INCREMENTAL SUPPORT VECTOR MACHINE APPROACH FOR DOS AND DDOS ATTACK DETECTION

by

Seunghee Lee

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the Degree of

Master of Science

Department of Computer and Information Technology

West Lafayette, Indiana

May 2019

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. John A. Springer, Chair

Department of Computer and Information Technology

Dr. Eric T. Matson,

Department of Computer and Information Technology

Dr. Vetria L. Byrd,

Department of Computer and Graphics Technology

Approved by:

Dr. Eric T. Matson

Head of the Graduate Program

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CSV	Candidate Support Vector
DoS	Denial of Service
DDoS	Distributed Denial of Service
HTTP	HyperText Transfer Protocol
IDS	Intrusion Detection System
IoT	Internet-of-Things
ISVM	Incremental Support Vector Machine
KNN	K-Nearest Neighbors
SVM	Support Vector Machine
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator

GLOSSARY

PCAP formatted file – A file format for capturing packets of network data. A PCAP formatted file is a data file that contains the packet data of a network. This file is mainly used to analyze network characteristics.

Support Vector Machine – A supervised machine learning algorithm that performs classification by finding the hyperplane that maximizes the margin between the classes.

Support Vectors – Subset of training samples that define the hyperplane in Support Vector Machine (SVM) algorithm.

ABSTRACT

Author: Lee, Seunghee. M.S.
Institution: Purdue University
Degree Received: May 2019
Title: Incremental Support Vector Machine Approach for DoS and DDoS Attack detection
Major Professor: John A. Springer

Support Vector Machines (SVMs) have generally been effective in detecting instances of network intrusion. However, from a practical point of view, a standard SVM is not able to handle large-scale data efficiently due to the computation complexity of the algorithm and extensive memory requirements. To cope with the limitation, this study presents an incremental SVM method combined with a k-nearest neighbors (KNN) based candidate support vectors (CSV) selection strategy in order to speed up training and test process. The proposed incremental SVM method constructs or updates the pattern classes by incrementally incorporating new signatures without having to load and access the entire previous dataset in order to cope with evolving DoS and DDoS attacks. Performance of the proposed method is evaluated with experiments and compared with the standard SVM method and the simple incremental SVM method in terms of precision, recall, F1-score, and training and test duration.

CHAPTER 1. INTRODUCTION

This thesis focuses on expanding knowledge in the field of incremental support vector machines for DoS and DDoS attack detection. This chapter provides an overview of the research study by presenting a background of the problem area and a research question. In addition, it covers the research significance, assumptions, limitations, and delimitations that define the extent of the study.

1.1 Background

Incremental and online learning refers to the machine learning methods in which sequentially collected data is continuously used to update the model, leading to a continually updated status. Even though many incremental and online algorithms have been proposed so far, they don't have a long history of usage and are pretty immature. However, in the era of big data, many incremental and online learning algorithms have begun to attract attention as there is an increasing number of problem domains where this concept is necessary to be applied to data streams or big data and thereby addressing issues in data availability and resource scarcity for large-scale tasks, respectively (Losing, Hammer, & Wersing, 2018).

In particular, the large data volume issue occurs in most areas and one derived problem is large-scale data training in the machine learning area. Network intrusion detection systems are also facing problems in this regard that result from evolving attack patterns and sizes of attacks. Especially, the magnitude of DoS and DDoS attacks has increased dramatically over time, and even attack patterns have become increasingly diverse, resulting in the accumulation of data samples with dozens of record attributes and making the size of training dataset continuously grows (Bhandari, Sangal, & Kumar, 2015; Bhuyan, Bhattacharyya, & Kalita, 2012). To accurately classify evolving network attacks, signature-based intrusion detection systems (IDSs) typically need to update models because they have limitations for unknown and new attack type. However, traditional non-incremental techniques that generate models by learning over the entire

training dataset at once are unable to handle large-scale data efficiently because they are costly in terms of memory and computing consumption. Sometimes even very large dataset cannot be loaded into the main memory. On the other hand, incremental learning is able to quickly learn and update the model dynamically from new samples without having to load and access the entire previous dataset (Yi et al., 2011). In network intrusion detection systems, however, only a few studies used incremental approaches due to issues such as dynamic updates of normal and attack profiles and the catastrophic forgetting phenomenon that forgets previously acquired knowledge that should not occur in network intrusion detection systems (Bhuyan et al., 2012).

One of the machine learning algorithms that has been widely applied in the field of DoS and DDoS attack detection is Support Vector Machine (SVM). Even though its outstanding performance for classification problem, SVM is not able to handle large-scale data efficiently due to the computation complexity and extensive memory requirement of the SVM training (Liao & Couillet, 2017). In this regard, this study proposes a new incremental SVM approach of updating the model and thereby incrementally covering new attack patterns to cope with evolving attack scenarios, and this study identifies how efficient this technique is in various aspects compared to the non-incremental SVM approach.

1.2 Problem Statement

Support vector machines (SVMs) are a supervised classifier successfully applied in the field of DoS and DDoS attack detection. However, the training process of standard SVM suffers from the problems of their high time-consumption and large memory requirement when applied on a large number of training samples for the reason that obtaining a decision function involves the solution of a quadratic programming problem (QP). This QP problem can be solved in $O(t^3)$ with $O(t^2)$ memory using a standard QP solver (Nalepa & Kawulok, 2018; Zeng, Yu, Xu, Xie, & Gao, 2008). With the accumulation of new data samples in network traffic, this issue is becoming more challenging nowadays. For large-scale problems, one solution is to adapt the SVM

learning algorithm to learn incrementally, which is called the incremental SVM (ISVM). However, the simple ISVM method, which combines a new batch of data with the support vectors from the previous learning steps, experiences a forgetting phenomenon that results in loss of previously learned information. In this regard, this study presents the incremental SVM approach combined with a k-nearest neighbors (KNN) based candidate vector selection strategy, targeting both batch-incremental learning and class-incremental learning.

1.3 Research Question

Can the incremental SVM method combined with a k-nearest neighbors (KNN) based candidate support vector selection strategy achieve the same score as the non-incremental SVM algorithm while shortening the training and test duration?

1.4 Significance

This study is meant to verify whether the incremental SVM method combined with k-nearest neighbors (KNN) based candidate vector selection strategy can be used for DoS and DDoS attack detection by discovering its advantages and limitations. Specifically, this study conducts experiments in the incremental environment to handle (1) the batch-incremental problems and (2) the class-incremental problems, which reflect the learning ability of new types of intrusions that do not exist in the previous training datasets.

1.5 Assumptions

The assumption for this study include:

- This study assumes that data is explicitly labeled and does not focus on learning from unlabeled or partially labeled data.

1.6 Limitations

The limitation for this study include:

- This study is conducted in a stationary environment using an offline dataset only.

1.7 Delimitations

The delimitations for this study include:

- This study does not consider other types of attacks other than DoS and DDoS.
- This study does not deal with the problem of concept drift in terms of attack pattern changes.

1.8 Summary

This chapter provided an overall description of the research including research background, problem statement, significance, research question, limitations, and delimitations that are considered in this study.

CHAPTER 2. REVIEW OF LITERATURE

This chapter provides a review of the literature relevant to this study. This chapter is organized as follows. Section 2.1 describes evolving DoS and DDoS attacks. Section 2.2 introduces various incremental and online learning algorithms with their merits and disadvantages. At last, datasets for an intrusion detection system evaluation will be discussed in section 2.3.

2.1 Evolving DoS and DDoS attacks

A Denial of Service (DoS) attack is a malicious attempt to affect the availability of one or more target systems such as a website or applications. Typically, an attacker generates a huge volume of packets or requests, ultimately overwhelming the target system. In a Distributed Denial of Service (DDoS) attack, the attacker uses multiple compromised sources to create the coordinated attack against the target system, and the attack is compromised of various elements such as the attacker, handlers, and agents. In general, the compromised attack agents called botnets and are infected with malicious software such as a virus or malware and controlled by the attacker. These infected endpoint systems are typically computers and servers, but in recent years, the number of infected Internet-of-Things (IoT) and mobile devices is increasing due to the exploitation of the fact that they still have very basic security vulnerabilities. Attackers find vulnerable devices that can be infected with phishing attacks, malicious ad attacks, and other large-scale infecting techniques to build such a system (Mahjabin, Xiao, Sun, & Jiang, 2017; Mirkovic & Reiher, 2004). With the massive number of IoT and mobile devices forming botnets, the scale of DDoS attacks has also grown significantly. For example, Domain Name System (DNS) provider Dyn came under a large distributed denial-of-service (DDoS) attack against their managed DNS infrastructure on October 21st, 2016 (York, 2016). They estimated that the attack involved 100,000 IP addresses and confirmed that the primary source of the significant volume of DDoS attack was Mirai botnet consisting of a large number of unsecured Internet-of-Things (IoT) devices such as

printers, IP cameras, and routers infected with the Mirai malware (*Dyn Analysis Summary Of Friday October 21 Attack — Dyn Blog*, n.d.). With its base of huge botnets, the attack reached 1.2 terabits per second (Tbps) at its peak (Schneier, 2016). This case demonstrated that the increased size of DDoS attacks requires the capability to quickly identify and block the large-size attacks.

2.1.1 Various Types of DoS and DDoS attacks

In addition to the increasing size of the attacks, both DoS and DDoS attacks have become more sophisticated and professional, making it more difficult to detect. Attacks are carried out long term nowadays, and attackers mimic the legitimate traffic or normal flash crowd traffic, even avoiding anomaly-based detection (Akilandeswari & Shalinie, 2012; Kandula, Katabi, Jacob, & Berger, 2005; Peng, Leckie, & Ramamohanarao, 2007). This subsection introduces some types of DoS attacks and particularly the DDoS attack that are covered in this study.

DoS Hulk DoS Hulk is a denial of service attack that is used to attack web servers by generating a large number of HTTP requests in a very short time. This attack constantly changes the headers and parameters in the uniform resource locator (URL) of the target website, thus bypassing caching engines and making the server allocates as many resources as possible to prevent legitimate users from receiving service (Behal & Kumar, 2017; Grafov, 2018).

DoS GoldenEye DoS GoldenEye is a Python-based DoS attack using the HTTP Keep-Alive method paired with No-cached feature to persist socket connection until it exhausts all available resource pools on the target HTTP server. The Keep-Alive method maximizes the size of a file transmitted over a single TCP connection. The No-cached messaging feature also disables HTTP cache control (Behal & Kumar, 2017; Jseidl, 2018).

DoS Slowloris DoS Slowloris sends a large number of incomplete HTTP requests to open and keep many simultaneous HTTP connections open for a long period of time, thus filling the maximum connection pool and eventually all the connections are used. Unlike bandwidth-consuming attacks, this attack can be performed using a small amount of bandwidth, and hence they can be effectively executed on low-performance hosts. Therefore, a volumetric-based detection system is generally not effective in detecting the Slowloris attack because the volume of this attack normally does not reach the detection thresholds. One known solution to this attack is to set a timeout value to limit the maximum time a client can stay connected (Cambiaso, Papaleo, Chiola, & Aiello, 2013; Jseidl, 2018).

DoS SlowHTTPTest DoS SlowHTTPTest is a denial of service attack aiming to deplete the resources of the target server, such as a DoS Slowloris attack. This attack manipulates the window size of TCP equal to zero, making the receiving process as slow as possible. This attack looks like normal HTTP requests and is difficult to distinguish from normal traffic. However, this attack has the characteristic of a window size connection that is much smaller than the normal flow (Park, Iwai, Tanaka, & Kurokawa, 2014).

DDoS In a Distributed Denial of Service (DDoS) attack, the attacker uses multiple compromised sources to create the coordinated attack against the target system, and the attack is commonly compromised of four different elements as illustrated in Figure 2.1. In the botnets based DDoS attack model, the DDoS attack is carried out in several phases. First, the attacker finds a significant number of machines with security loopholes. This security vulnerability is then exploited by the attacker to infect the system with malicious software such as a virus or malware. This process creates multiple compromised machines called masters or handlers. They are further used to scan for a sufficient number of other vulnerable computers and machines such as IoT devices to be indirectly controlled by the attacker. These compromised attack agents, called botnets, communicate with the attacker via

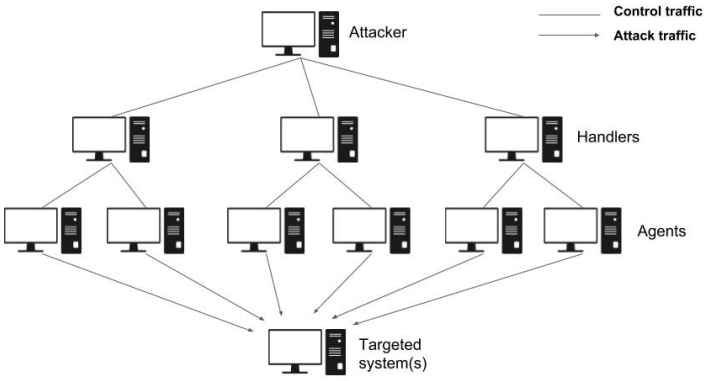


Figure 2.1. Botnets based DDoS Attack Model

handlers and are ready to attack. In the attack phase, many of these infected agents generate a large number of attack streams toward the intended victim, exhausting resources on the network and system and thus preventing access of legitimate users (Mahjabin et al., 2017; Mirkovic & Reiher, 2004).

In addition to the attacks described above, more and more attack patterns appear, and even some attacks involve different attack vectors simultaneously within an attack, making existing defense systems less effective.

2.1.2 Limitations of Existing Detection Systems for Evolving Attacks

In general, intrusion detection systems (IDSs) are classified into two types: signature-based and anomaly-based. The signature-based IDSs recognize known patterns or signatures such as malicious packet size or flow duration used by DoS and DDoS attacks. However, such systems are not capable of detecting novel or unknown attacks whose signatures are not stored. Therefore, in signature-based detection systems, new attacks are always a big concern unless the model is updated regularly. On the contrary, anomaly-based detection systems classify traffic as either normal or anomalous against the model built based on the normal profiles. Hence, anomaly-based IDSs have the ability to detect novel attacks because the detection schemes are based on deviations from what is

considered normal. However, the false positive rate, which is the ratio of instances that are incorrectly classified as an attack, is higher than signature-based methods. It is because a group of data with a different pattern from normal is detected as an attack incorrectly. In this regard, incremental methods can be useful to cope with the evolving network attacks for accurate classification in that they are able to construct or update the pattern classes by incrementally incorporating new signatures as they encounter them without having to learn from the beginning (Bhuyan et al., 2012; Kaur, Kumar, & Bhandari, 2017).

2.2 Incremental and Online Learning

Machine learning methods provide powerful technologies that enable to analyze data and infer structural information from given digital data by using computational methods to learn. Historically, most applications or systems that use machine learning methods operate on a batch setting where the training set is given prior training, assuming that the data and the structure are static (Gepperth & Hammer, 2016). However, there are a number of areas where classical machine learning methods are not appropriate. In this regard, Losing et al. stated the following:

> Classical batch machine learning does not continuously integrate new information into already constructed models but instead regularly reconstruct new models from scratch. This is not only very time consuming but also leads to potentially outdated models (Losing et al., 2018, p. 1)

To overcome this problem, incremental and online learning can be considered as a solution, because they keep the model all-time up-to-date by integrating new information into the model as soon as it is available and are efficient from processing time and resource perspectives (Losing et al., 2018). Given this capability, incremental online algorithms are well-suited for learning that enables devices to adapt to individual habits. An example can be the smart home products, online recommendation system, and autonomous robotics. As such, incremental and online learning has usually been a desirable choice in interactive

scenarios where data arrives as a stream with possible concept drift. That is, incremental online learning has advantages especially when data is (1) very large or (2) non-stationary. However, many studies use the definition of incremental and online learning ambiguously and interchangeably. In this regard, Losing et al. defined them as follow:

> We define an incremental learning algorithm as one that generates on a given stream of training data s_1, s_2, ..., s_t a sequence of models h_1, h_2, h_3,..., h_t. We specify online learning algorithms as incremental learning algorithms which are additionally bounded in model complexity and run-time, capable of endless/lifelong learning on a device with restricted resources (Losing et al., 2018, p. 2)

In addition, not all algorithms have a corresponding efficient online version, and it is still unclear which algorithm is appropriate to use for a particular task and how they behave in comparison to each other (Losing et al., 2018). Besides, one main challenge is to determine what and how information acquired from the previous learning step should be retained for the next training step while coping with increasing data samples (Chitrakar & Huang, 2014). Various incremental and online learning algorithms have been suggested in the literature so far with strengths and weakness. Some popular incremental and online methods are described in the following.

Incremental Support Vector Machine (ISVM). The incremental version of SVM was introduced by Syed et al. to handle the scalability issue of SVM. Based on the fact that the resulting decision function on SVM is determined only by the support vectors, they proposed a way to train an SVM incrementally by combining a new batch of data with the support vectors from the previous learning steps. Even though this approach is efficient in terms of storage cost, memory usage, and training time, it gives only approximate results unless all previously seen data is contained in the set of candidate vectors (Losing et al., 2018; Ruping, 2001; Syed et al., 1999). A recent application using ISVM can be the online human recognition from video surveillance using texture and color features (Y. Lu, Boukharouba, Boonært, Fleury, & Lecoeuche, 2014).

Online Random Forest (ORF). Online Random Forest is an online version of the Random Forest algorithm introduced by Saffari, Leistner, Santner, Godec, and Bischof. In order to handle a real-time environment where data arrives sequentially with possible changes in data distribution, this algorithm continuously grows new trees based on online bagging and extremely randomized forests and drops trees that predict poorly, increasing the adaptivity for a dynamic environment. The process of incremental learning and dropping is constantly occurring so continuously adapting to a changing environment (Saffari et al., 2009). Furthermore, they are insensitive to feature scaling and can be easily applied in practice (Losing et al., 2018). An example of incremental online random forest can be the visual object tracking (Saffari et al., 2009; A. Wang, Wan, Cheng, & Li, 2009).

Learn++. Learn++ algorithm, introduced by Polikar, Upda, Upda, and Honavar, is an incremental version of neural network classifier. For each chunk of an incoming sample, Learn++ trains base classifiers and generates a resulting ensemble of classifiers that are combined through the weighted majority voting procedure. Learn++ algorithm is capable of learning new information, and it does not forget previously acquired knowledge (Polikar et al., 2001).

Naive Bayes (NB_{Gauss}). Naive Bayes classifier supports incremental learning. The online version of Naive Bayes classifier merges traditional Gaussian Naive Bayes with a numerically stable online algorithm for calculating variance. For new training data, Naive Bayes updates relevant entries in the probability table, allowing efficient learning in an online manner. Even though Naive Bayes classifier is known as a loss-less algorithm, this algorithm has a drawback in that it cannot handle multi-modal distributions and quantitative data (Losing et al., 2018; J. Lu, Yang, & Webb, 2006). An application can be the network intrusion detection introduced by Gumus, Sakar, Erdem, and Kursun (Gumus et al., 2014).

Stochastic Gradient Descent (SGD_{Lin}). Stochastic Gradient Descent, also known as incremental gradient descent, is an efficient optimization approach for learning with gradient descent in an iterative manner, allowing update for parameters. This algorithm is capable of handling sparse and high-dimensional data especially combined with linear models in the context of large-scale learning. However, it also has some drawbacks in that it is sensitive to feature scaling and requires a number of hyper-parameters (Bottou, 2010; Losing et al., 2018). Recent applications are large-scale learning for image classification and video processing (Akata, Perronnin, Harchaoui, & Schmid, 2014; Sapienza, Cuzzolin, & Torr, 2014).

Incremental Learning Vector Quantization (ILVQ). Incremental Learning Vector Quantization (ILVQ) is an incremental version of the static Generalized Learning Vector Quantization (GLVQ) that grows dynamically, inserting new prototypes for each class when necessary and adjusting the number and value of prototypes during learning (Losing et al., 2018; Y. Xu, Furao, Hasegawa, & Zhao, 2009). Y. Xu et al. (2009) stated that "ILVQ network can grow gradually and store the learned patterns perfectly so that it can realize the incremental learning well" (p. 8).

Incremental Extreme Learning Machine (IELM). Incremental Extreme Learning Machine (IELM) is an adaption of the batch ELM least-squares approach to a sequential scheme (Liang, Huang, Saratchandran, & Sundararajan, 2006). Aiming for concept drift cases, one study proposed the approach that dynamically changes the number of the neurons in hidden layers in the classical ELM model. From the experiment, the study found that this algorithm provides higher and more stable accuracy as well as can be simplest from the structure perspective in stream environment (S. Xu & Wang, 2016).

Online K-Means K-Means is an unsupervised learning algorithm that clusters data into groups. In a streaming setting, online K-Means algorithm keeps the model up-to-date by applying changes over time. It usually utilizes the concept of forgetfulness to change the update rule in the K-Means algorithm. That is, in the online setting, a new parameter is added that balances the relative importance of new data with past history. An example of application is web-user clustering and web-log clustering.

In addition to the incremental and online learning algorithms described above, other Bayesian, linear, and instance-based models, as well as tree-ensembles and online-based neural networks methods, has been proposed by many researchers. Even though new versions of the algorithms are continuously proposed, it is often not clear which of them is appropriate for a particular task and how they performed compared to each other (Losing et al., 2018). However, in many cases, it is reasonable to expect the needs for incremental and online learning algorithms for various reasons such as large-scale processing and non-stationary environment with possible strong drift. Therefore, incrementality of the algorithm should sometimes be considered seriously and integrated into artificial learning systems from the design perspective.

2.2.1 Incremental Approaches for Network Intrusion Detection

Even though many traditional machine learning algorithms inherently support incremental learning, not all algorithms are applicable for DoS and DDoS attack detection because some of them fail to meet the following conditions which are essential requirements in this problem domain.

1. The model should be able to continuously incorporate new information into the model without a complete retraining. It should be capable of learning new types of attacks, i.e. new attack type, as well as dynamic updates of normal traffic.
2. Preservation of previous information and without the effect of catastrophic forgetting phenomenon (French, 1999).

3. Only a limited number of training examples are allowed to be maintained.

In the literature, only a few studies use incremental algorithms for network intrusion detection due to various issues in the dynamic updates of normal as well as attack data (Bhuyan et al., 2012). This subsection presents some literature related to intrusion detection methods using incremental and online approaches.

In 2008, a new anomaly detection method that dynamically updates normal system usage patterns was proposed by Ren, Hu, Liang, Liu, and Ren. A new program behavior is inserted into old profiles according to density-based incremental clustering method every time the pattern of system usage changes. This method is more efficient than the traditional re-clustering method (Ren et al., 2008).

In 2009, Yu and Lee proposed the incremental learning method of combining cascade service classifier with incremental tree inducer (ITI) for network anomaly detection. This method is composed of three phases: training, testing, and incremental learning phase. In the training phase, the service classifier partitions the training dataset according to the service type (e.g., FTP, SMTP, and TELNET), and ITI method is trained with the instances that have the same service value. In the testing phase, it finds the cluster to which each test instance belongs, and the ITI method is evaluated with the instances. In the incremental learning phase, the incremental learning method of nearest neighbor combination rule is used to train the existing ITI tree (Yu & Lee, 2009).

In 2014, online Naive Bayes classification was proposed by Gumus et al. to dynamically adapt to the recent attacks. In this work, the online Naive Bayes classifier distinguishes DoS and normal connections by updating the mean and standard deviation of each feature incrementally over time. This method was evaluated against KDD99 intrusion detection dataset (Gumus et al., 2014).

In addition to supervised incremental approaches, semi-supervised and unsupervised incremental methods have been proposed. Rasoulifard, Bafghi, and Kahani presented the incremental hybrid intrusion detection system that combines the incremental misuse detection with incremental anomaly detection. Their proposed incremental IDS is based on the ensemble of weak classifiers with about 50% correct classification accuracy, forms a final prediction, and is suitable for real-time intrusion detection (Rasoulifard et al.,

2008). Also, the anomaly detection with incremental clustering based method, namely ADWICE, is proposed by Burbeck and Nadjm-Tehrani to deal with dynamic network environments. The authors proposed the adaptive normality model with two incremental mechanisms: incremental extension with new elements of normal behavior, and a new feature that can forget outdated elements of normal behavior (Burbeck & Nadjm-Tehrani, 2007). As another example of an incremental clustering approach for intrusion detection, Zhong and Li presented the incremental clonal selection algorithm. This method optimizes clustering results in every iteration using the clonal selection algorithm which is based on a partitioning approach. The authors claimed this algorithm achieved high detection rate and low false positive rate in the experiments (Zhong & Li, 2008).

2.2.2 Incremental Support Vector Machine Classification

The concept of Support Vector Machine was introduced by Cortes and Vapnik. The SVM algorithm builds an optimal hyper-plane or a decision boundary in the form of support vectors based on a given dataset to determine to which category the new data belongs (Chauhan, Mishra, & Kumar, 2011; Cortes & Vapnik, 1995). The Support Vector Machine performed well for many classification problems in non-incremental ways. However, the traditional SVM is not able to handle large-scale data efficiently due to computation complexity and extensive memory requirements. To cope with these limitations, the basic incremental support vector machine classification was proposed by Syed et al.. This method preserves the support vectors from the previous learning step and merges them into the next training data for the next training phase as shown in the Figure 2.2. However, it gives only approximate results unless all previously seen data is contained in the set of candidate vectors because the discarded samples still contain some information about classification (Chitrakar & Huang, 2014). The results of their work demonstrated that the support vectors obtained by the SVM algorithm is a minimal set for incremental training, and it would be crucial to remove any of them since they form vital information of the original data space (Syed et al., 1999). Later, many researchers have proposed new and modified incremental SVM methods for better results.

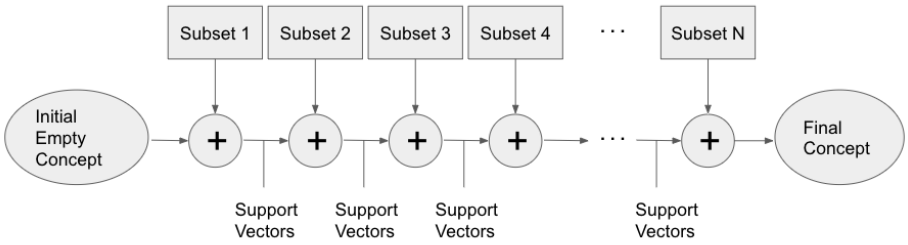


Figure 2.2. The simple incremental SVM training procedure (Syed et al., 1999)

In 2008, W. Wang explained that not only the chosen support vectors but also the redundant vectors near the margin hyperplanes in each increment have big potential to become support vectors in the next increment step. Based on this idea, they proposed an improved incremental SVM techniques, namely redundant incremental SVM (RISVM) (W. Wang, 2008).

Also, incremental SVM based on the reserved set (RS-ISVM) was developed by Yi et al. to deal with network intrusion detection. As seen from Figure 2.3, this method selects samples located in the ring region between two concentric circles with radii of R_1 and R_2, respectively, and adds them to the reserved set because they have higher probabilities to become support vectors in the following learning phase. The weight of each sample in the reserve set is calculated according to the distance to the hyperplane and considered as candidate vectors in the next training step. Their experiments showed that this method is superior to the simple ISVM in terms of both detection rate, false alarm rate, and training and testing time (Yi et al., 2011).

In 2014, Chitrakar and Huang introduced the modified version of the concentric circle method for incremental SVM. In this study, the authors pointed out that data points outside the outer circle should not be excluded from being considered as candidate support vectors and modified the algorithm for them to have opportunity to become candidate support vectors in the next training process. Also, they proposed a half-partition method, namely CSV-ISVM, based on the fact that the possible rotations of hyperplanes for the next increment are not affected by outer halves of the classes as illustrated in Figure 2.4. In the experiment, the result of the final iteration showed that RS-ISVM and

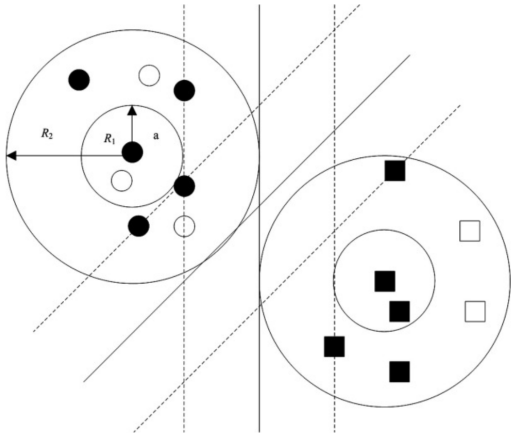


Figure 2.3. The concentric circle strategy (Yi et al., 2011)

CSV-ISVM algorithm achieved 89.817% and 90.148%, respectively in terms of the detection rate (Chitrakar & Huang, 2014). However, the half-partition strategy is designed to work for binary classification and is not suitable for multi-class classification problem because data points in the outer half of one class can become the support vectors in the next iteration in the multi-class classification problem.

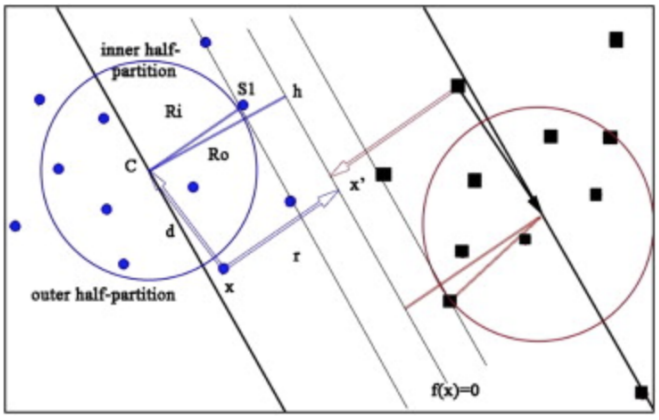


Figure 2.4. The half-partition strategy (Chitrakar & Huang, 2014)

2.3 Datasets for Intrusion Detection System Evaluation

One of the main challenges that researchers have faced is to test and evaluate an intrusion detection system. Historically, most intrusion detection researches, including DoS and DDoS attack detection, have used various datasets to evaluate the performance of their proposed methodologies. However, most existing IDS evaluation datasets have several shortcomings in that they are out of date, lack traffic diversity, and do not reflect current trends of attack patterns (Sharafaldin et al., 2018). Based on the survey conducted by Bhuyan et al., most incremental network intrusion detection methods have been evaluated using either the KDD99 dataset in an offline mode or the network trace based on their own testbed in online mode (Bhuyan et al., 2012). However, the KDD99 dataset has been constantly criticized by many researchers over the last ten years due to the deficiencies such as a large number of redundant records, duplicate records, outdated data, unrealistic network traffic, and no exact definition of the attacks (Sharafaldin et al., 2018; Tavallaee, Bagheri, Lu, & Ghorbani, 2009). Even though NSL-KDD, a refined version of KDD99 dataset, has been proposed by Tavallaee et al. to solve the inherent problems of KDD99 dataset, some problems in KDD99 still remain in NSL-KDD in terms of outdated data and lack of traffic diversity. As such, many researchers had difficulty finding a suitable and valid dataset to evaluate their proposed techniques in both qualitative and quantitative perspectives (Koch, Golling, & Rodosek, 2014; Tavallaee et al., 2009).

Considering the drawbacks of the existing datasets, the guidance of creating a comprehensive and valid IDS dataset has been suggested by Gharib, Sharafaldin, Lashkari, and Ghorbani in 2016. They suggested eleven characteristics of a framework to build a reliable IDS benchmark dataset as follow: complete network configuration, complete traffic, labeled dataset, complete interaction, complete capture, available protocols, attack diversity, anonymity, heterogeneity, feature set, and metadata. In this regard, they pointed out that KDD99 dataset has shortcomings in terms of the following items: complete traffic, attack diversity, and available protocols (Gharib et al., 2016).

In 2018, Sharafaldin et al. introduced a new intrusion detection dataset named CICIDS2017 covering all the suggested eleven criteria. Captured from 3rd July 2017 to 7th July 2017, this dataset consists of a 5 days network traffic created with computers using various operating systems such as Windows Vista, Mac, Ubuntu, and Kali. This dataset contains benign traffic and 14 different modern attack patterns such as Brute Force FTP, DoS, Heartbleed, Web Attack, Botnet, Infiltration, DDoS, and so on. The data is provided in both a) the PCAP format (which includes full packet payloads) and b) comma-separated values (CSV) formatted files for machine and deep learning purposes (Sharafaldin et al., 2018).

As new and diverse attack patterns emerge, new valid datasets having modern attack patterns need to be created regularly and made available to researchers for a valid evaluation of IDS. However, it is a challenging and expensive task to build such a dataset because it requires substantial effort to meet the above eleven criteria suggested by Gharib et al. (Gharib et al., 2016).

2.4 Summary

This chapter provided a review of the literature relevant to online and incremental learning and datasets for intrusion detection system evaluation. The next chapter provides the research design of this study.

CHAPTER 3. RESEARCH DESIGN

This study is exploratory research that attempts to connect an idea of KNN-based candidate support vector selection strategy to an incremental SVM method, and there is no formal hypothesis for this study.

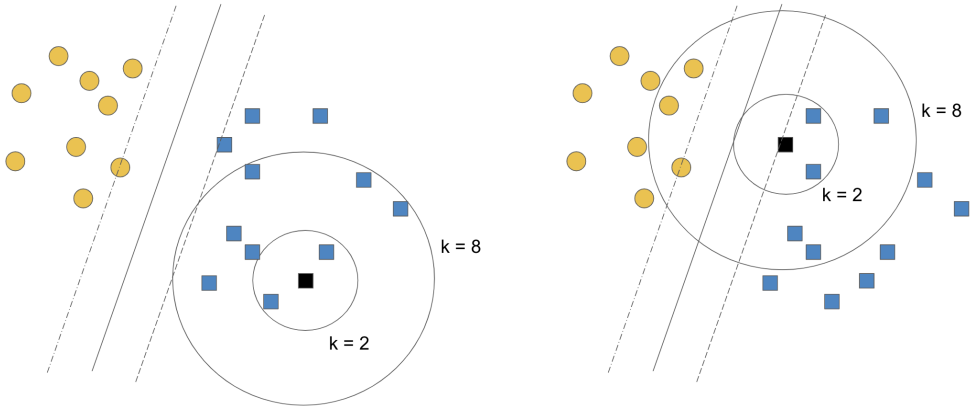
The main objectives of this work are (1) to apply the incremental SVM algorithm combined with the KNN based candidate support vector selection strategy into DoS and DDoS attack detection system and (2) to evaluate the results by comparison with the standard SVM and simple version of ISVM in terms of the evaluation criteria such as F1-score and training and test duration. With these objectives, this study focuses on exploring the advantages of the KNN-based incremental SVM method for DoS and DDoS detection over the standard SVM learning.

3.1 Incremental SVM based on KNN-based Sample Selection

As stated by Syed et al., the support vectors obtained by the SVM algorithm is a minimal set for incremental training, and it would be crucial to remove any of them since they form vital information of the original data space (Syed et al., 1999). However, it gives only approximate results because the discarded samples still contain some information about classification (Chitrakar & Huang, 2014). Therefore, this study proposes the incremental SVM algorithm combined with a strategy of selecting candidate vectors using a k-nearest neighbors (KNN) algorithm. In this study, both incremental SVM methods and the standard SVM method take Radial Basis Function (RBF) kernel SVM as the kernel function.

3.1.1 KNN-based Candidate Support Vector Selection Strategy

Not only the previous support vectors but also other samples have potential to become support vectors in the next incremental step. Considering that critical patterns in SVM are located near the decision boundary, this study uses the KNN algorithm to examine the chances of each sample becoming support vectors in the next learning step. As shown in Figure 3.1(a), the data points lying farther from the hyper-plane tend to have less probability of becoming support vectors, and they are less likely to contain data samples from other classes in the circle. On the contrary, the data samples which lie near a hyper-plane are more likely to contain other classes of data points in the circle as illustrated in Figure 3.1(b). Based on this idea, the probability of becoming a candidate vector is evaluated against the probability estimates obtained from the KNN algorithm. In this algorithm, if the probability estimates for the data sample by the KNN Classifier is smaller than a threshold value, the sample is added to the candidate support vector set for the next iteration phase. The KNN-based candidate support vector selection algorithm is presented in Algorithm 3.1.



(a) k-Nearest Neighbors for data sample 1 (b) k-Nearest Neighbors for data sample 2

Figure 3.1. k-Nearest Neighbors of samples in SVM algorithm

Algorithm 3.1 KNN based Candidate Support Vector Selection

function SELECTCSV(*sample_set* (X, y), k)
    // Initialize CSV set array and probability estimates array
    $candidate\_vectors = [\,]$
    $probability\_estimates = [\,]$
    $neigh = KNeighborsClassifier(n\_neighbors = k)$
    // Fit the KNN model using X as training data and y as target values
    $neigh.fit(X, y)$
    // Get probability estimates for every sample in the sample set X.
    $probability\_estimates = neigh.predict\_proba(X)$
    for *sample* in *sample_set* **do**
        if $probability\_estimates[sample][actual\_class] < threshold$ **then**
            $candidate\_vectors$.add($sample$)
        end if
    end for
    return $candidate\_vectors$
end function

In this method, the candidate support vector (CSV) selection process is performed to determine the importance of each sample every time a new incremental data subset appears. The entire process incremental SVM algorithm with the KNN-based candidate support vector selection is described in Algorithm 3.2. In addition, each class weight defined as the inverse of class frequencies in the input data subset is used for the support vector classification in order to handle imbalanced classes. This class weighting process is performed on every iteration.

Algorithm 3.2 Pseudo-code of ISVM algorithm combined with the KNN-based CSV selection strategy

Input: Previous CSV set CSV_{i-1} and new incremental sample set X_i
Output: The updated classifier SVM and the updated CSV set

Step 1: Add CSV_{i-1} set into the new incremental sample set X_i
Step 2: Get the classifier SVM_i and the support vector set SV_i by training on X_i;
Step 3: Get the CSV set
    Step 3.1: Build the KNN classifier with a given k
    Step 3.2: Train the KNN classifier with X_i
    Step 3.3: Calculate probability estimates for each sample in the data X_i
    Step 3.4: Get the CSV_i set according to Algorithm 3.1
Step 4: Append SV_i to CSV_i
Step 5: Set SVM_i and CSV_i as the output

3.2 Increment Setting

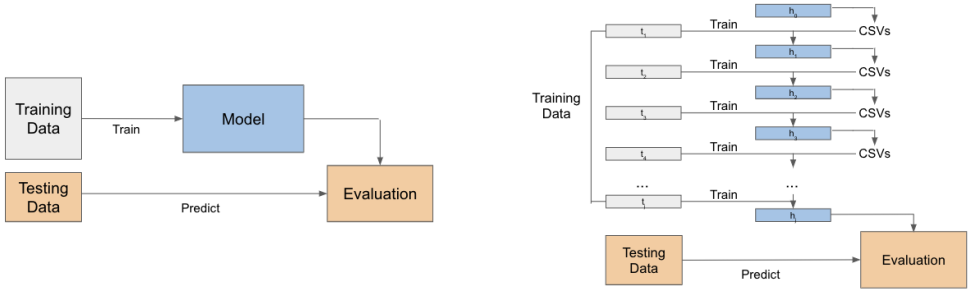
This study examines the performance of the algorithms in two incremental learning settings: (1) the batch incremental learning setting and (2) the class incremental learning setting. In the batch incremental learning setting, the SVM classifier is trained with a small subset of training data at each learning step, and the learning step is repeated until all training data is consumed. In the class incremental learning setting, the training data in each incremental learning phase contains benign data and one class of data that does not appear in the previous learning phase. The performance results of this setting indicate the learning ability of new classes of intrusions that do not exist in the previous training dataset for incremental misuse detection.

3.3 Experiment Design

This section describes the experiment design in two different settings, namely offline and online setting, that reflects various aspects of the incremental algorithm performance.

3.3.1 Off-line Setting

The performance of the incremental SVM algorithm is compared with the standard SVM in an off-line setting according to the performance evaluation criteria defined in Section 3.4. For the standard SVM learning method in the off-line setting, the whole training data are input to the model once and processed by the training system. In the test phase, the model is evaluated against the testing data as shown in Figure 3.2(b)



(a) Evaluation schema for standard SVM learning (b) Evaluation schema for incremental SVM learning

Figure 3.2. Evaluation schema in offline setting

For the incremental learning in the off-line setting, training data is processed sequentially in a predefined order, and the algorithm generates a sequence of models h_1, h_2, h_3, ... h_j. In the evaluation phase, all preceding models are neglected and only the last constructed model is used to evaluate the performance against the testing data as shown in Figure 3.2(b). This schema can be useful when training datasets are very large that it is not feasible to train them all at once, but the performance of the last trained model is required to be as accurate as possible.

3.3.2 On-line Setting

In addition to the offline setting, an online setting is adopted for incremental online learning. In this setting, all the intermediate models are evaluated in every iteration as shown in Figure 3.3. This setting provides a deeper insight in terms of immediate prediction.

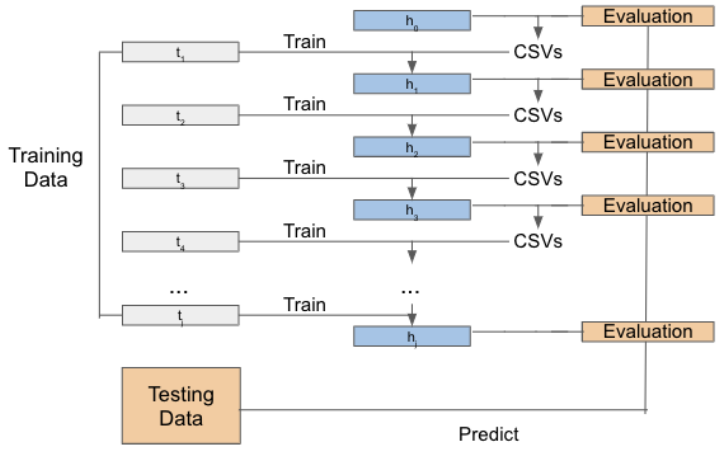


Figure 3.3. Evaluation schema for incremental SVM in online setting

3.4 Performance Evaluation Criteria

With the labeled testing dataset, the predicted label and the actual label can be compared. This research evaluates the performance by measuring the parameters such as precision, recall, and F1-score.

3.4.1 Confusion Matrix

Before introducing the measures used for evaluation, this subsection describes the performance indicators in a confusion matrix that will be used to calculate the performance of the classification models.

- True Positive (TP): The attack instance correctly identified as attack
- True Negative (TN): The benign instance correctly identified as benign
- False Positive (FP): The benign instance incorrectly identified as attack
- False Negative (FN): The attack instance incorrectly identified as benign

The four values in the confusion matrix are used for evaluation of multiclass classification. The classifier of this study classifies the data samples into 6 classes including Benign, DDoS, and four types of DoS attacks. An example of a confusion matrix for class 4 is shown in Figure 3.4.

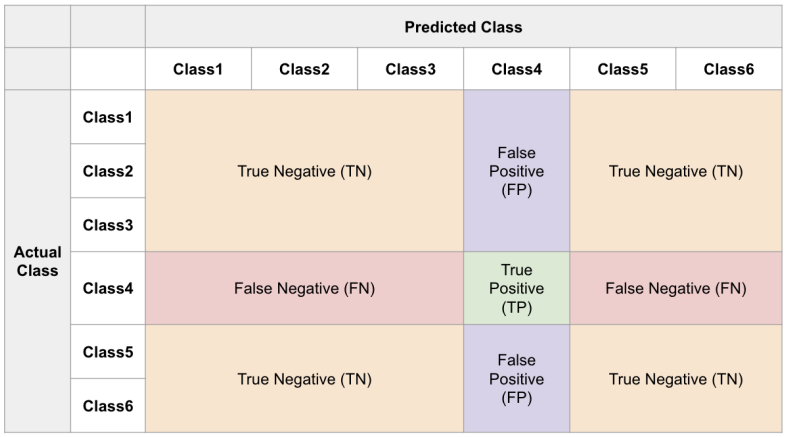


Figure 3.4. An example of confusion matrix for class 4

3.4.2 Performance Measurements

The results of this study are evaluated and compared according to the following performance measurements.

- **Precision:** Precision is the metric that measures the ratio of correctly predicted positive data to total predicted positive data.

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

- **Recall (True positive rate):** Recall is the metric that measures the ratio of correctly predicted positive data to total data in the actual class. This recall value usually denotes the detection rate for intrusion detection systems.

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

- **F1-Score:** F1-score is the weighted average of Precision and Recall. This concept is usually useful when distribution of data is uneven.

$$F1\ Score = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$$

- **Training and test duration:** This study considers the performance of the algorithms in terms of training and test duration because they are important aspects of any modeling to use an incremental algorithm.

3.5 Summary

This chapter described the incremental SVM algorithm based on the KNN-based CSV selection strategy for DoS and DDoS attack detection. In addition, the experimental design is provided to reflect various aspects of the algorithm performance according to the defined evaluation criteria.

CHAPTER 4. EXPERIMENTS

This chapter describes how the experiments were conducted. This includes the evaluated dataset, implementation process, and hyperparameter optimization.

4.1 Dataset and Features

This study uses the CICIDS2017 dataset that reflects the trends of modern network attack scenarios including various network attack patterns. The CICIDS2017 dataset, captured for 5 days starting Monday, July 3rd, 2017, contains benign traffic and the 14 kinds of network attack traffic such as the Brute Force FTP, DoS, Heartbleed, Web Attack, Botnet, Infiltration, and DDoS as shown in Table 4.1 (Sharafaldin et al., 2018).

Table 4.1. Daily label and PCAP file size of CICIDS2017 dataset (Sharafaldin et al., 2018)

Days	**Labels**	**Pcap File Size (GB)**
Monday	Benign (Normal human activities)	10.1
Tuesday	Brute Force, FTP-Patator and SSH-Patator	10.3
Wednesday	DoS slowloris DoS Slowhttptest DoS Hulk DoS GoldenEye Heartbleed Port 444 Attacks	12.5
Thursday	Web Attack - Brute Force Web Attack -XSS Web Attack - Sql Injection Infiltration Attacks - Dropbox download Infiltration - Cool disk	7.7
Friday	Botnet ARES Port Scan DDoS LOIT	8.2

In addition to PCAP formatted data, the CICIDS 2017 dataset provides comma-separated values formatted files for machine learning and deep learning purposes that are publicly available for researchers. The network flow in the processed files is either labeled as benign or as one of the 14 different kinds of attack. Along with the complete labels for all flows, they provide 78 network traffic features for the flow information. The list of features and descriptions are available in Appendix A. When the files are examined, it appears that they contain 2,830,743 flows in total. Table 4.2 shows the distribution of flow records for benign and intrusive flows in the dataset. For the DDoS attack flows, it consists of UDP, TCP, and HTTP requests generated by the DDoS attack tool called Low Orbit Ion Canon (LOIC), and it accounts for 4.52 percent of the total data flows (Sharafaldin et al., 2018).

Table 4.2. Distribution of flow records in CICIDS2017 dataset

Label	**Number of flows**	**Ratio (%)**
Benign	2,273,097	80.30
DoS Hulk	231,073	8.16
PortScan	158,930	5.61
DDoS	128,027	4.52
DoS GoldenEye	10,293	0.36
FTP-Patator	7,938	0.28
SSH-Patator	5,897	0.21
DoS Slowloris	5,796	0.20
DoS Slowhttp	5,499	0.19
Bot	1,966	0.07
Web Attack - Brute Force	1,507	0.05
Web Attack - XSS	652	0.02
Infiltration	36	Less than 0.01
Web Attack - Sql Injection	21	Less than 0.01
Heartbleed	11	Less than 0.01
Total	2,830,743	100

4.2 Implementation

This section describes the tools and methods used in this study and the implementation process covering experiment environment, data pre-processing, and creation of training and test data.

4.2.1 Environment

One of the evaluation criteria of this study is the training and test duration of the algorithm. However, the training and test duration may vary depending on the performance of the computer used. Because of this, the technical specifications of the computer used in the implementation are shared below.

- Central Processing Unit (CPU): 2.3 GHz Intel Core i5
- Random Access Memory (RAM): 16 GB 2133 MHz LPDDR3
- Operating System (OS): macOS High Sierra Version 10.13.6

4.2.2 Data Preprocessing

Data preprocessing is performed before applying the learning algorithms. Data preprocessing in this study includes data integration, data cleansing, feature elimination, feature standardization, and creation of training and testing data.

- Data integration: This study uses eight comma-separated values (CSV) formatted files with labeled network traffic analysis results. Each file has the traffic analysis result with the 79 features but contains different types of attacks and different time period. Before the data cleansing phase, the eight files are concatenated into one single file so that it contains all information for a total of five days.

- Data cleansing: One important step to consider in the data preprocessing phase is to convert non-numerical properties such as categorical and string values into numerical values used in machine learning algorithms. This study confirmed that the provided comma-separated values formatted files do not contain non-numerical values except a label tag as they are already refined for machine and deep learning purposes. However, ”Flow Bytes/s” and ”Flow Packets/s” attributes contain the values ”Infinity” and ”Nan” in addition to numerical values. This study removed all rows that have ”Infinity” in the features and modified ”Nan” to zero to make them suitable for machine learning algorithms.

- Feature elimination: This study identified that the following features have the same value in all samples: Bwd SH Flags, Bwd URG Flags, Fwd Avg Bytes/Bulk, Fwd Avg Packets/Bulk, Fwd Avg Bulk Rate, Bwd Avg Bytes/Bulk, Bwd Avg Packets/Bulk, and Bwd Avg Bulk Rate. These features are removed to improve the performance on this high-dimensional dataset.

- Feature scaling: Feature scaling was used to standardize the range of features of data since the range of values of data varies widely. In the application phase, the class named ”StandardScaler” in preprocessing module of Scikit-Learn library was used (*sklearn preprocessing StandardScaler*, n.d.).

4.2.3 Creation of Training and Test Data

Unlike the KDD99 and NSL-KDD datasets, the CICIDS2017 does not have separate files dedicated to training and testing. Therefore, this study divides the dataset into two parts with the ratio of 80% and 20% for training and test subsets, respectively. For the implementation of this part, the class named ”train_test_split” in Scikit-Learn library was used (*sklearn model selection train and test split*, n.d.). For multiclass classification evaluation, this study performed two types of training data creation for batch-incremental learning and class-incremental learning, respectively.

4.2.3.1 Data Preparation for Batch-Incremental Learning

For batch-incremental learning, the generated training data is divided into a total of 2000 subsets and each subset consists of 1000 data instances as shown in Figure 4.1.

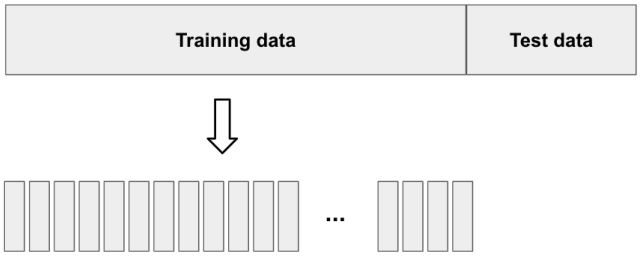


Figure 4.1. Data preparation for batch-incremental learning

4.2.3.2 Data Preparation for Class-Incremental Learning

For class-incremental learning, the generated training data is divided into 6 subsets as seen in figure 4.2, and each subset contains 1000 samples of benign data and one type of DoS attack or DDoS attack that does not appear in the previous learning phase as described in Table 4.3.

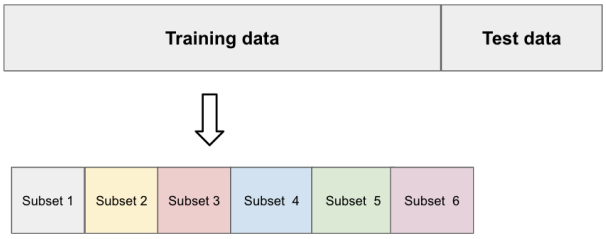


Figure 4.2. Data preparation for class-incremental learning

Table 4.3. Contained classes in each subset for class-incremental learning setting

Subset	Contained classes
Subset 1	Benign, Other types of attack
Subset 2	Benign, DoS Slowhttp
Subset 3	Benign, DoS Hulk
Subset 4	Benign, DDoS
Subset 5	Benign, DoS GoldenEye
Subset 6	Benign, DoS Slowloris

4.3 Hyper-parameter Setting

A better SVM classification accuracy can be achieved through proper tuning of hyper-parameters. This study will take Radial Basis Function (RBF) kernel SVM as the kernel function. Accordingly, two parameters, C and gamma (γ), should be determined in the SVM model. However, as stated by Losing et al., optimizing hyperparameter using the whole training set is usually not possible in practice and contradicts the paradigm of incremental learning. Regarding this issue, they claimed that hyper-parameters can be robustly chosen based on few samples as their experiment results showed that most incremental learning methods perform slightly worse than when all training samples are used (Losing et al., 2018). Therefore, this study determines hyper-parameters only with randomly selected 10000 samples of the training data to be closer to real-life scenarios. To find the best results, grid search across different parameter settings with 5-fold cross-validation is performed as follow (Chen & Lin, 2006; Hsu, Chang, Lin, et al., 2003).

Step 1 Consider a grid space of (C, γ) with $\log_{10} C \in \{-3, -2, \ldots, 8\}$ and $\log_{10} \gamma \in \{-5, -4, \ldots, 4\}$

Step 2 For each hyper parameter pair (C, γ) in the search space, conduct 5-fold cross-validation on the training set.

Step 3 Choose the parameter (C, γ) that leads to the lowest cross-validation error classification rate.

4.4 Summary

In this chapter, the evaluated dataset, implementation process, and hyperparameter optimization are described.

CHAPTER 5. RESULTS

This chapter provides the performance results of the KNN-based incremental SVM method and compares them with the standard SVM method according to the performance evaluation criteria defined in Chapter 3.

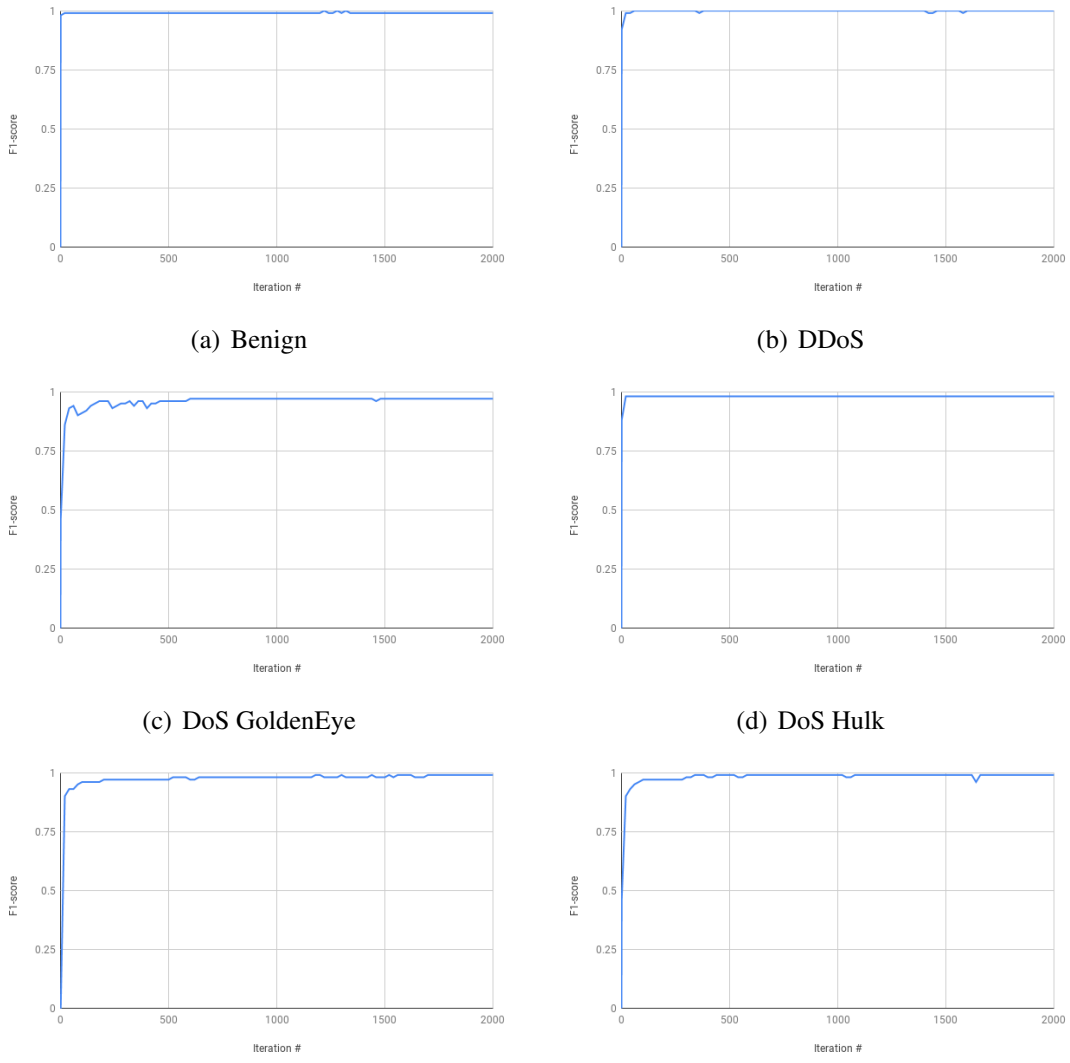
5.1 Results of Batch-Incremental SVM Learning

This section provides the results of the batch-incremental SVM learning using KNN-based CSV selection strategy conducted in the offline and online setting. First, Table 5.1 provides the result of the standard SVM method in order to compare with the data obtained from the KNN-based ISVM method. Table 5.2 shows the performance scores of the last constructed model of the KNN-based ISVM method. Also, the evaluation results in the online setting are illustrated in Figure 5.1 and provided in Appendix B.

Table 5.1. The scores of the standard SVM

	Precision	Recall	F1-score
Benign	1.00	0.99	0.99
DDoS	1.00	1.00	1.00
DoS GoldenEye	0.96	0.99	0.97
DoS Hulk	0.97	1.00	0.98
DoS Slowhttptest	0.98	0.99	0.98
DoS Slowloris	0.99	0.99	0.99

Table 5.3 shows the training and test duration of the standard SVM and the KNN-based ISVM in offline setting.



(a) Benign

(b) DDoS

(c) DoS GoldenEye

(d) DoS Hulk

(e) DoS Slowhttp

(f) DoS Slowloris

Figure 5.1. The scores of the KNN-based ISVM for batch-incremental learning in online setting

Table 5.2. The scores of the KNN-based ISVM for batch-incremental learning in offline setting (k=10, $threshold$=0.6)

	Precision	Recall	F1-score
Benign	1.00	0.99	0.99
DDoS	1.00	1.00	1.00
DoS GoldenEye	0.96	0.99	0.97
DoS Hulk	0.96	1.00	0.98
DoS Slowhttptest	0.99	0.99	0.99
DoS Slowloris	0.98	1.00	0.99

Table 5.3. Comparison of training and test duration for batch-incremental learning in offline setting

	Standard SVM	KNN-based ISVM
Training duration (s)	187529.902	198205.061
Test duration (s)	1958.586	1432.434

5.2 Results of Class-Incremental SVM Learning

This section provides the results of the class-incremental SVM learning using the KNN-based CSV selection strategy in the offline and online setting. The results are compared with the standard SVM and the simple ISVM method which retains only the support vectors and discarded the non-support vectors after each incremental learning process. In order to ensure that the results obtained during the application are solid, the experiments have been performed 10 times in succession. The results obtained are the arithmetic mean of the repeated operations.

Table 5.4, Table 5.5, and Table 5.6 show the performance scores of the standard SVM, the simple ISVM, and the KNN-based ISVM, respectively, for the class-incremental learning in offline setting.

Table 5.4. The scores of the standard SVM for class-incremental learning in offline setting

	Precision	Recall	F1-score
Benign	1.00	0.98	0.99
DDoS	0.99	1.00	1.00
DoS GoldenEye	0.93	0.99	0.96
DoS Hulk	0.93	1.00	0.97
DoS Slowhttptest	0.96	0.99	0.97
DoS Slowloris	0.96	0.99	0.97

Table 5.5. The scores of the simple ISVM for class-incremental learning in offline setting

	Precision	Recall	F1-score
Benign	0.98	0.98	0.98
DDoS	0.78	0.97	0.87
DoS GoldenEye	0.85	0.99	0.91
DoS Hulk	0.95	0.84	0.89
DoS Slowhttptest	0.95	0.74	0.84
DoS Slowloris	0.84	0.99	0.91

Table 5.6. The scores of the KNN-based ISVM for class-incremental learning in offline setting (k=10, $threshold$=0.8)

	Precision	Recall	F1-score
Benign	0.97	0.98	0.98
DDoS	0.80	1.00	0.90
DoS GoldenEye	0.89	1.00	0.94
DoS Hulk	0.96	0.82	0.89
DoS Slowhttptest	0.94	0.93	0.94
DoS Slowloris	0.91	0.99	0.95

Next, Table 5.7, Table 5.8 and Table 5.9 show the scores of the standard SVM, the simple ISVM, and the KNN-based ISVM method, respectively, for the class-incremental learning in online setting. Note that the number of candidate support vector (CSV) in table 5.8 represents the number of support vectors. At last, Table 5.10 shows the training and test duration comparison for the class-incremental learning in offline setting.

Table 5.7. The scores of the standard SVM for class-incremental learning in online setting

Iteration #	Type	Precision	Recall	F1-score	Training duration	Test duration	# of SV
1	Benign	0.86	0.99	0.92	67.835	109.841	2490
	DDoS	0.00	0.00	0.00			
	DoS GoldenEye	0.00	0.00	0.00			
	DoS Hulk	0.00	0.00	0.00			
	DoS Slowhttptest	0.00	0.00	0.00			
	DoS Slowloris	0.00	0.00	0.00			
2	Benign	0.86	0.99	0.92	86.498	139.093	3106
	DDoS	0.00	0.00	0.00			
	DoS GoldenEye	0.00	0.00	0.00			
	DoS Hulk	0.00	0.00	0.00			
	DoS Slowhttptest	0.67	0.99	0.80			
	DoS Slowloris	0.00	0.00	0.00			
3	Benign	0.96	0.99	0.97	1921.614	229.682	5018
	DDoS	0.00	0.00	0.00			
	DoS GoldenEye	0.00	0.00	0.00			
	DoS Hulk	0.78	1.00	0.88			
	DoS Slowhttptest	0.69	0.99	0.81			
	DoS Slowloris	0.00	0.00	0.00			
4	Benign	1.00	0.99	0.99	2576.198	273.015	5876
	DDoS	0.99	1.00	0.99			
	DoS GoldenEye	0.00	0.00	0.00			
	DoS Hulk	0.95	1.00	0.97			
	DoS Slowhttptest	0.70	0.99	0.82			
	DoS Slowloris	0.00	0.00	0.00			
5	Benign	1.00	0.99	0.99	11047.836	340.296	7152
	DDoS	0.99	1.00	1.00			
	DoS GoldenEye	0.93	0.99	0.96			
	DoS Hulk	0.96	1.00	0.98			
	DoS Slowhttptest	0.73	0.99	0.86			
	DoS Slowloris	0.00	0.00	0.00			
6	Benign	1.00	0.98	0.99	7799.155	416.216	7838
	DDoS	0.99	1.00	1.00			
	DoS GoldenEye	0.93	0.99	0.96			
	DoS Hulk	0.93	1.00	0.97			
	DoS Slowhttptest	0.96	0.99	0.97			
	DoS Slowloris	0.96	0.99	0.97			

Table 5.8. The scores of the simple ISVM for class-incremental learning in online setting

Iteration #	Type	Precision	Recall	F1-score	Training duration	Test duration	# of CSV
1	Benign	0.86	0.99	0.92	192.595	112.205	2533
	DDoS	0.00	0.00	0.00			
	DoS GoldenEye	0.00	0.00	0.00			
	DoS Hulk	0.00	0.00	0.00			
	DoS Slowhttptest	0.00	0.00	0.00			
	DoS Slowloris	0.00	0.00	0.00			
2	Benign	0.85	0.99	0.92	11.417	134.630	2998
	DDoS	0.00	0.00	0.00			
	DoS GoldenEye	0.00	0.00	0.00			
	DoS Hulk	0.00	0.00	0.00			
	DoS Slowhttptest	0.57	1.00	0.78			
	DoS Slowloris	0.00	0.00	0.00			
3	Benign	0.92	0.98	0.95	4473.617	298.013	6443
	DDoS	0.00	0.00	0.00			
	DoS GoldenEye	0.00	0.00	0.00			
	DoS Hulk	0.87	0.99	0.93			
	DoS Slowhttptest	0.68	0.99	0.80			
	DoS Slowloris	0.00	0.00	0.00			
4	Benign	0.96	0.98	0.97	78.712	218.092	4717
	DDoS	0.77	1.00	0.87			
	DoS GoldenEye	0.00	0.00	0.00			
	DoS Hulk	0.94	0.84	0.88			
	DoS Slowhttptest	0.67	0.90	0.78			
	DoS Slowloris	0.00	0.00	0.00			
5	Benign	0.98	0.98	0.98	21.949	260.762	5550
	DDoS	0.78	1.00	0.89			
	DoS GoldenEye	0.85	0.99	0.91			
	DoS Hulk	0.95	0.84	0.89			
	DoS Slowhttptest	0.69	0.87	0.78			
	DoS Slowloris	0.00	0.00	0.00			
6	Benign	0.98	0.98	0.98	22.577	283.117	5934
	DDoS	0.78	0.97	0.87			
	DoS GoldenEye	0.85	0.99	0.91			
	DoS Hulk	0.95	0.84	0.89			
	DoS Slowhttptest	0.95	0.74	0.84			
	DoS Slowloris	0.84	0.99	0.91			

Table 5.9. The scores of the KNN-based ISVM for class-incremental learning in online setting (k=10, $threshold$=0.8)

Iteration #	Type	Precision	Recall	F1-score	Training duration	Test duration	# of CSV
1	Benign	0.86	0.99	0.92	235.837	111.795	3277
	DDoS	0.00	0.00	0.00			
	DoS GoldenEye	0.00	0.00	0.00			
	DoS Hulk	0.00	0.00	0.00			
	DoS Slowhttptest	0.00	0.00	0.00			
	DoS Slowloris	0.00	0.00	0.00			
2	Benign	0.85	0.99	0.92	21.166	142.288	3686
	DDoS	0.00	0.00	0.00			
	DoS GoldenEye	0.00	0.00	0.00			
	DoS Hulk	0.00	0.00	0.00			
	DoS Slowhttptest	0.57	0.99	0.78			
	DoS Slowloris	0.00	0.00	0.00			
3	Benign	0.92	0.98	0.95	3025.550	288.053	7196
	DDoS	0.00	0.00	0.00			
	DoS GoldenEye	0.00	0.00	0.00			
	DoS Hulk	0.92	0.99	0.95			
	DoS Slowhttptest	0.67	0.97	0.82			
	DoS Slowloris	0.00	0.00	0.00			
4	Benign	0.97	0.98	0.97	118.143	219.375	5773
	DDoS	0.78	1.00	0.89			
	DoS GoldenEye	0.00	0.00	0.00			
	DoS Hulk	0.94	0.81	0.87			
	DoS Slowhttptest	0.67	0.98	0.82			
	DoS Slowloris	0.00	0.00	0.00			
5	Benign	0.98	0.98	0.98	36.279	272.695	6766
	DDoS	0.80	1.00	0.90			
	DoS GoldenEye	0.85	0.99	0.92			
	DoS Hulk	0.95	0.84	0.89			
	DoS Slowhttptest	0.71	0.97	0.84			
	DoS Slowloris	0.00	0.00	0.00			
6	Benign	0.97	0.98	0.98	35.323	299.918	7222
	DDoS	0.80	1.00	0.90			
	DoS GoldenEye	0.89	1.00	0.94			
	DoS Hulk	0.96	0.82	0.89			
	DoS Slowhttptest	0.94	0.93	0.94			
	DoS Slowloris	0.91	0.99	0.95			

Table 5.10. Comparison of training and test duration for class-incremental learning in offline setting

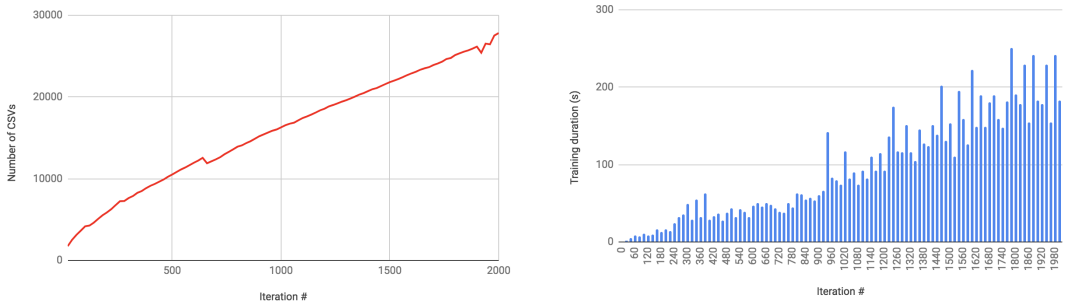
	Standard SVM	Simple ISVM	KNN-based ISVM
Training duration (s)	7799.155	4800.871	3472.300
Test duration (s)	416.216	283.117	299.918

CHAPTER 6. DISCUSSION AND CONCLUSIONS

This study proposed the incremental SVM method combined with k-nearest neighbors (KNN) based candidate support vectors selection strategy. Specifically, this thesis focused on the multi-class classification problem in batch-incremental and class-incremental SVM learning for DoS and DDoS attack detection.

In the batch-incremental learning setting, the last constructed model of the KNN-based ISVM method achieved almost the same scores as the standard SVM in terms of precision, recall, and F1-score. However, as seen in Figure 5.1(b), Figure 5.1(c), and Figure 5.1(f), the results of the batch-incremental learning in the online setting show that there can be an oscillation problem in the score of the KNN-based ISVM method when a new data subset is added.

In terms of the training duration, as a new batch of samples is added continuously, the number of candidate support vectors increases as shown in Figure 6.1(a), and thereby it makes the size of the training set eventually larger and the training time becomes longer. Therefore, keeping the number of candidate vectors low in each increment should be considered to prevent increases in the overall training and classification time. Also, when comparing the test duration between the standard SVM and the KNN-based ISVM, the KNN-based ISVM takes less than the standard SVM.



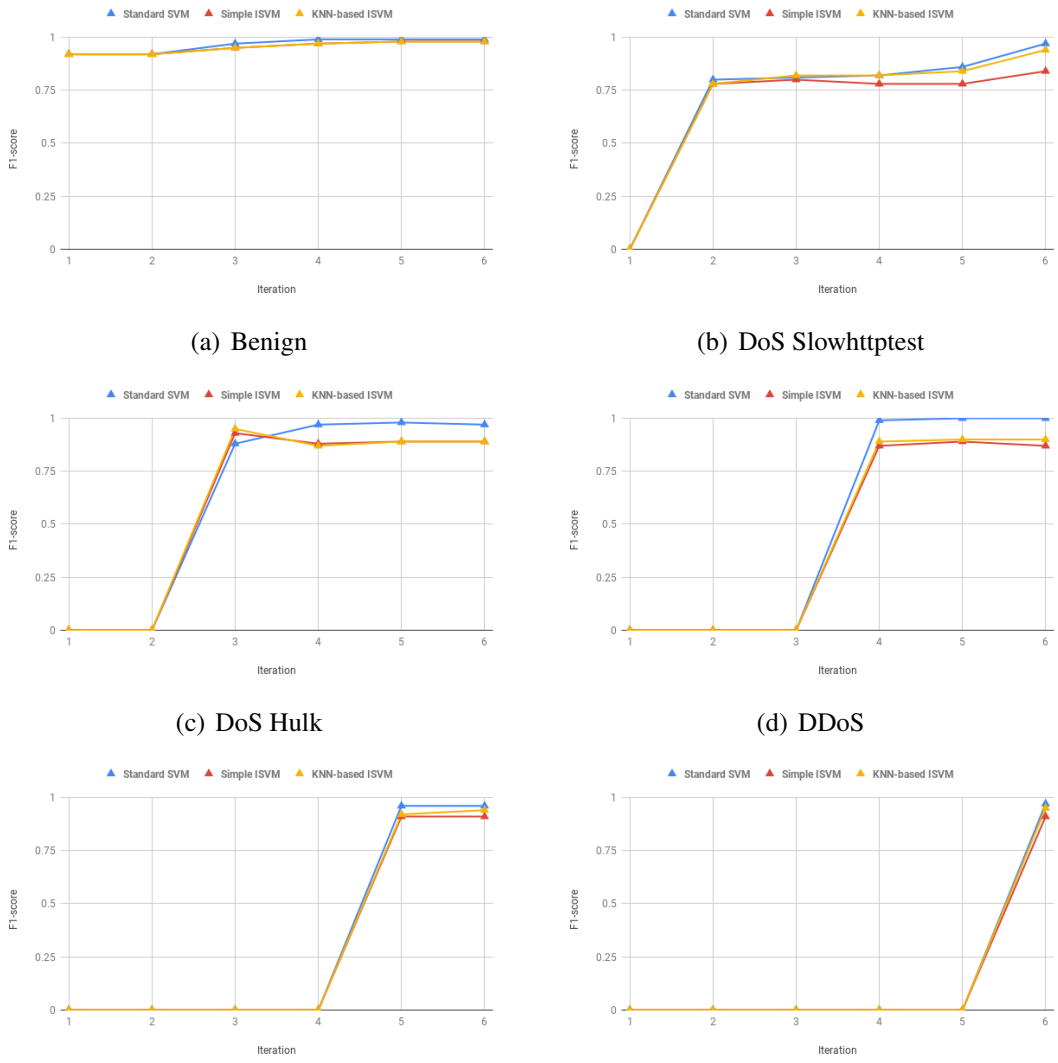
(a) Number of CSVs per iteration

(b) Training duration

Figure 6.1. Number of CSVs and the training duration of the KNN-based ISVM in online setting

In the class-incremental learning setting, the KNN-based ISVM method is compared with two methods: (1) standard SVM method and (2) simple ISVM method. The testing results show that the KNN-based ISVM can learn new pattern classes by incrementally incorporating new attack instances, but the achieved F1-scores are worse than the standard SVM method for all classes. However, the KNN-based ISVM method performed better when compared with the simple ISVM method as seen in table 5.5 and table 5.6. Figure 6.2 shows the compared scores between the standard SVM, the simple ISVM, and the KNN-based ISVM for the class-incremental learning in online setting. In general, the KNN-based ISVM method achieved higher F1-scores than the simple ISVM method. It was because the simple ISVM method discards samples which still can contain vital information of the original data space, on the other hand, the KNN-based ISVM method retains data samples that have a possibility to become support vectors in the next learning phase (Syed et al., 1999). Another notable point is that there is a degradation of the F1-score for both the simple ISVM method and the KNN-based ISVM method when the data subset of the new class is added as seen in Figure 6.2(c).

At last, in terms of the training and test duration, both the KNN-based ISVM and the simple ISVM method was faster than the standard SVM but there was little difference between the simple ISVM and the KNN-based ISVM as seen in Figure 6.3 and Figure 6.4.



(a) Benign

(b) DoS Slowhttptest

(c) DoS Hulk

(d) DDoS

(e) DoS GoldenEye

(f) DoS Slowloris

Figure 6.2. Score comparison between the standard SVM, the simple ISVM, and the KNN-based ISVM for class-incremental learning in online setting

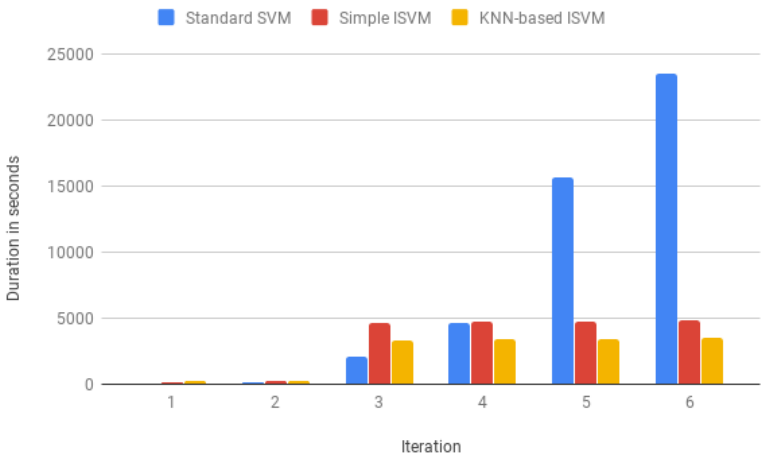


Figure 6.3. Cumulative training duration for class-incremental learning in online setting

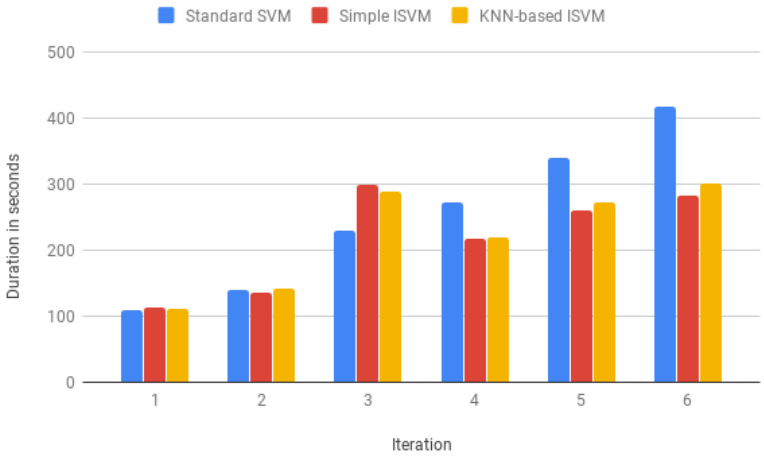


Figure 6.4. Test duration comparison for class-incremental learning in online setting

In conclusion, this study verified whether the incremental SVM method combined with k-nearest neighbors (KNN) based candidate vector selection strategy can achieve the same score as the standard SVM algorithm while shortening the training and test duration. The proposed method in the batch-incremental learning setup achieved almost the same scores as the standard SVM method. However, there was no significant time reduction in training duration, and the experimental results showed that the training time becomes longer as a new batch of sample is added continuously. In terms of the test duration, the KNN-based ISVM method performed faster than the standard SVM. In class-incremental learning setup, the F1-scores of the proposed method was lower than the scores of the standard SVM but higher than the simple ISVM method. However, the KNN-based ISVM method performed faster than the standard SVM method in terms of training and test duration. With respect to these results, there is room for improvement in the KNN-based ISVM method. Still, this study is meaningful in that it expands knowledge in the field of incremental support vector machine and laid the initial groundwork for the new idea of the KNN-based candidate support vector selection strategy.

Finally, this thesis suggests some future works that need to be pursued to improve this study. First, one possible future work is to explore a method to determine the parameter k used in the KNN algorithm and the threshold value used in selecting candidate support vectors because this study found the optimal values for the parameters by simply applying with different parameter settings. A second possible future work is to ease the oscillation phenomenon that occurred as shown in the results of the batch-incremental learning setting. Third, consideration must be given to how the number of candidate support vectors can be kept low in each increment in order to prevent increases in the overall training and classification duration. These unexplored parts will significantly improve the quality of the study. Additionally, this study was conducted using the CICIDS2017 dataset, but the approach can also be applied to other datasets having different data distribution. Therefore, it can be meaningful to explore how the results differ when applied to other datasets other than the CICIDS2017 dataset used in this study.

REFERENCES

Akata, Z., Perronnin, F., Harchaoui, Z., & Schmid, C. (2014). Good practice in large-scale learning for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(3), 507–520.

Akilandeswari, V., & Shalinie, S. M. (2012). Probabilistic neural network based attack traffic classification. In *Advanced computing (icoac), 2012 fourth international conference on* (pp. 1–8).

Behal, S., & Kumar, K. (2017). Characterization and comparison of ddos attack tools and traffic generators: A review. *IJ Network Security*, *19*(3), 383–393.

Bhandari, A., Sangal, A., & Kumar, K. (2015). Destination address entropy based detection and traceback approach against distributed denial of service attacks. *International Journal of Computer Network and Information Security*, *7*(8), 9.

Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2012). Survey on incremental approaches for network anomaly detection. *arXiv preprint arXiv:1211.4493*.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of compstat'2010* (pp. 177–186). Springer.

Burbeck, K., & Nadjm-Tehrani, S. (2007). Adaptive real-time anomaly detection with incremental clustering. *information security technical report*, *12*(1), 56–67.

Cambiaso, E., Papaleo, G., Chiola, G., & Aiello, M. (2013). Slow dos attacks: definition and categorisation. *International Journal of Trust Management in Computing and Communications*, *1*(3-4), 300–319.

Chauhan, A., Mishra, G., & Kumar, G. (2011). Survey on data mining techniques in intrusion detection. *International Journal of Scientific & Engineering Research*, *2*(7), 1–4.

Chen, Y.-W., & Lin, C.-J. (2006). Combining svms with various feature selection strategies. In *Feature extraction* (pp. 315–324). Springer.

Chitrakar, R., & Huang, C. (2014). Selection of candidate support vectors in incremental svm for network intrusion detection. *computers & security*, *45*, 231–241.

Cicflowmeter (formerly iscxflowmeter). (n.d.). Retrieved from `http://www.netflowmeter.ca/netflowmeter.html`

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

Dyn analysis summary of friday october 21 attack — dyn blog. (n.d.). Retrieved from `https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/`

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, *3*(4), 128–135.

Gepperth, A., & Hammer, B. (2016). Incremental learning algorithms and applications. In *European symposium on artificial neural networks (esann).*

Gharib, A., Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2016). An evaluation framework for intrusion detection dataset. In *Information science and security (iciss), 2016 international conference on* (pp. 1–6).

Grafov. (2018, Jun). *grafov/hulk.* Retrieved from `https://github.com/grafov/hulk`

Gumus, F., Sakar, C. O., Erdem, Z., & Kursun, O. (2014). Online naive bayes classification for network intrusion detection. In *Proceedings of the 2014 ieee/acm international conference on advances in social networks analysis and mining* (pp. 670–674).

Hsu, C.-W., Chang, C.-C., Lin, C.-J., et al. (2003). A practical guide to support vector classification.

Jseidl. (2018, Jun). *jseidl/goldeneye.* Retrieved from `https://github.com/jseidl/GoldenEye`

Kandula, S., Katabi, D., Jacob, M., & Berger, A. (2005). Botz-4-sale: Surviving organized ddos attacks that mimic flash crowds. In *Proceedings of the 2nd conference on symposium on networked systems design & implementation-volume 2* (pp. 287–300).

Kaur, P., Kumar, M., & Bhandari, A. (2017). A review of detection approaches for distributed denial of service attacks. *Systems Science & Control Engineering*, *5*(1), 301–320.

Koch, R., Golling, M., & Rodosek, G. D. (2014). Towards comparability of intrusion detection systems: New data sets. In *Terena networking conference* (Vol. 7).

Liang, N.-Y., Huang, G.-B., Saratchandran, P., & Sundararajan, N. (2006). A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on neural networks*, *17*(6), 1411–1423.

Liao, Z., & Couillet, R. (2017). A large dimensional analysis of least squares support vector machines. *arXiv preprint arXiv:1701.02967*.

Losing, V., Hammer, B., & Wersing, H. (2018). Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, *275*, 1261–1274.

Lu, J., Yang, Y., & Webb, G. I. (2006). Incremental discretization for naive-bayes classifier. In *International conference on advanced data mining and applications* (pp. 223–238).

Lu, Y., Boukharouba, K., Boonært, J., Fleury, A., & Lecoeuche, S. (2014). Application of an incremental svm algorithm for on-line human recognition from video surveillance using texture and color features. *Neurocomputing*, *126*, 132–140.

Mahjabin, T., Xiao, Y., Sun, G., & Jiang, W. (2017). A survey of distributed denial-of-service attack, prevention, and mitigation techniques. *International Journal of Distributed Sensor Networks*, *13*(12), 1550147717741463.

Mirkovic, J., & Reiher, P. (2004). A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, *34*(2), 39–53.

Nalepa, J., & Kawulok, M. (2018). Selecting training sets for support vector machines: a review. *Artificial Intelligence Review*, 1–44.

Park, J., Iwai, K., Tanaka, H., & Kurokawa, T. (2014). Analysis of slow read dos attack. In *2014 international symposium on information theory and its applications* (pp. 60–64).

Peng, T., Leckie, C., & Ramamohanarao, K. (2007). Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Computing Surveys (CSUR)*, *39*(1), 3.

Polikar, R., Upda, L., Upda, S. S., & Honavar, V. (2001). Learn++: An incremental learning algorithm for supervised neural networks. *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, *31*(4), 497–508.

Rasoulifard, A., Bafghi, A. G., & Kahani, M. (2008). Incremental hybrid intrusion detection using ensemble of weak classifiers. In *Computer society of iran computer conference* (pp. 577–584).

Ren, F., Hu, L., Liang, H., Liu, X., & Ren, W. (2008). Using density-based incremental clustering for anomaly detection. In *2008 international conference on computer science and software engineering* (Vol. 3, pp. 986–989).

Ruping, S. (2001). Incremental learning with support vector machines. In *Data mining, 2001. icdm 2001, proceedings ieee international conference on* (pp. 641–642).

Saffari, A., Leistner, C., Santner, J., Godec, M., & Bischof, H. (2009). On-line random forests. In *Computer vision workshops (iccv workshops), 2009 ieee 12th international conference on* (pp. 1393–1400).

Sapienza, M., Cuzzolin, F., & Torr, P. H. (2014). Learning discriminative space–time action parts from weakly labelled videos. *International journal of computer vision*, *110*(1), 30–47.

Schneier, B. (2016). Lessons from the dyn ddos attack. *Schneier on Security Blog*, *8*.

Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Icissp* (pp. 108–116).

sklearn model selection train and test split. (n.d.). Retrieved from `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

sklearn preprocessing standardscaler. (n.d.). Retrieved from `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`

Syed, N. A., Huan, S., Kah, L., & Sung, K. (1999). Incremental learning with support vector machines.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. In *Computational intelligence for security and defense applications, 2009. cisda 2009. ieee symposium on* (pp. 1–6).

Wang, A., Wan, G., Cheng, Z., & Li, S. (2009). An incremental extremely random forest classifier for online learning and tracking. In *Image processing (icip), 2009 16th ieee international conference on* (pp. 1449–1452).

Wang, W. (2008). A redundant incremental learning algorithm for svm. In *2008 international conference on machine learning and cybernetics* (Vol. 2, pp. 734–738).

Xu, S., & Wang, J. (2016). A fast incremental extreme learning machine algorithm for data streams classification. *Expert Systems with Applications*, *65*, 332–344.

Xu, Y., Furao, S., Hasegawa, O., & Zhao, J. (2009). An online incremental learning vector quantization. In *Pacific-asia conference on knowledge discovery and data mining* (pp. 1046–1053).

Yi, Y., Wu, J., & Xu, W. (2011). Incremental svm based on reserved set for network intrusion detection. *Expert Systems with Applications*, *38*(6), 7698–7707.

York, K. (2016). Dyn statement on 10/21/2016 ddos attack. *Dyn Blog, October*.

Yu, W.-Y., & Lee, H.-M. (2009). An incremental-learning method for supervised anomaly detection by cascading service classifier and iti decision tree methods. In *Pacific-asia workshop on intelligence and security informatics* (pp. 155–160).

Zeng, Z.-Q., Yu, H.-B., Xu, H.-R., Xie, Y.-Q., & Gao, J. (2008). Fast training support vector machines using parallel sequential minimal optimization. In *2008 3rd international conference on intelligent system and knowledge engineering* (Vol. 1, pp. 997–1001).

Zhong, C., & Li, N. (2008). Incremental clustering algorithm for intrusion detection using clonal selection. In *2008 ieee pacific-asia workshop on computational intelligence and industrial application* (Vol. 1, pp. 326–331).

APPENDIX A. FEATURES IN CICIDS2017 DATASET

This appendix lists the features and descriptions in the processed format (comma-separated values (CSV) formatted files) of the CICIDS2017 dataset used in this study (*CICFlowMeter (Formerly ISCXFlowMeter)*, n.d.).

Table A.1.: Features in CICIDS2017 Dataset

No	Feature Name	Feature Description
1	Destination Port	Destination Port
2	Flow Duration	Flow duration in microsecond
3	Total Fwd Packets	Total packets in the forward direction
4	Total Backward Packets	Total packets in the backward direction
5	Total Length of Fwd Packets	Total size of packet in forward direction
6	Total Length of Bwd Packets	Total size of packet in backward direction
7	Fwd Packet Length Max	Maximum size of packet in forward direction
8	Fwd Packet Length Min	Minimum size of packet in forward direction
9	Fwd Packet Length Mean	Mean size of packet in forward direction
10	Fwd Packet Length Std	Standard deviation size of packet in forward direction
11	Bwd Packet Length Max	Maximum size of packet in backward direction
12	Bwd Packet Length Min	Minimum size of packet in backward direction
13	Bwd Packet Length Mean	Mean size of packet in backward direction
14	Bwd Packet Length Std	Standard deviation size of packet in backward direction
15	Flow Bytes/s	Number of flow bytes per second
16	Flow Packets/s	Number of flow packets per second
17	Flow IAT Mean	Mean length of a flow
18	Flow IAT Std	Standard deviation length of a flow
19	Flow IAT Max	Maximum length of a flow
20	Flow IAT Min	Minimum length of a flow

Table A.1 continued from previous page

No	Feature Name	Feature Description
21	Fwd IAT Total	Total time between two packets sent in the forward direction
22	Fwd IAT Mean	Mean time between two packets sent in the forward direction
23	Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
24	Fwd IAT Max	Maximum time between two packets sent in the forward direction
25	Fwd IAT Min	Minimum time between two packets sent in the forward direction
26	Bwd IAT Total	Total time between two packets sent in the backward direction
27	Bwd IAT Mean	Mean time between two packets sent in the backward direction
28	Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
29	Bwd IAT Max	Maximum time between two packets sent in the backward direction
30	Bwd IAT Min	Minimum time between two packets sent in the backward direction
31	Fwd PSH Flags	Number of packets with PUSH
32	Bwd PSH Flags	Number of times the PSH flag was set in packets traveling in the backward direction (0 for UDP)
33	Fwd URG Flags	Number of times the URG flag was set in packets traveling in the forward direction (0 for UDP
34	Bwd URG Flags	Number of times the URG flag was set in packets traveling in the backward direction (0 for UDP)

Table A.1 continued from previous page

No	Feature Name	Feature Description
35	Fwd Header Length	Total bytes used for headers in the forward direction
36	Bwd Header Length	Total bytes used for headers in the backward direction
37	Fwd Packets/s	Number of forward packets per second
38	Bwd Packets/s	Number of backward packets per second
39	Min Packet Length	Minimum inter-arrival time of packet
40	Max Packet Length	Maximum inter-arrival time of packet
41	Packet Length Mean	Mean inter-arrival time of packet
42	Packet Length Std	Standard deviation inter-arrival time of packet
43	Packet Length Variance	Packet Length Variance
44	FIN Flag Count	Number of packets with FIN
45	SYN Flag Count	Number of packets with SYN
46	RST Flag Count	Number of packets with RST
47	PSH Flag Count	Number of packets with PUSH
48	ACK Flag Count	Number of packets with ACK
49	URG Flag Count	Number of packets with URG
50	CWE Flag Count	Number of packets with CWE
51	ECE Flag Count	Number of packets with ECE
52	Down/Up Ratio	Download and upload ratio
53	Average Packet Size	Average size of packet
54	Avg Fwd Segment Size	Average size observed in the forward direction
55	Avg Bwd Segment Size	Average size observed in the backward direction
56	Fwd Header Length	Header length in the forward direction
57	Fwd Avg Bytes/Bulk	Average number of bytes bulk rate in the forward direction
58	Fwd Avg Packets/Bulk	Average number of packets bulk rate in the forward direction
59	Fwd Avg Bulk Rate	Average bulk rate in the forward direction

Table A.1 continued from previous page

No	Feature Name	Feature Description
60	Bwd Avg Bytes/Bulk	Average number of bytes bulk rate in the backward direction
61	Bwd Avg Packets/Bulk	Average number of packets bulk rate in the backward direction
62	Bwd Avg Bulk Rate	Average number of bulk rate in the backward direction
63	Subflow Fwd Packets	The average number of packets in a sub flow in the forward direction
64	Subflow Fwd Bytes	The average number of bytes in a sub flow in the forward direction
65	Subflow Bwd Packets	The average number of packets in a sub flow in the backward direction
66	Subflow Bwd Bytes	The average number of bytes in a sub flow in the backward direction
67	Init Win bytes forward	The total number of bytes in a sub flow in the forward direction
68	Init Win bytes backward	The total number of bytes in a sub flow in the backward direction
69	act data pkt fwd	Count of packets with at least 1 byte of TCP data payload in the forward direction
70	min seg size forward	Minimum segment size observed in the forward direction
71	Active Mean	Mean time a flow was active before becoming idle
72	Active Std	Standard deviation time a flow was active before becoming idle
73	Active Max	Maximum time a flow was active before becoming idle

Table A.1 continued from previous page

No	Feature Name	Feature Description
74	Active Min	Minimum time a flow was active before becoming idle
75	Idle Mean	Mean time a flow was idle before becoming active
76	Idle Std	Standard deviation time a flow was idle before becoming active
77	Idle Max	Maximum time a flow was idle before becoming active
78	Idle Min	Minimum time a flow was idle before becoming active
79	Label	Label

APPENDIX B. RESULTS OF KNN-BASED BATCH-INCREMENTAL LEARNING IN ONLINE SETTING

This appendix contains the results of KNN-based batch-incremental SVM learning with the processed format (comma-separated values (CSV) formatted files) of the CICIDS2017 dataset. The table B.1 provides the results every 20 iterations.

Table B.1.: Results of the KNN-based ISVM for batch-incremental learning in online setting

Iteration #		Benign	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS Slowloris	Training duration	Test duration	# of CSV
1	Precision	0.97	0.94	0.95	0.91	0.00	0.59	0.325	13.993	326
	Recall	0.98	0.90	0.30	0.85	0.00	0.38			
	F1-score	0.98	0.92	0.46	0.88	0.00	0.46			
20	Precision	1.00	1.00	0.89	0.96	0.97	0.91	1.803	74.241	1737
	Recall	0.99	0.99	0.84	0.99	0.84	0.90			
	F1-score	0.99	0.99	0.86	0.98	0.90	0.90			
40	Precision	1.00	0.99	0.93	0.96	0.97	0.95	4.749	109.429	2520
	Recall	0.99	1.00	0.94	1.00	0.9	0.9			
	F1-score	0.99	0.99	0.93	0.98	0.93	0.93			
60	Precision	1.00	1.00	0.93	0.97	0.97	0.95	8.013	135.891	3126
	Recall	0.99	1.00	0.96	0.99	0.9	0.95			
	F1-score	0.99	1.00	0.94	0.98	0.93	0.95			
80	Precision	1.00	1.00	0.87	0.97	0.97	0.97	7.669	164.168	3640
	Recall	0.99	1.00	0.93	0.99	0.92	0.96			
	F1-score	0.99	1.00	0.9	0.98	0.95	0.96			
100	Precision	1.00	1.00	0.88	0.97	0.98	0.97	10.479	195.254	4175
	Recall	0.99	1.00	0.94	0.99	0.94	0.96			
	F1-score	0.99	1.00	0.91	0.98	0.96	0.97			
120	Precision	1.00	1.00	0.9	0.97	0.98	0.98	8.254	193.868	4266
	Recall	0.99	1.00	0.94	1.00	0.95	0.96			
	F1-score	0.99	1.00	0.92	0.98	0.96	0.97			
140	Precision	1.00	0.99	0.93	0.97	0.97	0.98	9.794	207.150	4627
	Recall	0.99	1.00	0.95	1.00	0.95	0.96			
	F1-score	0.99	1.00	0.94	0.98	0.96	0.97			
160	Precision	1.00	0.99	0.94	0.97	0.97	0.98	16.448	224.569	5087
	Recall	0.99	1.00	0.96	1.00	0.95	0.96			

Table B.1 continued from previous page

Iteration #		Benign	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS Slowloris	Training duration	Test duration	# of CSV
	F1-score	0.99	1.00	0.95	0.98	0.96	0.97			
	Precision	1.00	0.99	0.95	0.97	0.97	0.97			
180	Recall	0.99	1.00	0.97	1.00	0.95	0.96	12.524	238.239	5520
	F1-score	0.99	1.00	0.96	0.98	0.96	0.97			
	Precision	1.00	1.00	0.95	0.97	0.97	0.98			
200	Recall	0.99	1.00	0.97	1.00	0.96	0.96	16.674	255.715	5873
	F1-score	0.99	1.00	0.96	0.98	0.97	0.97			
	Precision	1.00	1.00	0.94	0.97	0.98	0.98			
220	Recall	0.99	1.00	0.98	1.00	0.97	0.96	14.710	287.187	6268
	F1-score	0.99	1.00	0.96	0.98	0.97	0.97			
	Precision	1.00	0.99	0.91	0.97	0.98	0.98			
240	Recall	0.99	1.00	0.95	1.00	0.97	0.96	24.250	327.463	6766
	F1-score	0.99	1.00	0.93	0.98	0.97	0.97			
	Precision	1.00	0.99	0.92	0.97	0.98	0.98			
260	Recall	0.99	1.00	0.95	1.00	0.97	0.96	31.967	356.783	7241
	F1-score	0.99	1.00	0.94	0.98	0.97	0.97			
	Precision	1.00	1.00	0.94	0.97	0.98	0.98			
280	Recall	0.99	1.00	0.96	1.00	0.97	0.97	35.661	340.443	7272
	F1-score	0.99	1.00	0.95	0.98	0.97	0.97			
	Precision	1.00	1.00	0.94	0.97	0.98	0.99			
300	Recall	0.99	1.00	0.96	1.00	0.97	0.98	49.584	360.362	7630
	F1-score	0.99	1.00	0.95	0.98	0.97	0.98			
	Precision	1.00	1.00	0.95	0.97	0.98	0.99			
320	Recall	0.99	1.00	0.97	1.00	0.97	0.98	28.850	370.380	7885
	F1-score	0.99	1.00	0.96	0.98	0.97	0.98			
	Precision	1.00	1.00	0.92	0.97	0.98	0.99			
340	Recall	0.99	1.00	0.97	1.00	0.97	0.99	55.112	388.698	8270
	F1-score	0.99	1.00	0.94	0.98	0.97	0.99			
	Precision	1.00	0.98	0.95	0.97	0.97	0.98			
360	Recall	0.99	1.00	0.97	1.00	0.97	0.99	32.613	401.507	8489
	F1-score	0.99	0.99	0.96	0.98	0.97	0.99			
	Precision	1.00	0.99	0.94	0.97	0.97	0.98			
380	Recall	0.99	1.00	0.97	1.00	0.97	0.99	62.346	418.212	8841
	F1-score	0.99	1.00	0.96	0.98	0.97	0.99			
	Precision	1.00	1.00	0.9	0.97	0.96	0.99			
400	Recall	0.99	1.00	0.96	1.00	0.97	0.98	28.749	455.800	9130

Table B.1 continued from previous page

Iteration #		Benign	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS Slowloris	Training duration	Test duration	# of CSV
	F1-score	0.99	1.00	0.93	0.98	0.97	0.98			
	Precision	1.00	1.00	0.93	0.97	0.97	0.99			
420	Recall	0.99	1.00	0.96	1.00	0.97	0.98	33.377	414.058	9361
	F1-score	0.99	1.00	0.95	0.98	0.97	0.98			
	Precision	1.00	1.00	0.93	0.97	0.97	0.99			
440	Recall	0.99	1.00	0.97	1.00	0.97	0.99	37.242	425.482	9627
	F1-score	0.99	1.00	0.95	0.98	0.97	0.99			
	Precision	1.00	1.00	0.94	0.97	0.97	0.99			
460	Recall	0.99	1.00	0.97	1.00	0.97	0.99	27.373	473.007	9889
	F1-score	0.99	1.00	0.96	0.98	0.97	0.99			
	Precision	1.00	1.00	0.95	0.97	0.97	0.99			
480	Recall	0.99	1.00	0.97	1.00	0.97	0.99	37.560	478.036	10227
	F1-score	0.99	1.00	0.96	0.98	0.97	0.99			
	Precision	1.00	1.00	0.95	0.97	0.97	0.99			
500	Recall	0.99	1.00	0.97	1.00	0.97	0.99	43.751	487.416	10518
	F1-score	0.99	1.00	0.96	0.98	0.97	0.99			
	Precision	1.00	1.00	0.95	0.97	0.98	0.99			
520	Recall	0.99	1.00	0.97	1.00	0.97	0.99	31.948	519.156	10818
	F1-score	0.99	1.00	0.96	0.98	0.98	0.99			
	Precision	1.00	1.00	0.95	0.97	0.98	0.97			
540	Recall	0.99	1.00	0.98	1.00	0.98	0.99	42.947	523.423	11123
	F1-score	0.99	1.00	0.96	0.98	0.98	0.98			
	Precision	1.00	1.00	0.95	0.97	0.98	0.98			
560	Recall	0.99	1.00	0.98	1.00	0.98	0.99	39.237	537.656	11375
	F1-score	0.99	1.00	0.96	0.98	0.98	0.98			
	Precision	1.00	1.00	0.95	0.97	0.98	0.98			
580	Recall	0.99	1.00	0.98	1.00	0.98	0.99	32.263	562.460	11664
	F1-score	0.99	1.00	0.96	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.97	0.96	0.99			
600	Recall	0.99	1.00	0.98	1.00	0.98	0.99	47.271	576.259	11961
	F1-score	0.99	1.00	0.97	0.98	0.97	0.99			
	Precision	1.00	1.00	0.96	0.96	0.96	0.99			
620	Recall	0.99	1.00	0.98	1.00	0.98	0.99	50.202	588.899	12224
	F1-score	0.99	1.00	0.97	0.98	0.97	0.99			
	Precision	1.00	1.00	0.96	0.96	0.97	0.99			
640	Recall	0.99	1.00	0.98	1.00	0.98	0.99	45.890	602.860	12553

Table B.1 continued from previous page

Iteration #		Benign	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS Slowloris	Training duration	Test duration	# of CSV
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.99			
660	Recall	0.99	1.00	0.98	1.00	0.98	0.99	50.587	566.692	11887
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.97	0.98	0.99			
680	Recall	0.99	1.00	0.98	1.00	0.98	0.99	48.040	575.257	12126
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.97	0.98	0.99			
700	Recall	0.99	1.00	0.98	1.00	0.98	0.99	43.611	586.607	12358
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.99			
720	Recall	0.99	1.00	0.98	1.00	0.98	0.99	39.239	599.729	12625
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.99			
740	Recall	0.99	1.00	0.98	1.00	0.98	0.99	37.957	617.090	12991
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.99			
760	Recall	0.99	1.00	0.98	1.00	0.98	0.99	50.449	631.184	13286
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.97	0.98	0.99			
780	Recall	0.99	1.00	0.98	1.00	0.98	0.99	44.319	647.788	13596
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.97	0.98	0.99			
800	Recall	0.99	1.00	0.98	1.00	0.98	0.99	62.897	660.316	13919
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.99			
820	Recall	0.99	1.00	0.99	1.00	0.98	0.99	61.530	672.218	14076
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.99			
840	Recall	0.99	1.00	0.99	1.00	0.98	0.99	54.823	683.793	14358
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.99			
860	Recall	0.99	1.00	0.99	1.00	0.98	0.99	56.830	696.667	14592
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.97	0.98	0.99			
880	Recall	0.99	1.00	0.99	1.00	0.98	0.99	53.799	720.332	14887

Table B.1 continued from previous page

Iteration #		Benign	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS Slowloris	Training duration	Test duration	# of CSV
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.99			
900	Recall	0.99	1.00	0.99	1.00	0.98	0.99	60.305	735.222	15196
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.99			
920	Recall	0.99	1.00	0.99	1.00	0.98	0.99	65.934	747.336	15420
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.95	0.96	0.98	0.99			
940	Recall	0.99	1.00	0.98	1.00	0.98	0.99	141.994	743.423	15650
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.95	0.96	0.98	0.99			
960	Recall	0.99	1.00	0.99	1.00	0.98	0.99	83.334	743.773	15872
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.95	0.97	0.98	0.99			
980	Recall	0.99	1.00	0.99	0.99	0.98	0.99	80.156	761.687	16030
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.95	0.96	0.98	0.99			
1000	Recall	0.99	1.00	0.99	1.00	0.98	0.99	74.342	772.028	16289
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.95	0.96	0.98	0.99			
1020	Recall	0.99	1.00	0.99	1.00	0.98	0.99	116.787	781.398	16548
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.95	0.96	0.98	0.97			
1040	Recall	0.99	1.00	0.99	1.00	0.98	0.99	81.912	793.797	16738
	F1-score	0.99	1.00	0.97	0.98	0.98	0.98			
	Precision	1.00	1.00	0.96	0.97	0.98	0.98			
1060	Recall	0.99	1.00	0.99	0.99	0.98	0.99	89.723	803.221	16860
	F1-score	0.99	1.00	0.97	0.98	0.98	0.98			
	Precision	1.00	1.00	0.95	0.96	0.98	0.98			
1080	Recall	0.99	1.00	0.99	1.00	0.98	0.99	74.288	819.705	17162
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	0.99	0.95	0.96	0.98	0.98			
1100	Recall	0.99	1.00	0.99	0.99	0.98	0.99	91.834	828.455	17437
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.95	0.96	0.99	0.98			
1120	Recall	0.99	1.00	0.99	0.99	0.98	0.99	82.211	837.691	17637

Table B.1 continued from previous page

Iteration #		Benign	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS Slowloris	Training duration	Test duration	# of CSV
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.95	0.96	0.98	0.99			
1140	Recall	0.99	1.00	0.99	1.00	0.98	0.99	110.192	847.534	17864
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.99			
1160	Recall	0.99	1.00	0.99	1.00	0.98	0.99	92.539	859.288	18109
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.96	0.99	0.99			
1180	Recall	0.99	1.00	0.99	1.00	0.99	0.99	115.101	870.274	18378
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.95	0.96	0.99	0.99			
1200	Recall	0.99	1.00	0.99	1.00	0.99	0.99	92.227	879.842	18572
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.97	0.98	0.99			
1220	Recall	0.99	1.00	0.99	1.00	0.99	0.99	136.61	892.201	18854
	F1-score	1.00	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.97	0.98	0.99			
1240	Recall	0.99	1.00	0.99	0.99	0.99	0.99	174.98	884.281	19032
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.99			
1260	Recall	0.99	1.00	0.99	1.00	0.99	0.99	116.84	904.347	19229
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.97	0.98	0.99			
1280	Recall	0.99	1.00	0.99	1.00	0.99	0.99	115.772	924.498	19435
	F1-score	1.00	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.96	0.99	0.99			
1300	Recall	0.99	1.00	0.99	0.99	0.99	0.99	151.553	935.191	19621
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.96	0.97	0.99			
1320	Recall	0.99	1.00	0.99	1.00	0.99	0.99	115.801	960.641	19828
	F1-score	1.00	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.96	0.97	0.99			
1340	Recall	0.99	1.00	0.99	0.99	0.99	0.99	105.06	949.452	20056
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.95	0.96	0.97	0.99			
1360	Recall	0.99	1.00	0.99	1.00	0.99	0.99	145.879	966.413	20305

Table B.1 continued from previous page

Iteration #		Benign	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS Slowloris	Training duration	Test duration	# of CSV
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.95	0.96	0.97	0.99			
1380	Recall	0.99	1.00	0.99	0.99	0.99	0.99	127.485	988.044	20483
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.95	0.97	0.97	0.98			
1400	Recall	0.99	1.00	0.99	1.00	0.99	0.99	124.540	983.854	20728
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	0.98	0.96	0.96	0.97	0.98			
1420	Recall	0.99	1.00	0.99	1.00	0.99	0.99	151.605	994.601	20960
	F1-score	0.99	0.99	0.97	0.98	0.98	0.99			
	Precision	1.00	0.99	0.95	0.97	0.98	0.98			
1440	Recall	0.99	1.00	0.99	1.00	0.99	0.99	138.228	974.331	21103
	F1-score	0.99	0.99	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.94	0.96	0.97	0.98			
1460	Recall	0.99	1.00	0.99	1.00	0.99	0.99	201.759	1025.967	21353
	F1-score	0.99	1.00	0.96	0.98	0.98	0.99			
	Precision	1.00	1.00	0.95	0.96	0.97	0.98			
1480	Recall	0.99	1.00	0.99	1.00	0.99	0.99	131.080	1036.890	21593
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.95	0.96	0.98	0.98			
1500	Recall	0.99	1.00	0.99	1.00	0.99	0.99	152.899	2960.209	21827
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.98			
1520	Recall	0.99	1.00	0.99	1.00	0.99	0.99	110.901	1063.902	22017
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.98			
1540	Recall	0.99	1.00	0.99	1.00	0.99	1.00	195.105	1051.175	22219
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	0.99	0.96	0.96	0.98	0.98			
1560	Recall	0.99	1.00	0.99	1.00	0.99	0.99	158.998	1208.143	22446
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	0.99	0.96	0.96	0.98	0.98			
1580	Recall	0.99	1.00	0.99	1.00	0.99	1.00	126.167	1114.680	22689
	F1-score	0.99	0.99	0.97	0.98	0.99	0.99			
	Precision	1.00	0.99	0.96	0.96	0.98	0.98			
1600	Recall	0.99	1.00	0.99	1.00	0.99	1.00	222.067	1122.152	22904

Table B.1 continued from previous page

Iteration #		Benign	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS Slowloris	Training duration	Test duration	# of CSV
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.96	0.99	0.98			
1620	Recall	0.99	1.00	0.99	1.00	0.99	0.99	148.979	1133.438	23103
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.92			
1640	Recall	0.99	1.00	0.99	1.00	0.99	0.99	189.695	1105.470	23346
	F1-score	0.99	1.00	0.97	0.98	0.98	0.96			
	Precision	1.00	1.00	0.96	0.96	0.97	0.99			
1660	Recall	0.99	1.00	0.99	1.00	0.99	0.99	148.752	1117.989	23519
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.95	0.96	0.98	0.98			
1680	Recall	0.99	1.00	0.99	1.00	0.99	0.99	180.674	1078.514	23657
	F1-score	0.99	1.00	0.97	0.98	0.98	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.98			
1700	Recall	0.99	1.00	0.99	1.00	0.99	0.99	190.200	1144.961	23914
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.97	0.99	0.98			
1720	Recall	0.99	1.00	0.99	1.00	0.99	0.99	159.252	1159.872	24108
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.96	0.99	0.98			
1740	Recall	0.99	1.00	0.99	1.00	0.99	0.99	148.205	1175.930	24327
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.95	0.97	0.99	0.98			
1760	Recall	0.99	1.00	0.99	1.00	0.99	0.99	181.271	1159.877	24652
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.98			
1780	Recall	0.99	1.00	0.99	1.00	0.99	0.99	251.341	1178.601	24779
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.95	0.96	0.98	0.98			
1800	Recall	0.99	1.00	0.99	1.00	0.99	1.00	190.334	1194.716	25136
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.98			
1820	Recall	0.99	1.00	0.99	1.00	0.99	1.00	178.112	1212.173	25349
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.96	0.99	0.98			
1840	Recall	0.99	1.00	0.99	1.00	0.99	1.00	228.736	1221.519	25553

Table B.1 continued from previous page

Iteration #		Benign	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS Slowloris	Training duration	Test duration	# of CSV
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.98			
1860	Recall	0.99	1.00	0.99	1.00	0.99	1.00	154.793	1230.658	25710
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.99			
1880	Recall	0.99	1.00	0.99	1.00	0.99	1.00	241.381	1264.586	25929
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.97	0.98	0.98			
1900	Recall	0.99	1.00	0.99	1.00	0.99	1.00	182.946	1234.206	26173
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.97	0.98	0.98			
1920	Recall	0.99	1.00	0.99	1.00	0.99	1.00	178.112	1324.234	26342
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.96	0.98	0.99			
1940	Recall	0.99	1.00	0.99	1.00	0.99	1.00	228.736	1356.234	26543
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.97	0.98	0.98			
1960	Recall	0.99	1.00	0.99	1.00	0.99	1.00	154.793	1382.432	26453
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.97	0.98	0.98			
1980	Recall	0.99	1.00	0.99	1.00	0.99	1.00	241.381	1402.234	27352
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			
	Precision	1.00	1.00	0.96	0.97	0.98	0.98			
2000	Recall	0.99	1.00	0.99	1.00	0.99	1.00	182.946	1432.434	27832
	F1-score	0.99	1.00	0.97	0.98	0.99	0.99			