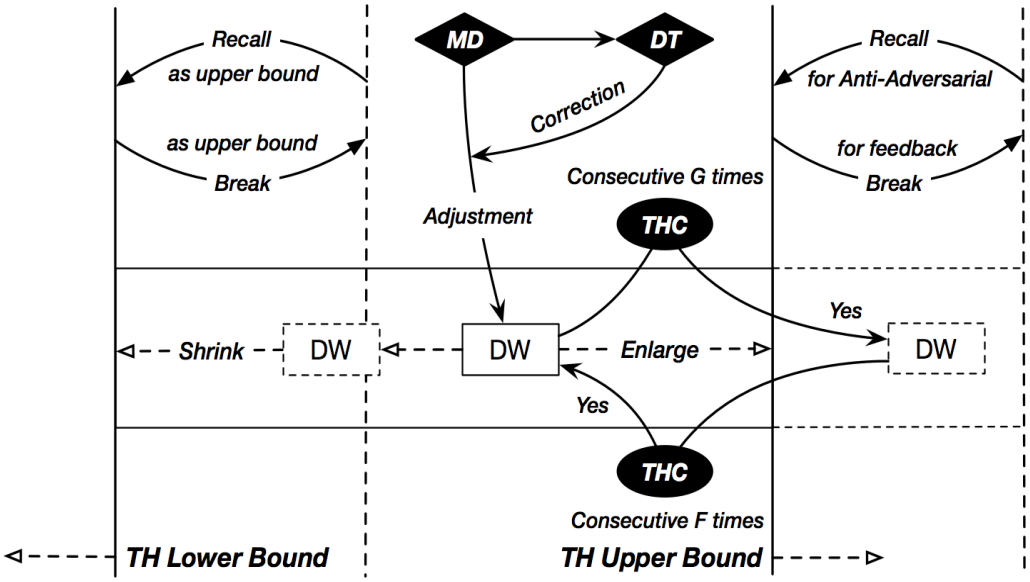

Figure 6. Overview logic chain of threshold.

From Figure 4, as an anti-adversarial variable, the TH acts as a switch that is responsible for making the final DW adjustment. As Figure 6 shows, the DW is bounded by the lower and upper bounds of TH. To understand the acceptable range of TH and the safeguards against improper values, if the MD and DT contribute to enlarging the DW to a value that is higher than the current upper bound, the TH would refuse this adjustment request and set the DW to the middle point of the current upper and lower bounds, unless it consecutively receives the same request for F times (a threshold/counter for increasing the upper bound; Table 1). Correspondingly, for the TH recall procedure, if the MD and DT contribute to reducing the DW to a value that is lower than the current lower bound, the TH would also refuse this adjustment request and set the DW to the middle point of the current upper and lower bounds, unless it consecutively receives the same request for G times (a threshold/counter for recall the increased upper bound; Table 1). There are two reasons for these operations. (1) Since the network pattern is extremely irregular, "spurs" (normal, but temporary pattern fluctuation) can be found anywhere. If we set the model response and adjusted the DW to any arbitrary value given these spurs, the DW might be enlarged or reduced to either a too large or small a value. The model would suffer from either high PE or PV and the accuracy would considerably decrease. Therefore, the values of F and G were set as the metrics of TH to ignore these spurs and stabilize the accuracy. (2) Similar to the spur activity, malicious traffic that intends to misleadingly train the NIDS would also be omitted once an appropriate value of F and G are set based on the characteristics of the specific attack.

Furthermore, if the TH consecutively triggers the request to enlarge the DW for F times due to the normal pattern change (necessary feedback operation), instead of directly adjusting the DW to the intended WW, the DW would be adjusted to the average of the current upper bound plus the average of the sum of the intended WWs during the past F consecutive windows. Correspondingly, the upper bound would be "broken" and increased by the average of the sum of the intended WWs during the past F consecutive windows. The lower bound would be increased by the same step as the upper bound. The similar operations would also be applied to the DW recall procedure. As a result, this series of operations further enhance the model's adaptive abilities toward normal and intensive pattern fluctuations.

Once the attackers successfully misleading the model, the next two things they would do are to stop sending the malicious training traffic and launch the real attacks that cannot be detected by the misled model. Notably, the pattern of malicious training traffic would be different from that of the actual attacks. Most importantly, even if the AA-HMM is misled, it can automatically recover itself to the normal state using the recall operation—the extended sections of upper and lower bounds

would be recalled once the malicious training samples have not been consecutively received G times. The values of F and G can be flexibly set based on the defensive strategy and the desired security level.

Table 1. Variables of AA-HMM.

Name	Description	Default Value
window width	width of the dynamic window	10
window base line	minimum width of the dynamic window	10
threshold lower bound	lower bound of the dynamic window	10
threshold upper bound	upper bound of the dynamic window	1000
threshold break	a counter that records the number of consecutive requests of increasing the threshold upper bound	starts from 0
F	threshold for increasing the upper bound	3
threshold recall	a counter that records the number of consecutive times of the increased section of threshold has not been used	starts from 0
G	threshold for recall the increased upper bound	3
threshold controller	controlling the difficulty of threshold adjustment	0
MD vector	levels of resizing the window based on the MD	from 1.5 to 0.5 step by −0.1
DT vector_en	he levels of enlarging the window based on the DT	from 1 to 1.2 step by 0.02
DT vector_sh	levels of reducing the window based on the DT	from 1 to 0.8 step by −0.02
levels	graininess of WW adjustment for MD and DT	from 0 to 1 step by 0.1

Particularly, repeatedly enlarging or reducing the DW is not a wise strategy due to the high PE or PV, even if the MD and DT trigger the DW adjustment because the DW cannot be rapidly decreased or increased to common values once this extremely fluctuated pattern passes. Accordingly, to tackle this kind of extreme case, another feedback variable called Threshold Controller (THC) was introduced to control the difficulty of breaking the bounds of TH. THC is responsible for counting the times of breaking bounds (Figure 6). The more breaking operations are accomplished, the more difficult to break the current bounds again. For instance, if the model increases the upper bound two times (THC = 2), the current upper bound would not be increased again unless the TH consecutively receives F + THC times requests for increasing TH, which is triggered by the DW increasing operation.

5. Implementation

This section presents the AA-HMM on the lowest/coding level, which involves a comprehensive set of descriptions and settings for all variables. Since most variables work well on common data sets using their default settings, we employed the AA-HMM using these default configurations or adjusted a few significant variables, such as the MD vector, for improved performance.

5.1. Variables

The AA-HMM was implemented using R Language and its essential procedures, such as Forward-Backward (FB) and Baum-Welch (BW), were invoked from an existing package named HMM [29]. Both the FB and BW do not suffer from the problem of floating-point underflow because the package's implementation avoids it by converting the probability values into logarithms during the calculation and then converting them back at the end [32]. Table 1 presents the all variables and their default values.

5.2. Ensemble

Although many pre-processing algorithms, such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), effectively reduce the dimensions of a data set, these approaches typically result in information (pattern or knowledge) losses [25]. Therefore, we wanted to achieve the same goal in an opposing manner: (1) keeping as many features as possible; (2) training a sub-model for

each feature, separately; and (3) obtaining the final prediction results through ensemble. This approach not only avoids the high-dimensional issue, but also has two extra advantages:

1. Even if the number of qualified sub-models is not enough for ensemble, we can build multiple dummy sub-models with varied parameters (e.g., the WW) on the same feature. Since the trajectories of different DWs would vary, the prediction results toward the same sample within different windows would be distinctive, which can be used to ensemble the result.
2. Since the sub-models are working concurrently, and considering the time cost of ensemble procedure remains constant, the total time cost is only bounded by the sub-model with the highest time cost: TotalTimeCost(TTC) = Max[T(M_1), T(M_2), ... T(M_n)], where the function T(M_x) represents the time cost of each sub-model.

5.3. Pseudocode

AA-HMM (default setting, minor variables are omitted)

Initial the model

While (until reach the end of the data set)

DW moves forward by the step of the current WW

BW updates the model based on the current window

Quantifying the current model $(\mathbf{M_n})$

Calculating $\mathbf{MD_m} = (\mathbf{M_{n-1}}, \mathbf{M_n})$

Adjusting Window Width based on $\mathbf{AD_m}$ **:**

```
if MD in (0, 0.1) then
        WW = WW × MD vector [0]
else if MD in (0.1, 0.2) then
        WW = WW × MD vector [1]
else if MD in (0.2, 0.3) then
        WW = WW × MD vector [2]
……
else MD in (1, ∞) then
        WW = WW × MD vector [10]
end if
```

Calculating $\mathbf{DT} = (\mathbf{MD_{m-1}}, \mathbf{MD_m})$

Adjusting Window Width based on DT:

```
if DT in (0, 0.1) then
        WW = WW × DT vector_en [0]
else if DT in (0.1, 0.2) then
        WW = WW × DT vector_en [1]
else if DT in (0.2, 0.3) then
        WW = WW × DT vector_en [2]
……
else DT in (1, ∞) then
        WW = WW × DT vector_en [10]
end if

if DT in (−0.1, 0) then
        WW = WW × DT vector_rd [0]
else if DT in (−0.2, −0.1) then
        WW = WW × DT vector_ rd [1]
else if DT in (−0.3, 0.2) then
        WW = WW × DT vector_ rd [2]
……
```

else DT in (−∞, −1) then
WW = WW × DT vector_ rd [10]
end if

Adjusting WW & TH:
if TH break > F + THC then
new WW = {current upper bound + average [sum (intended WWs in recent F times)]}/2
new TH upper bound = {current upper bound + average [sum (intended WWs in recent F times)]/2}
new TH lower bound = current lower bound + increased section of upper bound
THC ++
else then
new WW = (current lower bound + current upper bound)/2
end if

if TH recall > G then
new WW = {current lower bound − average [sum (intended WWs in recent G times)]/2}
new lower bound = {current lower bound − average [sum (intended WWs in recent G times)]}/2
new upper bound = current upper bound − decreased section of lower bound
THC --
else then
new WW = (current lower bound + current upper bound)/2
end if
BW updates the model based on the current window
Quantifying the current model $(\mathbf{M_{n-1}})$
Calculating MD$_{\mathbf{m-1}} = (\mathbf{M_{n-2}}, \mathbf{M_{n-1}})$
FW predicts the samples in DW → sub-result: $\mathbf{x_n}$
End while
Ensemble → majority voting among $(x_1, x_2, \ldots x_n)$ → ensemble results **X**

5.4. Workflow

Figure 7 shows an overview of the AA-HMM workflow. The variables MD, DT, TH, and THC work sequentially, as described in the previous sections, to adjust every WW to LOWW based on the underlying pattern. In addition, the BW is only executed two times for improving the efficiency.

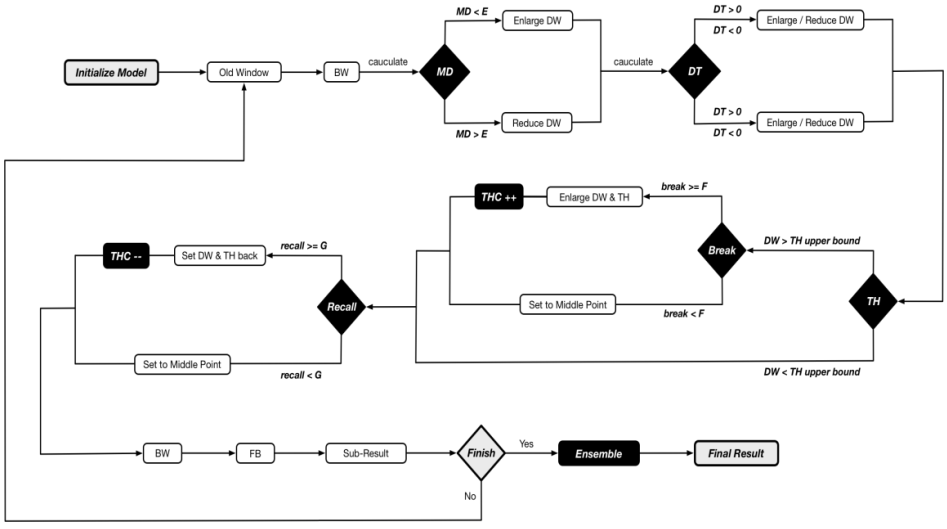

Figure 7. Workflow of AA-HMM.

6. Measurement Metrics

6.1. Precision and Recall

General accuracy may not reveal NIDS performance in real settings because the cost of misclassifying malicious samples is much higher than misclassifying benign samples. Therefore, the evaluation of NIDS should focus on specific precision and recall for benign and malicious samples, respectively. Although precision best reflects the model's accuracy based on its definition, recall can reveal the real losses caused by unidentified attacks in a practical setting. Other metrics, such as Receiver Operating Characteristic (ROC) curve, are inappropriate for evaluating the NIDS, which has been proven by a well-known review paper [33].

6.2. Efficiency Matrix

Cost matrix is a good metric for evaluating the NIDS because it can directly reflect the security (protection) level established by the models deployed in an operational environment. However, the common values, such as true positive, true negative, false positive, and false negative, have not been published and cannot be calculated given the published metrics. Therefore, referring to the definition of cost matrix, an alternative metric called Efficiency Matrix (EM) is defined as below, which evaluates the security level by calculating the precisions and recalls of benign and malicious samples, separately.

In Table 2, the matrices E_{11}, E_{12}, E_{21}, and E_{22} are the efficiency values of Precision (B), Recall (B), Precision (M), and Recall(M), respectively. So, the efficiency of the evaluated model should be calculated by:

$$\text{Efficiency(Model)} = \text{Precision(B)} \times E_{11} + \text{Recall(B)} \times E_{12} + \text{Precision(M)} \times E_{21} + \text{Recall(M)} \times E_{22} \quad (5)$$

The EM should be interpreted in a different manner from the cost matrix: the higher the efficiency, the better the performance. The pre-defined value E_{xy} varies between scenarios (applications). For evaluating NIDS, since misclassifying malicious samples is more costly than benign samples and the recall is a better metric than the precision in terms of reflecting the protection level provided by the model, the four efficiencies should maintain the following relationship:

$$E_{22} > E_{21} > E_{12} > E_{11} \quad (6)$$

Table 2. Definition of Efficiency Matrix (EM).

Efficiency Matrix	Precision	Recall
Benign	E_{11}	E_{12}
Malicious	E_{21}	E_{22}

7. Experiments

7.1. Goal and Strategy

Overall, the goal of the following two experiments was to verify the effectiveness of the designed feedback mechanism/variables, so the two AA-HMM models being tested adopted the default settings and the initial matrices (transition and emission) were set with balanced parameters (Section 7.2.1.) for better demonstrations. Therefore, considerable amounts of work needed to be completed to further improve the performance of the specific data set.

In the first experiment, we employed NSL-KDD [34] as the benchmark data set. Given the most recent evaluation of the NIDS data sets in [28], NSL-KDD is one of the two most frequently used data sets for evaluating the NIDS and has no distinct disadvantage compared with the newest data sets. Since numerous researchers have trusted the effectiveness and quality of and tested their algorithms on the NSL-KDD, it is the best data set for a comprehensive performance comparison.

In order to conduct a rigorous comparison, we selected the algorithms/papers for comparison based on the following criteria: (1) the cited paper has to clearly state that it used the two-label version of NSL-KDD instead of the five-label version; (2) the cited paper had to clearly state that it used Test+ (includes all the difficult samples that do not contained in the other test set, test-21) as the test set, because the NSL-KDD is valuable at providing researchers a great test set Test+ with distinct patterns from the training sets, which better reveal the performance difference between the different models; (3) the cited paper had to use Test+ as a separate test set, so that we could ensure that the difficult patterns/samples would not leak to the model during training phase through other evaluation approaches such as cross-validation; and (4) the cited paper had to publish enough metrics for a comprehensive comparison. Therefore, we selected three deep-learning algorithms: (1) deep neural network (DNN) [35], (2) soft-max regression (SMR), and (3) self-taught learning (STL) [36] as the comparison algorithms based on the aforementioned four criteria, which were evaluated on NSL-KDD in terms of accuracy, precision, recall, and efficiency.

In the second experiment, we verified the effectiveness of the designed mechanisms on a new data set (patterns), CTU-13 [37]. Specifically, in order to evaluate the AA-HMM's abilities to adapt and act as an anti-adversary on the current and intensively changed traffic patterns, we employed the no. 10 data set of CTU-13, which is composed of violently fluctuating traffic patterns caused by an intensive distributed denial-of-service (DDoS) attack.

7.2. Evaluation Methodology

7.2.1. Balanced Initial Model

The enhanced versions of HMMs published in other papers [16–23] usually use some sort of prior knowledge to initialize the transition and emission matrices, which enables the adopted parameters' distribution comply with the real data distribution to be evaluated. As a result, we could not distinguish if the improvement in performance was due to the skewed initial model or the enhancement mechanisms created by us. Accordingly, to eliminate biasing factors, the AA-HMM was initialized as a Balance Model (BM)—the probabilities in the transition and emission matrices were evenly distributed, as shown in Tables 3 and 4. Since the initial model was not biased to either one of the two classes, any accuracy improvement would then be attributable to the actual mechanism of the algorithm.

Table 3. Initial transition matrix.

Transition	Benign	Malicious
Benign	0.5	0.5
Malicious	0.5	0.5

Table 4. Initial emission matrix.

Emission	Observation 1	Observation 2	...	Observation *n*
Benign	0.5	0.5	...	0.5
Malicious	0.5	0.5	...	0.5

7.2.2. Preprocessing: Compact Matrices (CM)

As the method performance is determined by the transition and emission matrices, every row would become very long if there were too many items (hidden states or observations) in the two matrices. In practice, to maintain a valid model during evaluation, each state or observation would occupy a portion of the total probability of one, even if it is not present in the current window, which in turn complicates distinguishing the true hidden state harder during the FB procedure. Therefore,

creating and maintaining a Compact Matrix (CM) would contribute to the model generating higher accurate results, as indicated in [23].

To verify this conclusion, we performed an extensive experiment on a variety of data sets, as shown in Figure 8. One of the experiments involved discretizing a feature into different bins (from 3 to 30 and step of 1) as the observation sequences. Then, we ran the same AA-HMM (WW = 125) on each data set. We concluded that the more bins (possible values) of a feature, the lower the accuracy. Therefore, we verified that grouping the possible values into a small number of bins (CM) would be one of the best pre-processing approaches for HMM-based algorithms, as similar results were obtained using other data sets, because it not only improves the accuracy, but also thwarts the OE (Section 4 Methodology). Consequently, due to the monotone decreasing characteristic of the accuracy trajectory (Figure 8), we adopted a conservative pre-processing strategy that discretizes all the numeric features into only three to five bins for all the following experiments, even if larger bins (e.g., 6, 7, 8, 9, 10, etc.) are also appropriate (i.e., perform as good as 3–5 bins).

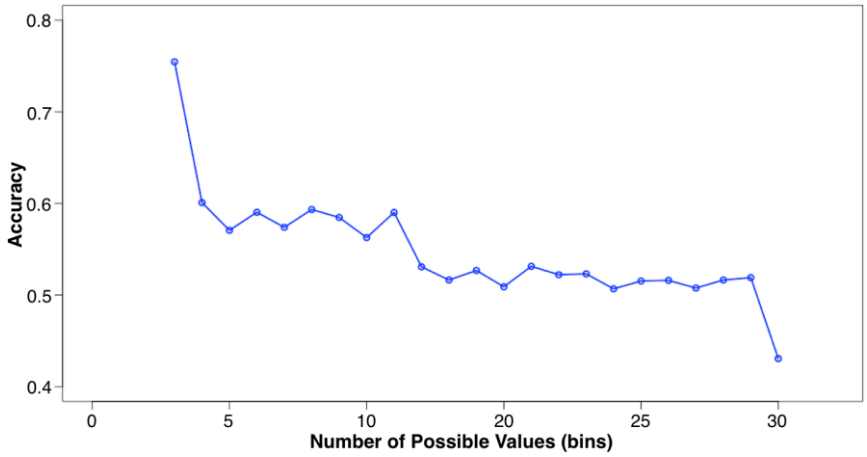

Figure 8. Correlation between accuracy and number of possible values.

7.2.3. Ensemble: Assigning Different Weights

Although a more accurate result can be obtained by only ensemble the models built upon the top N features, the overall performance might be lower or not stable in the future because the top N features would be replaced by others when the pattern changes. Therefore, to build a robust model, we improved the accuracy by assigning additional two to four weights to the top N models.

7.3. Experiment about Accuracy and Adaptivity: NSL-KDD

7.3.1. Introduction to NSL-KDD

NSL-KDD [34] is an optimized version of the well-known data set KDD. It solves some of the inherent problems of the original data set and has been frequently cited by researchers. The amount of records in NSL-KDD training and testing sets are reasonable, enabling the affordable completion experiments on the complete set without the need to randomly select a small portion. Consequently, the evaluation results of different research work would be consistent and comparable. There are 41 features in NSL-KDD data sets and the samples are labeled into two classes: benign and malicious. Furthermore, the attacks can be classified into four categories: (1) denial of service, (2) probing, (3) user to root, and (4) remote to local. A detailed introduction can be found in [38].

7.3.2. Evaluation on NSL-KDD

Data Pre-Processing

We eliminated 13 features before running the AA-HMM because all the samples were concentrated on a single possible value of those features either before or after discretization, which should be treated uninformative features. As a result, 28 of 41 features were adopted by the AA-HMM and only one sub-was is assigned a greater weight during the ensemble procedure.

Performance: Precision and Recall

Tables 5–8 depict the performance of DNN, SMR, STL, and AA-HMM, respectively. Four points can be gained from the results: (1) AA-HMM outperformed the DNN on all metrics. (2) SMR is not a balanced model as its performance concentrates more on metrics precision (B) and recall (M). Also, since the SMR's precision (M) is very low, it achieves high recall (M) by predicting as many malicious records as possible. Deploying a SMR model in the real setting would delay the service (e.g., web service) response time, as it would block too many legitimate packets due to low recall (B), which causes the blocked data to be re-transmitted to the end users. A balanced model should control the differences in the four metrics within 10%, like AA-HMM (only 2.225%). (3) Although STL is better than the DNN and SMR, its precision (B) and recall (M) were much lower than the AA-HMM. Also, the metric recall (M) shows that the STL is not a reliable NIDS because it would miss too many attacks due to the low recall (M). (4) AA-HMM is a balanced and most accurate model, and would provide the highest security level to potential victims in the real settings.

Table 5. Performance of deep neural network (DNN).

DNN	Precision	Recall	Accuracy
Benign	83.00%	75.00%	74.67%
Malicious	65.80%	74.20%	

Table 6. Performance of soft-max regression (SMR).

SMR	Precision	Recall	Accuracy
Benign	96.56%	63.73%	78.06%
Malicious	66.93%	97.00%	

Table 7. Performance of self-taught learning (STL).

STL	Precision	Recall	Accuracy
Benign	85.44%	95.95%	88.39%
Malicious	93.62%	78.41%	

Table 8. Performance of AA-HMM.

AA-HMM	Precision	Recall	Accuracy
Benign	93.37%	95.31%	93.48%
Malicious	93.63%	91.06%	

Performance: Efficiency Matrix

Given the setting principle of efficiency matrix (see Equation (6)), the efficiency matrix for this experiment is defined in Table 9.

Table 9. Efficiency Matrix.

Efficiency Matrix	Precision	Recall
Benign	10	15
Malicious	100	150

As the calculated efficiencies result from Equations (7)–(10), AA-HMM also outperformed DNN, SMR, and STL in terms of efficiency (security/protection level provided to users in practice).

$$\begin{aligned}\textbf{Efficiency (DNN)} &= 0.8300 \times 10 + 0.7500 \times 15 + 0.6580 \times 100 + 0.7420 \times 150 \\ &= 8.3 + 11.25 + 65.8 + 111.3 = \textbf{196.6500}\end{aligned} \tag{7}$$

$$\begin{aligned}\textbf{Efficiency (SMR)} &= 0.9656 \times 10 + 0.6373 \times 15 + 0.6693 \times 100 + 0.9700 \times 150 \\ &= 9.656 + 8.9373 + 66.93 + 145.5 = \textbf{231.0233}\end{aligned} \tag{8}$$

$$\begin{aligned}\textbf{Efficiency (STL)} &= 0.8544 \times 10 + 0.9595 \times 15 + 0.9362 \times 100 + 0.7841 \times 150 \\ &= 8.544 + 14.3925 + 93.6 + 117.615 = \textbf{234.1715}\end{aligned} \tag{9}$$

$$\begin{aligned}\textbf{Efficiency (AA-HMM)} &= 0.9337 \times 10 + 0.9531 \times 15 + 0.9363 \times 100 + 0.9106 \times 150 \\ &= 9.337 + 14.2965 + 93.63 + 136.59 = \textbf{253.8535}\end{aligned} \tag{10}$$

Verifications: MD and DT

After improving the performance, it was necessary to verify if the designed variables worked as expected and the performance improvement resulted from these variables. Taking one of the sub-models as an example (all the sub-models shared similar curves), its accuracy trajectory is shown in Figure 9. In the first five windows, the accuracies are very low because the model is adapting to the pattern from its initial state (BM). After fitting, the pattern starts to fluctuate, which lowers the accuracy again. However, the model rapidly fits the fluctuated pattern and maintains the accuracy at a high level until processing all the samples.

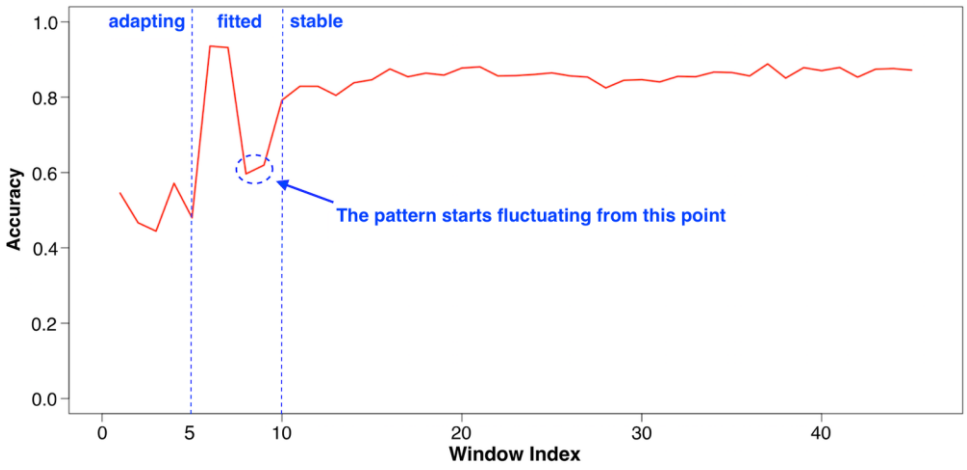

Figure 9. Trajectory of the accuracy.

Figure 10 shows the trajectories of MD and DT during evaluation; their values are very high in the early windows. Since the MD and DT act as the indicators of the variation and trend of the pattern in the current window, respectively, the two trajectories indicate that the accuracy would be low (high) when the |MD| and |DT| are high (low). This is consistent with the accuracy trajectory in Figure 9: the accuracy increases as the |MD| and |DT| gradually approach zero, which indicates that the model successfully fitted the dynamic patterns and reached a local optimal state. In conclusion, MD and DT are sound metrics and effective adaptive mechanisms for AA-HMM.

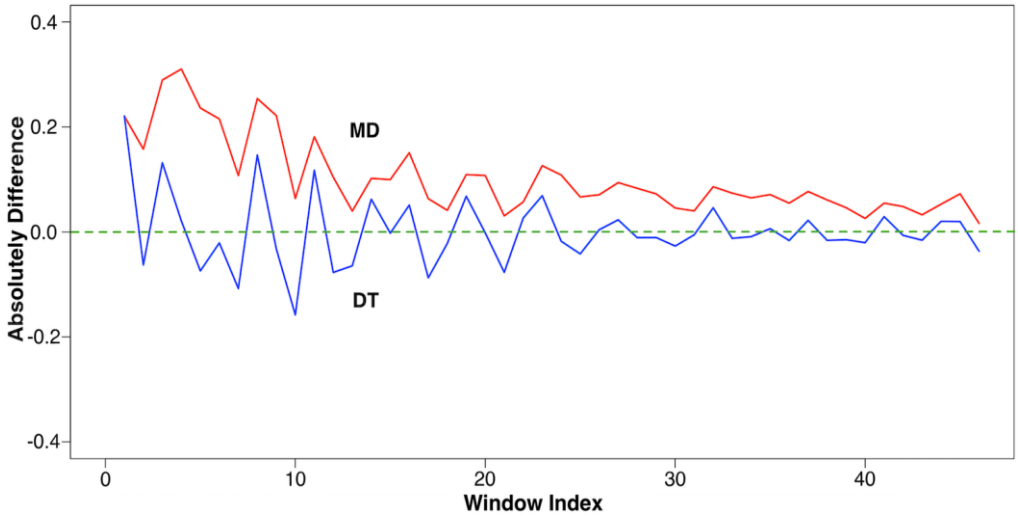

Figure 10. Trajectories of the MD and DT.

Verification: DW and TH

Figure 11 shows the trajectory of DW during evaluation, which gradually increased from the initial width of 10 and finally stabilized between 505 and 910. Referring to the accuracy trajectory in Figure 9, the stage of DW stabilization overlapped the stable stage of the accuracy, which also indicates that the model successfully fitted the dynamic pattern and reached a local optimal state. In addition, the reason that the DW always reset to 505 is that TH indicates the pattern fluctuation is temporary and not strong (Section 4 Methodology), so it was not necessary to enlarge the DW to a value higher than the current upper bound of TH under this circumstance. The accuracy improved and stabilized with the aid of the TH. In conclusion, both the DW and TH are effective mechanisms for improving performance.

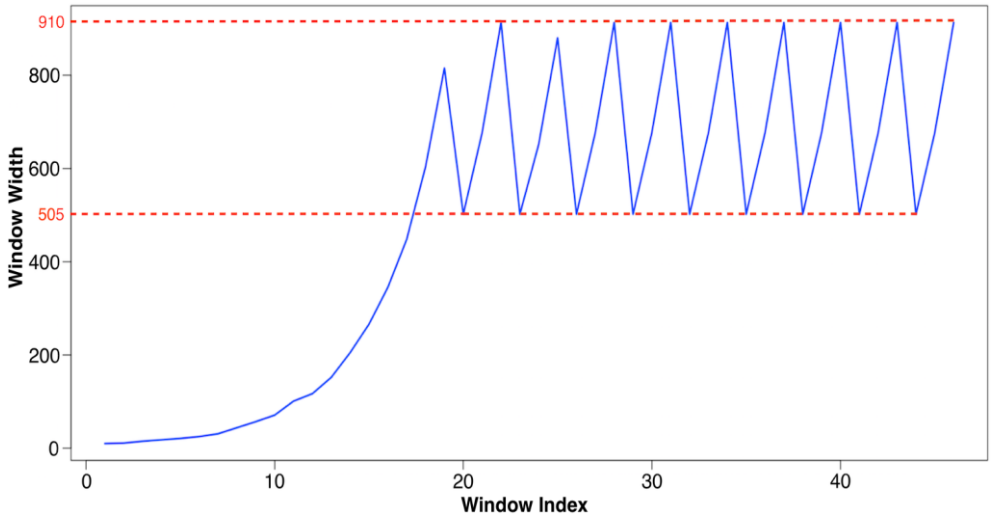

Figure 11. DW trajectory.

Evaluation: Adapting Rate

Since the model fit the pattern after the fifth window (Figure 9) and referring to the DW trajectory (Figure 11), the total number of samples in the first five windows (10, 11, 15, 18, and 21) was only 75. Compared with the total number of records (22,544) in the entire data set, the adaptive rate of AA-HMM was very high.

7.4. Anti-Adversarial Experiment: CTU-13

7.4.1. Introduction to CTU-13

CTU-13 is a set of botnet traffic, which captures a large amount of real botnet traffic mixed with normal traffic and background traffic. The CTU-13 data set consists of 13 scenarios of different botnet samples. In each scenario, the creators execute a specific malware that uses several protocols and performs different actions. A detailed introduction can be found in [37].

7.4.2. Evaluation on CTU-13

Data Pre-Processing

The User Datagram Protocol (UDP) DDoS data set (no. 10) of CTU-13 was used as the evaluation set and the samples were labeled as benign or malicious. To test the AA-HMM's performance on intensive malicious attacks, we extracted a subset with an intensive pattern change (from no. 440,000 to no. 520,000–80,001 records in total). The benign and malicious samples alternatively dominated the first and second half of the subset, respectively. This subset contains intensive DDoS attacks that are suddenly launched.

Anti-Adversary and Adaptivity Performance

As Figure 12 shows, the thicker blue line represents the percentage of benign samples within each window. An intensive malicious attack (DDoS) occurs in this data set. However, the model maintained very high precision toward the benign samples (the thinner red line) when being attacked by an intensive DDoS attack.

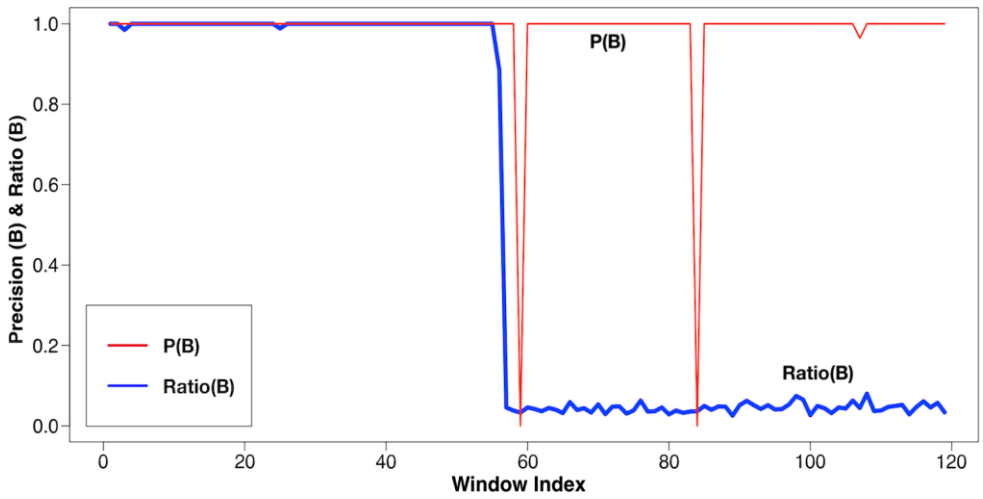

Figure 12. Correlation between precision (B) and intensive attacks demonstrated in the drastically fluctuating pattern.

Performance: Imbalance Classes

In Figure 13, the thinner blue line represents the percentage of malicious instances in each window and the thicker red line is the recall of malicious samples. Although the attacks occur only twice (the magnified two dents in the plot), in the first half of the data set, the recall of malicious samples within the two dents is 100%. AA-HMM is capable of identifying trivial malicious samples (one type of sample), which were overwhelmed by the benign samples (the other sample types). In conclusion, AA-HMM has a strong ability to resolve the imbalanced classes issue.

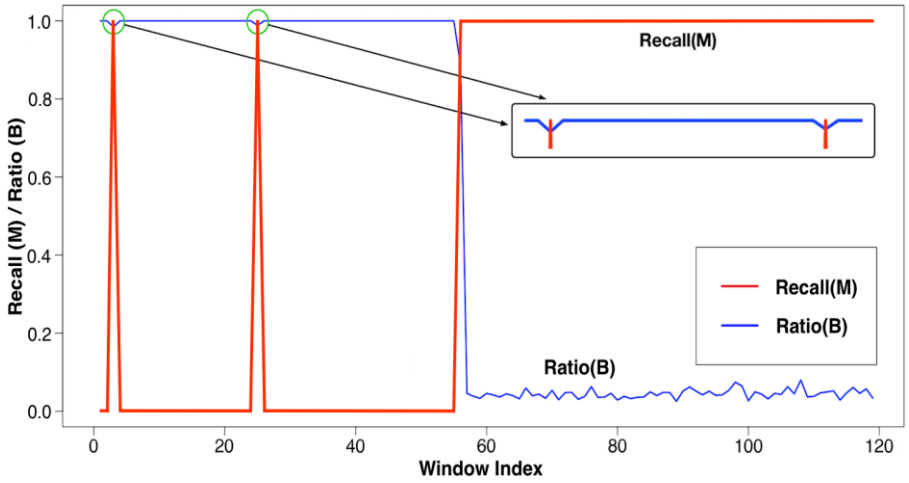

Figure 13. Correlation between recall toward malicious samples and ratio of benign samples.

Attack Visualization

Since different attacks have different patterns and result in varied WWs, the attacks can be visualized by the DW trajectory. As shown in Figure 14, the first half of the DW trajectory is significantly different from the second half, which is compliant with the actual scenario (the DDoS attacks were launched and dominated the traffic after the middle point). Therefore, AA-HMM can be applied as a novel attack visualization tool to detect if the NIDS is being attacked or to even identify the attack types via the DW trajectory.

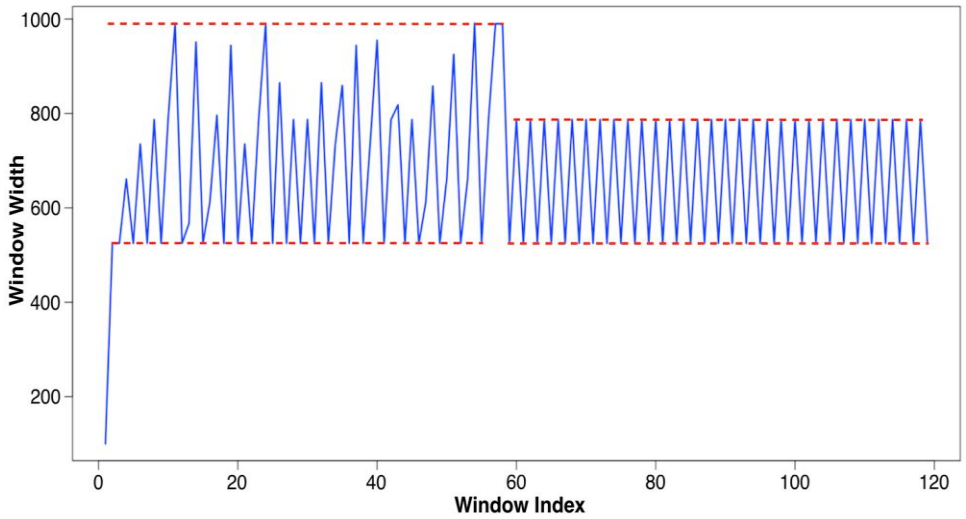

Figure 14. DW trajectory.

Re-Verification: MD and DT

From the perspective of attack distribution, there are two challenges in forming predications on this data set. As shown in Figure 15, (1) spur 1 adapts the model to the first half of traffic from the BM and (2) spur 2 adapts the model to the second half of the traffic (DDoS attacks) from the first half (normal traffic). Based on the trajectories of MD and DT, we concluded that both challenges and adaptation processes were resolved and completed rapidly because the MD and DT were reduced to a very low level within only three windows, which shows that the AA-HMM is responsive to intensive attacks.

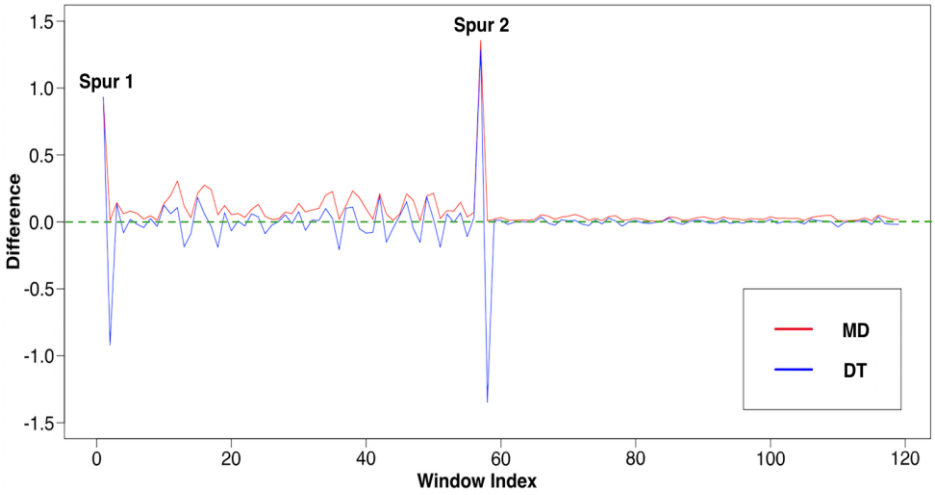

Figure 15. Trajectories of MD and DT.

7.5. Time Cost Experiment

An AA-HMM model running in RStudio (RStudio, Redmond, United States) on a regular laptop can accomplish both the self-training and the evaluation processes on 10,000 samples within 2.5 s. When compared with the three deep-learning algorithms in the previous experiment, which usually require tens of seconds on the same amount of data, AA-HMM outperformed these algorithms in execution cost and accuracy, rendering the AA-HMM an extremely viable solution as a NIDS.

8. Conclusions

Building an anti-adversarial model is one of the most innovative research topics in the anomaly-based network intrusion detection field, including the software-defined network-based NIDS model [39]. As a successful anti-adversarial prototype, AA-HMM quantifies the model difference (MD) and difference trend (DT) between adjacent windows as indicators of accuracy and pattern fluctuation, which transforms a regular HMM into an online algorithm with strong adaptability. The threshold mechanism (TH and THC) is the core anti-adversarial technique adopted by AA-HMM, which further enhances the adaptability and stability of the model. Particularly, the online architecture of DW, which predicts samples section-by-section, used in AA-HMM may wrap other quantified algorithms to largely improve the performance of the base models. In addition, AA-HMM could be used as a novel visualization tool to indicate if the NIDS is being attacked and to even distinguish the attack type based on the DW trajectory.

Author Contributions: Conceptualization, C.S.; methodology, C.S.; software, C.S.; validation, C.S., A.P. and K.Y.; writing—original draft preparation, C.S.; writing—review and editing, A.P., K.Y. and C.S.; supervision, A.P. and K.Y.; funding acquisition, A.P.

Funding: This research was funded by the Florida Center for Cybersecurity, 2016-1017 FC2 Collaborative Seed Award Program.

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

Abbreviations

The following abbreviations are used in this manuscript:

AA-HMM	Anti-Adversarial Hidden Markov Model
B	benign (samples)
BM	Balance Model
BW	Baum-Welch
CM	Compact Matrix

DDoS	Distributed Denial-Of-Service
DNN	Deep neural network
DT	Difference Trend
DW	Dynamic Window
EM	Efficiency Matrix
FB	Forward-Backward
HMM	Hidden Markov Model
LDA	Linear Discriminant Analysis
LOWW	Local Optimal Window Width
M	Malicious (samples)
MD	Model Difference
ML	Machine Learning
NIDS	Network Intrusion Detection System
OE	Optimal Evasion
PCA	Principal Component Analysis
PE	Pattern Entropy
PERD	PE ReDuction
PV	Pattern Variation
ROC curve	Receiver Operating Characteristic curve
SMR	Soft-Max Regression
STL	Self-Taught Learning
TH	Threshold
THC	Threshold Controller
TTC	Total Time Cost
WW	Window Width

References

1. Sommer, R.; Paxson, V. Outside the Closed World—On Using Machine Learning for Network Intrusion Detection. In Proceedings of the IEEE Symposium on Security and Privacy, Berkeley/Oakland, CA, USA, 16–19 May 2010.
2. Linden, G.; Smith, B.; York, J. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Comput.* **2003**, *7*, 76–80. [CrossRef]
3. Gomez-Uribe, C.A.; Hunt, N. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Trans. Manag. Inf. Syst.* **2016**, *6*, 13. [CrossRef]
4. Khan, N.H.; Adnan, A. Urdu Optical Character Recognition Systems: Present Contributions and Future Directions. *IEEE Access* **2018**, *6*, 46019–46046. [CrossRef]
5. Chen, K.; Zhao, T.; Yang, M.; Liu, L.; Tamura, A.; Wang, R.; Utiyama, M.; Sumita, E. A Neural Approach to Source Dependence Based Context Model for Statistical Machine Translation. *IEEE Access* **2018**, *6*, 266–280. [CrossRef]
6. Hsia, J.H.; Chen, M.S. Language-model-based detection cascade for efficient classification of image-based spam e-mail. In Proceedings of the 2009 IEEE international conference on Multimedia and Expo ICME'09, New York, NY, USA, 28 June–3 July 2009; pp. 1182–1185.
7. Zhang, F.; Chan, P.P.; Biggio, B.; Yeung, D.S.; Roli, F. Adversarial Feature Selection Against Evasion Attacks. *IEEE Trans. Cybern.* **2016**, *46*, 766–777. [CrossRef] [PubMed]
8. Polychronakis, M.; Anagnostakis, K.G.; Markatos, E.P. Real-world Polymorphic Attack Detection using Network-level Emulation. In Proceedings of the 4th Annual Workshop on Cyber Security and Information Intelligence Research: Developing Strategies to Meet the Cyber Security and Information Intelligence Challenges Ahead, Oak Ridge, TN, USA, 12–14 May 2008.
9. Kaur, H.; Singh, G.; Minhas, J. A Review of Machine Learning based Anomaly Detection Techniques. *Int. J. Comput. Appl. Technol. Res.* **2013**, *2*, 185–187. [CrossRef]
10. Bhuyan, M.H.; Bhattacharyya, D.K.; Kalita, J.K. Network Anomaly Detection: Methods, Systems and Tools. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 303–336. [CrossRef]
11. Kuncheva, L.I. *Combining Pattern Classifiers: Methods and Algorithms*; Wiley: Hoboken, NJ, USA, 2004.

12. Kuncheva, L.I. Diversity in multiple classifier systems. *Inf. Fusion* **2005**, *6*, 3–4. [CrossRef]
13. Weng, F.; Jiang, Q.; Shi, L.; Wu, N. An Intrusion Detection System Based on the Clustering Ensemble. In Proceedings of the International Workshop on Anti-Counterfeiting, Security and Identification (ASID), Xiamen, China, 16–18 April 2007; pp. 121–124.
14. Hodo, E.; Bellekens, X.; Hamilton, A.; Tachtatzis, C.; Atkinson, R. Shallow and Deep Networks Intrusion Detection System: A Taxonomy and Survey. *arXiv* **2017**, arXiv:1701.02145.
15. Shankar, V.; Chang, S. Performance of Caffe on QCT Deep Learning Reference Architecture—A Preliminary Case Study. In Proceedings of the IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), New York, NY, USA, 26–28 June 2017; pp. 35–39.
16. Khreich, W.; Granger, E.; Sabourin, R.; Miri, A. Combining Hidden Markov Models for Improved Anomaly Detection. In Proceedings of the IEEE International Conference on Communications, Dresden, Germany, 14–18 June 2009; pp. 1–6.
17. Hu, J.; Yu, X.; Qiu, D.; Chen, H.H. A simple and efficient hidden Markov model scheme for host-based anomaly intrusion detection. *IEEE Netw.* **2009**, *23*, 42–47. [CrossRef]
18. Hurley, T.; Perdomo, J.E.; Perez-Pons, A. HMM-Based Intrusion Detection System for Software Defined Networking. In Proceedings of the 15th IEEE International Conference on Machine Learning and Applications (ICMLA), Anaheim, CA, USA, 18–20 December 2016; pp. 617–621.
19. Jain, R.; Abouzakhar, N.S. Hidden Markov Model based anomaly intrusion detection. In Proceedings of the International Conference for Internet Technology and Secured Transactions, London, UK, 10–12 December 2012; pp. 528–533.
20. Song, X.; Chen, G.; Li, X. A Weak Hidden Markov Model based intrusion detection method for wireless sensor networks. In Proceedings of the International Conference on Intelligent Computing and Integrated Systems, Guilin, China, 22–24 October 2010; pp. 887–889.
21. Ren, H.; Ye, Z.; Li, Z. Anomaly detection based on a dynamic Markov model. *Inf. Sci.* **2017**, *411*, 52–65. [CrossRef]
22. Ahmadian, R.A.; Rasoolzadegan, A.; Javan, J.A. A systematic review on intrusion detection based on the Hidden Markov Model. *Stat. Anal. Data Min. ASA Data Sci. J.* **2018**, *11*, 111–134. [CrossRef]
23. Ariu, D.; Tronci, R.; Giacinto, G. HMMPayl: An intrusion detection system based on Hidden Markov Model. *Comput. Secur.* **2011**, *30*, 221–241. [CrossRef]
24. Russell, S.J.; Norvig, P. *Artificial Intelligence: A Modern Approach*, 3rd ed.; Pearson: Kuala Lumpur, Malaysia, 2009.
25. Tan, P.N.; Steinbach, M.; Kumar, V. *Introduction to Data Mining*; Pearson: London, UK, 2006.
26. Rabiner, L.R. A Tutorial on Hidden Markov Model and Selected Applications in Speech Recognition. *Proc. IEEE* **1989**, *77*, 257–286. [CrossRef]
27. Zhao, F.; Zhao, J.; Niu, X.; Luo, S.; Xin, Y. A Filter Feature Selection Algorithm Based on Mutual Information for Intrusion Detection. *Appl. Sci.* **2018**, *8*, 1535. [CrossRef]
28. Hindy, H.; Brosset, D.; Bayne, E.; Seeam, A.; Tachtatzis, C.; Atkinson, R.; Bellekens, X. A Taxonomy and Survey of Intrusion Detection System Design Techniques, Network Threats and Datasets. *arXiv* **2018**, arXiv:1806.03517.
29. The R Project for Statistical Computing. Available online: https://cran.r-project.org/web/packages/HMM/HMM.pdf (accessed on 11 October 2018).
30. Lowd, D.; Meek, C. Adversarial learning. In Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, Chicago, IL, USA, 21–24 August 200; pp. 641–647.
31. Nelson, B.; Rubinstein, B.I.; Huang, L.; Joseph, A.D.; Lee, S.J.; Rao, S.; Tygar, J.D. Query strategies for evading convex-inducing classifiers. *J. Mach. Learn. Res.* **2012**, *13*, 1293–1332.
32. Churbanov, A.; Winters-Hilt, S. Implementing EM and Viterbi algorithms for Hidden Markov Model in linear memory. *BMC Bioinform.* **2008**, *9*, 224. [CrossRef] [PubMed]
33. McHugh, J. Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. *ACM Trans. Inf. Syst. Secur.* **2000**, *3*, 262–294. [CrossRef]
34. Canadian Institute for Cybersecurity. Available online: http://www.unb.ca/cic/datasets/nsl.html (accessed on 11 October 2018).

35. Tang, T.A.; Mhamdi, L.; McLernon, D.; Zaidi, S.A.R.; Ghogho, M. Deep Learning Approach for Network Intrusion Detection in Software Defined Networking. In Proceedings of the International Conference on Wireless Networks and Mobile Communications (WINCOM), Fez, Morocco, 26–29 October 2016.
36. Niyaz, Q.; Sun, W.; Javaid, A.Y.; Alam, M. A Deep Learning Approach for Network Intrusion Detection System. In Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies, BICT'15, New York, NY, USA, 3–5 December 2015; pp. 21–26.
37. Garcia, S.; Grill, M.; Stiborek, J.; Zunino, A. An empirical comparison of botnet detection methods. *Comput. Secur. J.* **2014**, *45*, 100–123. [CrossRef]
38. Dhanabal, L.; Shantharajah, S.P. A Study on NSL-KDD Data set for Intrusion Detection System Based on Classification Algorithms. *Int. J. Adv. Res. Comput. Commun. Eng.* **2015**, *4*, 446–452.
39. Song, C.; Perez-Pons, A.; Yen, K.K. Building a Platform for Software-Defined Networking Cybersecurity Applications. In Proceedings of the 15th IEEE International Conference on Machine Learning and Applications (ICMLA), Anaheim, CA, USA, 18–20 December 2016; pp. 482–487.