Received June 18, 2021, accepted June 26, 2021, date of publication July 1, 2021, date of current version July 9, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3093830

Building an Intrusion Detection System to Detect Atypical Cyberattack Flows

ULYA SABEEL[1], **SHAHRAM SHAH HEYDARI**[1], **(Senior Member, IEEE), KHALID ELGAZZAR**[2], **(Senior Member, IEEE), AND KHALIL EL-KHATIB**[1], **(Member, IEEE)**

[1]Faculty of Business and IT, University of Ontario Institute of Technology, Oshawa, ON L1G 0C5, Canada
[2]Faculty of Engineering and Applied Science, University of Ontario Institute of Technology, Oshawa, ON L1G 0C5, Canada

Corresponding author: Ulya Sabeel (ulya.sabeel@ontariotechu.net)

This work was supported by Natural Sciences and Engineering Research Council of Canada (NSERC) Engage Grant.

ABSTRACT Artificial Intelligence (AI) techniques provide effective solutions for the detection of many aberrant network traffic patterns and attack flows. However, the validation of these techniques often relies on one training dataset. Recent results show that such training may fail in the face of dynamically-changing cyberattacks. Given the increased sophistication of cyberattacks nowadays, it is imperative to examine and improve the performance of such AI models. This paper proposes a defensive AI engine combined with a twofold feature selection technique and hyperparameter optimization of the AI model. In this work, we utilize the proposed system for binary attack flow identification and the AI models are trained and validated on the CICIDS2017 dataset. The system is then evaluated using synthesized atypical attack flows to mimic real-world scenarios. We demonstrate the effectiveness of the proposed atypical attack flow detection approach using several Deep Learning and Machine Learning models including DNN, Linear-SVC, and Stacked Decision Tree Classifier (S-DTC). Simulation results demonstrate that the proposed defensive AI engine significantly improves the True Positive Rate (TPR) of AI models on multiple atypical attacks.

INDEX TERMS Artificial Intelligence (AI), atypical attacks, Denial of Service, feature profile, intrusion detection.

I. INTRODUCTION

Artificial Intelligence provides powerful solutions for improving cybersecurity in general, and intrusion detection systems [1], [2] in particular due to improvement in attack detection rates, precision, and recall as well as the reduction in false positives and negatives. Such techniques have also been applied successfully in network analytics to analyze complex encrypted traffic flows [3], [4]. These AI models can learn sequential features from raw network traffic flows as well as classify them as attack or benign. AI techniques have been typically applied to network intrusion detection by training Deep Learning (DL) or Machine Learning (ML) models on a benchmark dataset and validating them on a split of the observed data to detect similar future attack patterns. The main advantage of AI models is their ability to learn the representative features of an object and create a general perception of the target object for future sample predictions. For example, a DL model that is trained on a sufficiently large database of bird pictures can correctly identify a picture of an unknown bird (different from the ones already present in training data) based on the generic features learned during training. In the case of network intrusion detection, an AI model should be able to inspect the network traffic and detect any attack-related abnormal patterns such as changes in the data rate and inter-arrival times, or incomplete requests from specific network addresses. Training such models requires labeled data or synthetically generated datasets for binary or multi-class identification of non-malicious and malicious patterns.

The main predicament in AI-based network intrusion detection is that attack patterns can differ substantially in each case. Especially when attackers dynamically mutate multiple features to change the attack profile such as port numbers and data rates using sophisticated tools to evade detection. The majority of AI research in network security

The associate editor coordinating the review of this manuscript and approving it for publication was Pedro R. M. Inácio.

typically uses a portion of benchmark training data to evaluate the model performance. For example, a dataset is divided into three parts-70% for training, 20% for validation, and 10% for testing or evaluating the model. Such models are not evaluated or tested on any external synthesized datasets [5]–[7]. The fundamental challenge with this approach is that the data usually represents only specific attack scenarios on specific network settings. Consequently, the training and testing data subsets may exhibit the same characteristics. As such, these datasets may share certain tool-dependent characteristics such as packet size distributions, port numbers, bandwidth, and data rates. Since the training and testing data have similar characteristics, such models often achieve high detection rates. This does not necessarily show the equal success of such a model in real-world scenarios when exposed to attacks of the same nature but with different parameters. While adaptive re-training of the observed model may improve the detection rate, it will not reduce its vulnerability to rapidly evolving attack parameters.

In this paper, we propose a highly efficient generic defensive AI engine that detects atypical attack flows with a novel feature selection and training algorithm. The focus of this work is on identifying DoS/DDoS attacks through flow analysis using the proposed defensive AI engine. We chose flow-based identification over packet-based identification since it contains aggregated information about the entire network [8]. Additionally, it is easier and faster to identify an attacker in high bandwidth networks using this technique [8]. Although we apply the proposed technique to identify only DoS/DDoS attack flows, it is equally applicable to identify any type of network attack.

The main contributions of this research are as follows:

- We employ a two-phase feature selection technique to select the best features to improve the Intrusion Detection System (IDS) training. The first phase is the pre-screening of features based on the knowledge and understanding of the attack characteristics. The second phase is passing a refined feature subset through an ensemble of feature selectors to generate the final feature subset. The advantages of this approach are discussed in more detail in the methodology section.
- We test several AI-based IDS models on the synthesized atypical attack flows to improve generalization and ensure an unbiased evaluation of these models. The models run through a hyperparameter optimization process if their performance is below a predefined threshold. Pre-trained and optimized models are then reevaluated on atypical attack flows until satisfactory performance is achieved.

The remainder of the paper is organized as follows. Section III provides detailed information about the related work. Section II discusses the concept of feature profile and elaborately explains an atypical attack in our context. Section IV provides insights on background techniques applied for this research. Section V describes our methodology in great detail. Section VI describes the experimental setup and performance evaluation. Section VII discusses the hyperparameter optimization process and provides a comparative analysis of AI models on different atypical attacks. Lastly, Section VIII concludes the paper, discusses several challenges identified during this research, and outlines the scope for future work.

II. ATYPICAL ATTACK

In our context, the term *feature profile* for an attack is defined as the range of values of features that constitute an attack [9]. For example, the feature profile of a TCP SYN DoS attack may include certain port numbers, a certain number of half-open connections, a certain number of set SYN flags, and a specific data rate. Attacks of different feature profiles can be generated by tweaking the feature range values randomly to perform a successful attack. We define an *atypical attack* as a cyberattack with similar features like the typical attacks in the training dataset but having a different feature profile. In other words, such attacks have a different range of values for features as compared to the ones the IDS has been previously trained on during the learning process.

An atypical attack scenario in our case is shown in Fig. 1. The attacker launches an attack on the target network using sophisticated attack tools. The target network is equipped with a Network Intrusion Detection System (NIDS) to identify the attack. If the attack is identified, the attacker is blocked by the target network. This attacker later changes the feature profile (range of values of features) of the attack by mutating certain parameters to launch a new atypical attack. For example, *F1* is changed to *F1'* and *F3* is changed to *F3'* as shown in the figure. The goal of this attacker is to attack the network in such a way that it can evade detection from the NIDS by launching attack flows as close to benign data flows as possible. The attacker will keep on launching atypical attacks until it is successful in evading detection by the target network or until it runs out of features.

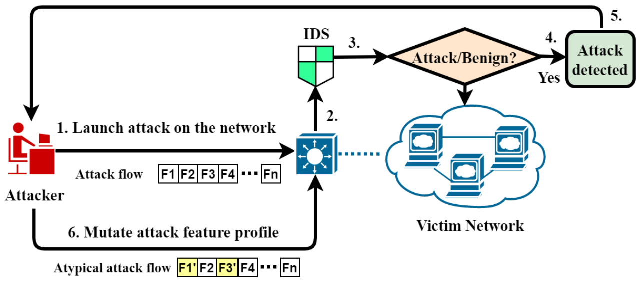


FIGURE 1. An atypical attack scenario.

Our recent research has also shown that current AI-based attack detection techniques often fail to detect such atypical attacks [9]. These attacks can pose a great challenge for our cyber environment if we rely on traditional security techniques. Therefore, an improved security system to combat such attacks is the need of the hour.

III. RELATED WORK

AI-based models for intrusion detection offer many benefits in forging insights into different cyberattacks. These techniques are broadly classified as *static* and *dynamic*. Static models are also known as offline models because they are trained in a stationary environment. While dynamic or online models are trained in a continuously changing or real-world environment.

A. STATIC LEARNING IN INTRUSION DETECTION

Static AI models are used where data is not continuously changing. Models are trained once with benchmark data and then are deployed for static network attack detection. A considerable number of AI-based research studies have been conducted for static or offline network attack detection using both ML and DL models.

1) MACHINE LEARNING

Machine learning has been an active and hot research topic for network attack detection. This section provides a brief overview of different ML algorithms and shows how current researchers use benchmark datasets to train and evaluate their models offline. Syarif and Gata [10] have used a combination of Binary Particle Swarm Optimization (PSO) and K-Nearest Neighbour (KNN) for network intrusion detection. Shah *et al.* [11] propose a network attack detection technique based on Sparse Logistic Regression. Sparse Logistic Regression in their case is used for feature selection and attack classification. The mini batch K-means clustering approach has been also effectively used in intrusion detection systems [12]. The algorithm is compared with traditional K-means based on clustering time and the Calsski Harabasz (CH) indicator to improve the performance. However, no comparison has been conducted based on the accuracy or detection rate for attacks, only clustering time is compared. The research techniques mentioned above rely on the same benchmark dataset both for training and evaluation of their models and these models typically fail to generalize in detecting attacks of a similar nature but with a different feature profile [9].

Aksu *et al.* [13] compare the performance of SVM, KNN, and DT algorithms for attack detection. The authors use the Fisher Scoring technique based on the maximum likelihood function for feature selection and report improvement in the accuracy of decision tree classifiers afterward. Ahmim *et al.* [14] propose a hybrid attack detection system based on DT and Rule-Based models. The authors claim that their technique outperforms the state of the art. However, the model fails to generalize to capture atypical attacks. Abdulhammed *et al.* [15] apply Principal Component Analysis (PCA) for feature reduction on various defensive ML models such as Random Forest, Bayesian Network, Linear Discriminant Analysis, and Quadratic Discriminant Analysis. Although they report improved accuracy using their technique, no evaluation of their trained models on atypical attacks is reported.

2) DEEP LEARNING

Deep learning has also been a powerful tool for intrusion detection in computer networks. Kim *et al.* [1] use Long Short Term Memory (LSTM) for network intrusion detection. The authors focus more on improving the attack detection rate while neglecting the benign detection rate which makes their predictions have high false positives. This means the benign packets in their case are classified as attacks. Ma *et al.* [16] propose an ensemble IDS based on spectral clustering and DNN for sensor networks. While the precision rates on DoS and probe attacks are high, their work does not focus on improving detection rates for atypical attacks of different feature profiles. Wang *et al.* [17] propose an attack detection technique based on the knowledge gained from low-level spatial-temporal features using CNN and high-level spatiotemporal features using LSTM. The authors use the same benchmark dataset for both training and validation, achieving high accuracy.

Vinayakumar *et al.* [18] propose a framework using a DNN model for identifying attacks. Although the authors conduct extensive experiments with different benchmark datasets and report high accuracies in most cases, their models have not been evaluated using attacks that are even slightly different from the training dataset. Some other research studies have been conducted recently using CNN [19], [20] and LSTM [21], [22] to identify typical attacks with high accuracy. While these techniques present favorable results for the detection of attack flows using benchmark data, no testing is provided on atypical attacks.

Many researchers employ unsupervised one-class anomaly detection techniques to identify attacks in the network. In this case, the AI-based IDS model is only trained on benign observations whereas the rare attack observations in the evaluation data are treated as anomalies. Bovenzi *et al.* [23] propose a hierarchical MultiModal Deep Autoencoder model to identify anomalies in an IoT network. Although the authors report very low FAR (≤ 1), this evaluation is provided only for a maximum of 25 packets. Many anomaly detection-based research works report improved accuracy [24], [25], but the main problem with these models is their high false alarm rate [26]. Such intrusion detection models may not be able to generalize effectively on an unknown observation and may classify it as an attack even though it is an unknown benign observation [27]. Furthermore, in a dynamically changing attack scenario, an attacker finds ways to mutate the feature profile and generate attack distributions that are close to benign. In such situations, the unsupervised anomaly detection models may fail to correctly identify attack flows thus increasing the false negative rate. We provide a comparison of current static learning-based approaches in Table 1.

B. CONTINUOUS/ DYNAMIC LEARNING IN INTRUSION DETECTION

In real-world scenarios, attackers can mutate the attack feature profile quite often, thus making it difficult for AI-based IDS systems to identify such newly evolved attacks.

TABLE 1. Comparison between current static learning based approaches for intrusion detection. Here TA represents typical attack (hold-out dataset from benchmark data), AA represents an atypical attack, FS represents feature selection, HPO represents hyperparameter optimization, and N/A represents "not available".

AI Approach	Main Idea	Dataset	FS	HPO	Accuracy - TA	Evaluation - AA or Anomalies
Binary PSO, KNN [10]	Hybrid Algorithm for feature selection using Binary PSO and Intrusion Detection using KNN.	KDD'99	Yes	N/A	98.46%	No
Sparse Logistic Regression (SPLR) [11]	Intrusion Detection using discriminative feature selection and SPLR.	KDD'99	Yes	N/A	96.51%	No
Mini Batch K-means [12]	Intrusion Detection using Mini Batch K-means and dimension reduction using Principal Component Analysis (PCA).	KDD'99	Yes	N/A	N/A	No
SVM, KNN, DT [13]	Feature selection using Fisher Score algorithm and performance comparison of SVM, DNN and DT for intrusion detection.	CICIDS2017	Yes	N/A	SVM-57.57% KNN-99.97% DT-99%	No
Reduced Error Pruning (REP) Tree, JRip, Forest PA [14]	A hybrid Intrusion Detection System using multiple tree-based classifiers based on WEKA.	CICIDS2017	No	N/A	96.66%	No
Autoencoder (AE), PCA [15]	Employ AE and PCA for dimension reduction, then design an efficient intrusion detection system using multiple AI models such as Random Forest (RF), Bayesian Network (BN), Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA).	CICIDS2017	Yes	N/A	PCA-RF 98.8% PCA-BN 97.6% PCA-LDA 95.7% PCA-QDA 98.9%	No
LSTM [1]	A Network Intrusion Detection System based on LSTM.	KDD'99	No	Yes	96.93%	No
Spectral Clustering and DNN [16]	A hybrid ensemble Intrusion Detection System based on DNN and Spectral Clustering for sensor networks. Spectral clustering is employed to generate feature clusters. The datasets are divided into multiple data subsets for performance evaluation.	KDD'99 NSL-KDD	Yes	N/A	Data1-91.97% Data2-92.03% Data3-92.1% Data4-72.64% Data5-71.83% Data6-44.55%	No
CNN, LSTM [17]	A hybrid network for Intrusion Detection using CNN-LSTM. CNN is used to identify the spatial features whereas LSTM identifies temporal features.	KDD'99 ISCX2012	Yes	N/A	KDD'99- 99.68% ISCX2012-99.69%	No
DNN [18]	A Scalable Hybrid Intrusion Detection System based on DNN. This research work employs training and evaluation of the IDS using multiple datasets to identify attacks.	KDD'99 NSL-KDD UNSW-NB15 Kyoto WSN-DS CICIDS2017	Yes	Yes	For DNN 5 layers: KDD'99-92.7% NSL-KDD-78.9 UNSW-NB15-76.1 Kyoto-88.5%, WSN-DS 98.2% CICIDS2017-93.1%	No
CNN, LSTM [19]	A DDoS Attack detection system using a hybrid model consisting of CNN and LSTM for IoT Networks. A multi-objective optimization technique is applied for dimension reduction.	CICIDS2017	Yes	N/A	99.03%	No
CNN, CNN-LSTM [20]	A Network Intrusion Detection System based on a one-dimensional CNN model. For resolving the class imbalance problem, random sampling technique is used to balance attack and benign observations.	UNSW-NB15	Yes	Yes	1D-CNN 3 Layers-91.20%	No
LSTM [21]	A Network attack detection system based on LSTM by applying fine-tuning of hyperparameters.	CICIDS2017	No	Yes	LSTM with 5 layers-99.08%	No
LSTM [22]	Identifying DDoS Attacks using LSTM Model.	CICIDS2017	No	N/A	96.25%	No
MultiModal Deep Autoencoder [23]	An anomaly detection system based on a hierarchical Deep Autoencoder (DAE).	Bot-IoT dataset	Yes	N/A	-	FAR <= 1%
Variational Autoencoder (VAE) [24]	Flow-based unsupervised anomaly detection to identify unknown intrusions in the network.	CICIDS2017	No	Yes	-	AUC-75.96% FAR: N/A
AE, IF [25]	A two-stage anomaly detection System for Fog Computing based on Autoencoder and Isolation Forest (IF). The outputs of stage1 (AE) are fed as inputs to stage2 (IF) to identify any anomalies missed in stage1.	NSL-KDD	No	N/A	-	Accuracy 95.4% FAR: N/A

These systems need continuous or dynamic updates with incoming packet streams to keep up to date. This dynamic learning technique for such AI-based IDS models is also referred to as *incremental learning* or *online learning* or *adaptive learning*. Extensive research has been done in this area to tackle the outstanding challenge.

TABLE 2. Comparison between current dynamic learning based approaches for intrusion detection. Here TA represents typical attack (hold-out dataset from benchmark data), AA represents an atypical attack, FS represents feature selection, HPO represents hyperparameter optimization, and N/A represents "not available".

AI Approach	Main Idea	Dataset	FS	HPO	Accuracy - TA	Evaluation - AA or Anomalies
Autoencoder (AE) Ensemble [28]	Anomaly detection for a local IoT network using unsupervised AE ensemble. The features are extracted dynamically from network traffic.	IoT Dataset	Yes	N/A	-	TPR at FPR=0.1%, m=1: ARP-0.004% Fuzzing-100% Mirai-99.7% OS Scan-0.98% SSDP F-99.9% SSL R-2.51% SYN DoS-24.6% Video Inj-1.06% Wiretap-0.046%
MVO, ANN [29]	An Intrusion Detection System based on Multi-verse Optimizer (MVO) evolutionary algorithm. MVO is applied to optimize the Artificial Neural Network (ANN) for attack identification.	NSL-KDD: UNSW-NB15	No	N/A	NSL-KDD 98.21% UNSW-NB15: 99.61%	No
SADE-ELM [30]	Intrusion Detection based on Self-Adaptive Differential Evolutionary Extreme Machine Learning algorithm. The evolutionary algorithm is applied to optimize the IDS.	KDD'99	No	N/A	99.35%	No
I-ELM [31]	Intrusion Detection System based on incremental extreme learning based approach. The adaptive PCA is applied for feature selection based on the best accuracy metric.	NSL-KDD UNSW-NB15	Yes	N/A	NSL-KDD: 81.22% UNSW-NB15: 70.51%	No
Ensemble Incremental learning [32]	An Intrusion Detection System based on concept drift and ensemble incremental learning. Concept drift measures data variance over time. The ensemble consists of two classifiers (primary and backup) to yield stable detection results.	KDD'99	No	N/A	94.91%	No

Mirsky *et al.* [28] propose an online unsupervised network intrusion detection system using an autoencoder ensemble for a local IoT network. At a FAR threshold of 0.1%, the IDS can classify only three attack classes out of a total of 6 with a TPR of above 99%. For the other three attacks, the model performs poorly. Benmessahel *et al.* [29] present an enhanced neural network that uses Evolutionary Multi-verse Optimization for selecting a set of hyperparameters for its performance improvement. The authors, however, have not performed any extensive feature selection in this case. In Table 2 we highlight some of the current research work in this field.

Multiple researchers use a variation of Extreme Machine Learning (ELM) to detect network attacks. For instance, Ku *et al.* [30] use Self-Adaptive Differential Evolutionary Extreme Machine Learning (SADE-ELM) algorithm for the optimization of neural network node parameters. Gao *et al.* [31] use an incremental extreme learning machine (I-ELM) to detect attacks in the network with Adaptive PCA for feature reduction. Although all these approaches report a good performance for known attacks, yet none of them evaluate their models for atypical attack scenarios. Yuan *et al.* [32] propose a concept drift-based incremental learning approach for attack identification. Concept drift measures changes in the statistical properties of the data over time. Their idea is to use a primary and secondary classifier for detection. When the primary classifier is used for attack detection, the secondary classifier updates itself with new incoming data and then both classifiers swap their roles for continuous learning. Although authors report good performance for their system, no latency measurement is reported while retraining and switching the classifiers.

It is evident from the current research works using both static and continuous learning that the training data and test data are subsets of a single benchmark dataset and share the same probabilistic distribution and feature profile. ML models trained on this data do not generalize well for the real-life detection of evolved or atypical attack patterns. Hence, such models may not be successful in detecting atypical attacks [9]. In the real world, rapidly changing attacks pose severe threats to networks since their feature profile differs from the ones with which the AI models are trained. This, in turn, decreases the attack detection rates of such systems. The main objective of this research is to close this research gap by examining and improving the detection of AI-based models on slightly evolved synthesized attacks with different feature profiles. For this purpose, we incorporate our proposed extensive feature selection technique along with hyperparameter optimization to improve AI model performance and generalization on test data. The proposed approach outperforms the state-of-the-art techniques in the literature and demonstrates highly accurate results.

IV. BACKGROUND AI TECHNIQUES

This section gives detailed background information about various feature selection and hyperparameter optimization (HPO) techniques applied in our methodology.

A. FEATURE SELECTION METHODS

Feature selection is the process of selecting the best group of attributes from a dataset without mutating them [33]. It is an important component of ML/DL techniques as it reduces the computational complexity, makes interpretations easy, and avoids overfitting [34]. This is achieved by reducing the loss while training a model for a specific target using certain mathematical functions [35] as we outline in the following.

1) FILTER METHOD

Features are selected based on their scores in various statistical tests. These scores are created based on the correlation between a continuous or categorical feature and a continuous or categorical target variable. An example of this feature selection method is the Chi2 statistical test. This test measures whether an observed feature is related to the target variable [36]. If the Chi-Square value is high, the target is dependent on that feature and this feature has high importance and vice versa. Since our features are continuous (for example, Inter arrival time and flow duration) and the target variable is categorical (attack or benign), we use the Chi-Square statistical test to select the best features that correlate to the target class.

2) WRAPPER METHOD

The wrapper method identifies feature subsets based on their efficacy for improving the performance of an AI model. However, since the model is trained iteratively until no better performance is achieved, the process is computationally expensive and time-consuming. An example of these methods is Recursive Feature Elimination (RFE), which selects the best features based on the Greedy technique. The best features are selected at each iteration and their ranking is done based on the order of elimination. RFE combines the best attributes of all the wrapper methods and thus we adopt it in this research.

3) EMBEDDED METHOD

In these methods, feature selection is a part of the learning algorithm. At each model iteration, the features that contribute to the best model performance are set aside. These methods combine the best attributes of both filter and wrapper methods. These are adopted for this research since they are efficient, computationally less expensive, and do not require dataset partitioning for training, validation, and model retraining [34].

Linear models such as LR and L-SVC select features by using regularization for overfitting. This causes features that have a weak association with the target class to shrink to zero and get eliminated. Tree-based models such as Random Forest Classifier (RFC) and Extra Tree Classifier (ETC) select features based on the mean decrease in impurity known as Gini index [37]. Ensemble Gradient Boosting models such as Light Gradient Boosting Machine (L-GBM) find the most relevant feature based on some threshold from a local set of features in each tree. The selected feature with the highest score indicates its usefulness in the construction of the tree. This process is repeated until we get the final feature subset [38].

B. HYPERPARAMETER OPTIMIZATION TECHNIQUES

In machine learning, a hyperparameter (HP) is defined as a parameter whose value is set before training to control the learning process such as the network topology and number of layers in neural networks for instance. Hyperparameters also include the learning rate, batch size, maximum depth of the tree, minimum sample split, and the number of estimators for our tree-based models. ML/DL models must be tuned with the most appropriate hyperparameters to make better predictions. In our case, this means minimizing the loss function on continuously evolving attacks in the test data. Finding the best hyperparameters is a challenging and exhaustive task because the model performance is highly sensitive to any small change in the hyperparameter.

Hyperparameter Optimization (HPO) is the global optimization of the computationally expensive error function f_e for the ML/DL model [39]. This function selects a hyperparameter h from the list of H hyperparameters. The selected hyperparameter is then mapped to the validation error of the ML/DL model with L_n learned parameters [39]. The eq.(1) for HPO in general is given as:

$$min_{h \epsilon R^H} f_e(h, L_n; d_{val}) \tag{1}$$

where, L_n equals:

$$argmin\, f_e(h, L_n; d_{train})$$

Here, d_{train} and d_{val} are training and validation data and L_n parameters can be learnt by minimizing the error of the model. Different HPO techniques are given below.

1) MANUAL SEARCH HPO (MS-HPO)

This is a simple trial and error method for the hyperparameter search. Hyperparameters are tweaked manually and the model is retrained again until it achieves the best performance in terms of minimizing the loss function. This technique is extremely complex and highly sub-optimal.

2) GRID SEARCH HPO (GS-HPO)

Grid Search is an exhaustive search of hyperparameters which means it searches for all the possible hyperparameter combinations given by the user. Each different set of hyperparameter combinations is treated as a separate ML/DL model. The outcome is the best-performing model. Grid search suffers from the high dimensionality problem as the number of searches and function evaluations can grow exponentially [40]. The grid search HPO for an algorithm $a \epsilon A$ parametrized with H hyperparameters can be represented by the following eq.(2):

$$h = argmin\, f_o(H; a, d_{train}, d_{val}, f_e) \tag{2}$$

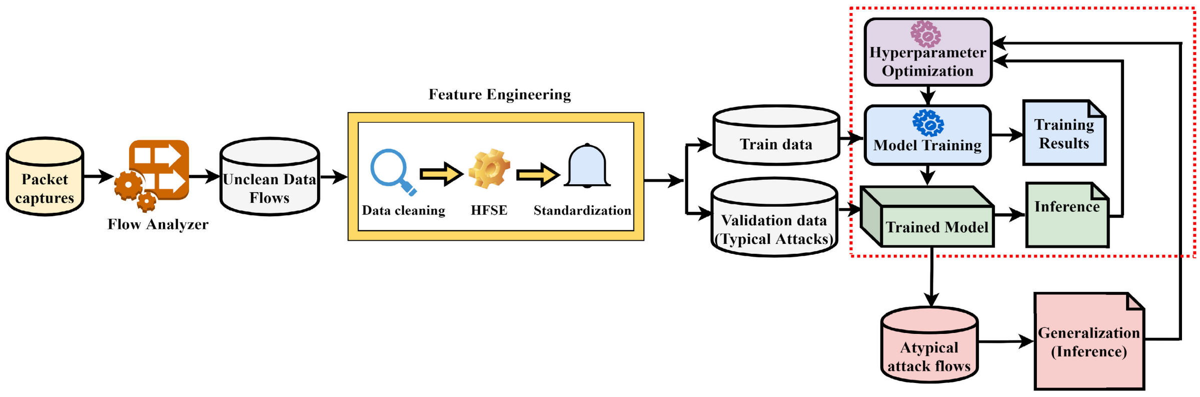


FIGURE 2. **The defensive AI engine for training and generalization of the AI models against atypical attack flows.**

where, $h \epsilon H$ is an improved subset of hyperparameters selected by Grid Search CV used for optimizing model M with an objective function f_o and loss (error) function f_e. The training and validation data are represented by d_{train} and d_{val} respectively.

3) EVOLUTIONARY SEARCH HPO (ES-HPO)

Evolutionary Search HPO can solve optimization problems by mimicking biological evolution. Problems are solved by sequential selection, combination, and mutation of different hyperparameters. Firstly, an initial population of randomly selected hyperparameters is created and ranked based on their fitness function. Like natural selection, the worst-performing hyperparameters are replaced by the new hyperparameters created by mutation and crossover between parents. These new hyperparameters proceed to the next generation. This process is repeated until the performance of the algorithm no longer improves. The population of N members, in this case, can be evaluated in parallel as suggested in [41].

V. METHODOLOGY

Fig. 2 illustrates the proposed defensive AI approach against atypical attacks in computer networks. It consists of three important phases, namely, feature selection, training and generalization, and hyperparameter optimization (HPO).

A. OFFLINE FEATURE SELECTION

In this phase, the raw packet captures are first passed through the flow analyzer which generates data flows that are subjected to cleaning as a standard ML practice. This cleaned benchmark data undergoes the feature selection process which consists of two main phases: manual feature pre-screening and Heterogeneous Feature Selection Ensemble (HFSE). The features, in this case, represent flow-based information about the entire network such as IP addresses, network ports, protocols, connection time duration, arrival times, data/packet size, and some other miscellaneous flags.

1) FEATURE PRE-SCREENING

Most AI systems are black boxes providing limited intuition as to why certain predictions were made. Some explainable algorithms such as Local Interpretable Model-Agnostic Explanations [42] can be used to resolve this issue by explaining the AI predictions based on the selected features, but they do not work efficiently for all types of AI models [43]. When the training data does not fairly reflect important features, the AI algorithm will learn insignificant patterns that do not aid in improving the detection rates, leading to AI bias [43]. Also, AI-based models may pick up a feature that is insignificant from the point of view of a network attack because such models are trained on currently available public datasets that are often created by standard attack generator tools that have a configured range for various feature values. As such, these AI models may pick up a contributing feature simply because the attack generator was configured to use a fixed feature profile. In such cases, an initial manual feature pre-screening identifies the priority features which reflect the vulnerabilities of the target system.

While we do not know the feature profile of the attack in advance, we do have some information on how an attack could potentially affect the target. Manual pre-screening is performed based on the vulnerabilities of the target. For example, the packet size might not be a determining factor in a SYN DoS attack on a web server, so it may be eliminated in the pre-screening process. However, the same factor could be important in a DDoS attack on a networking device, thus it could be included in the features that the defensive AI considers. In other words, pre-screening focuses on the target vulnerabilities that can be determined offline. While in our process the manual pre-screening is done at the start, it can also be revisited over time when the system states change. However, it remains offline and not part of the dynamic real-time AI response. The refined feature set is then passed through the Heterogeneous Feature Selection Ensemble for further filtering. The main advantage of using this approach is to ensure that the AI selects the most significant features that make the results more accurate, reduce AI bias, and improve the performance against atypical attacks.

2) HETEROGENEOUS FEATURE SELECTION ENSEMBLE (HFSE)

Ensemble Feature selection combines the results of multiple models strategically to find the best feature subset. The feature subset selected manually further undergoes filtering using the Heterogeneous Feature Selection Ensemble in an offline mode. Multiple feature selectors as described above are chosen to be a part of this ensemble as the name implies. The feature selectors in this ensemble vote for the best features using a ranking score. The feature that receives the highest votes is considered the best while the feature with the lowest votes is the least significant. Fig. 3 represents our proposed offline feature selection ensemble technique.

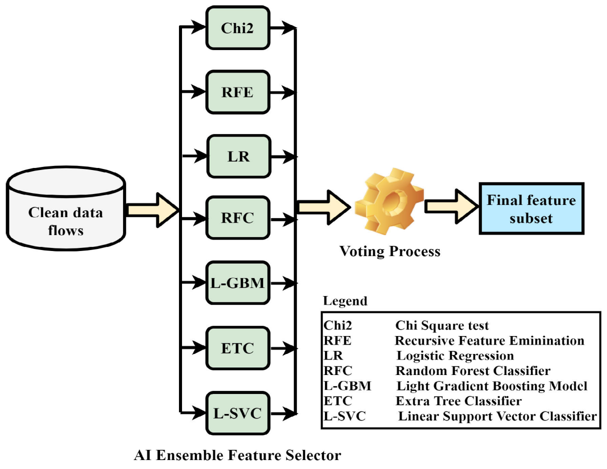

Clean data flows
Chi2
RFE
LR
RFC
L-GBM
ETC
L-SVC
Voting Process
Final feature subset
Legend
Chi2 Chi Square test
RFE Recursive Feature Eminination
LR Logistic Regression
RFC Random Forest Classifier
L-GBM Light Gradient Boosting Model
ETC Extra Tree Classifier
L-SVC Linear Support Vector Classifier
AI Ensemble Feature Selector

FIGURE 3. **Offline Heterogeneous Feature Selection Ensemble (HFSE).**

The main advantage of this approach is to build a hypothesis using multiple models [44] and then combine their results to achieve a better outcome. Using a variety of models for selecting features is advantageous because it controls the variance, and reduces the likelihood of poor feature selection [45].

3) DATA STANDARDIZATION PROCESS

The selected feature subset is inspected again using Quantile-Quantile plot (QQ Plot) to check for normality [46]. We apply the z-score normalization on the data to create a Gaussian Distribution with a zero mean and a standard deviation of one [47]. The equation for Z-score normalization is given in eq.(3).

$$Z - score = \frac{y_i - y'}{\sigma_i} \tag{3}$$

Here, y_i represents the feature vector, y' represents the mean of the feature vector and σ_i is the standard deviation.

Data standardization is an important step for most AI algorithms, especially gradient descent-based, such as neural networks, logistic regression, and distance-based algorithms like SVM, KNN, and K-means [48]. These models may not behave as expected [49] if data is not standardized. For gradient descent-based models, feature values affect the step size of the gradient descent [48]. If the features have different ranges, this might cause variations in the step sizes of each feature. The gradient descent will converge smoothly towards the minima when the features have the same scale [48]. Distance-based algorithms use distances between different observations to classify them into different categories. If some features have different ranges, chances are that they will have a higher impact on the AI results, therefore leading to AI bias. By introducing scaling, all features will contribute equally to generate the results [48].

B. SYNTHESIZING ATYPICAL ATTACKS

For this research, we assume that the attacker may employ sophisticated attack tools with the ability to change attack parameters to launch an atypical attack. For this purpose, we have simulated a virtual network where an attacker is launching real attacks on a target Apache server using commonly available DoS tools such as Slowloris and Slow Httptest. These slow-rate attacks exploit the HTTP vulnerability and send incomplete requests to the target server to open as many connections as possible. The target's resources and connections are kept engaged while denying access to legitimate users thus leading to a Denial of Service attack [50]. While Slowloris is a slow HTTP header attack, the Slow Httptest attack can be launched in different modes such as slow header, slow body, and slow read [50]. To understand how these atypical attacks are launched we have also added a diagrammatic representation as depicted in Fig. 4. Although we have employed only two attack classes to evaluate the IDS, our work is equally applicable for other classes of attacks as well.

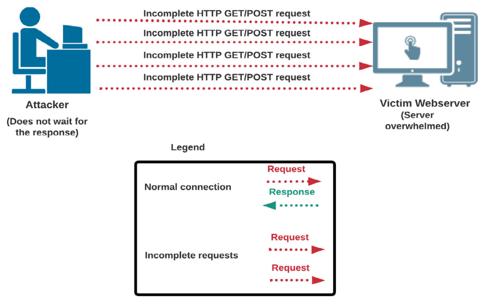

Incomplete HTTP GET/POST request
Incomplete HTTP GET/POST request
Incomplete HTTP GET/POST request
Incomplete HTTP GET/POST request
Attacker
(Does not wait for the response)
Victim Webserver
(Server overwhelmed)
Legend
Normal connection
Request
Response
Incomplete requests
Request
Request

FIGURE 4. **Low bandwidth HTTP DoS attack.**

The atypical attacks are synthesized after running Denial of Service (DoS) attack traffic on a target server. All the attacks are separately launched successfully from the attacker to the Apache Server installed on the target machine by sending incomplete GET/POST requests to the target server. The feature profile of the attack can be changed by mutating the tool-based parameters from their default values to randomized values to launch a new attack. Packets are captured using the Wireshark tool running in the background on the

Algorithm 1: Atypical attack generation & identification

```
Input:
IDS input- original training data $d_{tr}$;
Test data $d_{ts}$ with typical attacks $d_{ta}$ and benign flows
$d_{be}$;
Number of iterations for atypical attack generation - $n$
Output:
Atypical Attacks $d_{aa}$
Trained IDS;
initialize Heterogeneous Feature Selection Ensemble;
initialize True Positive Rate (TPR) threshold as T;
for $i = 1$ to $n$ do
    Attacker $A$ synthesizes atypical attacks $d_{aa}$ by
      randomly changing attack parameters;
end
while true do
    for IDS train_epochs do
        IDS identifies typical attacks $d_{ta}$, atypical
          attacks $d_{aa}$ and benign flows $d_{be}$;
    end
    if $TPR<T$ then
        Hyperparameter Optimization();
    end
    else
        Break;
    end
end
```

target machine. The captured data is then analyzed offline and converted into different features using the network flow analyzer *CICflowmeter* [51], [52]. We validate the success of these attacks on the target server by checking its status for the duration of the attack to confirm access disruption. Fig. 5 depicts the success of the Slow Httptest DoS attack on the target server. The target's services are successfully disrupted by the attacker after requesting 5000 simultaneous connections. The figure shows service unavailable and target unreachable for the duration of the attack.

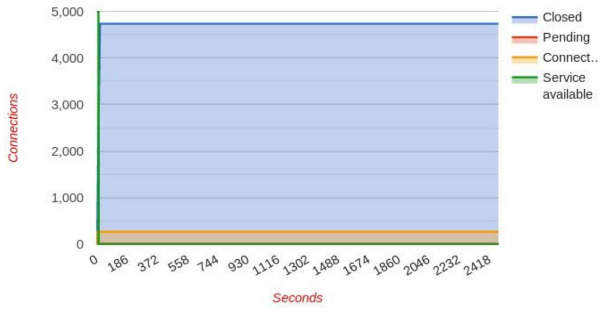


FIGURE 5. Successful Slow Httptest DoS attack (Slow header mode) on the target apache server.

C. TRAINING AND RESULT GENERALIZATION

The next phase for our system includes training AI models with the benchmark data until it achieves the best performance followed by the validation of results. The validation step calculates the loss value of the trained model to measure its performance. The evaluation of the final model is conducted after the training and validation phases are completed.

The term *generalization* in the context of network security refers to the AI-based IDS model's ability to identify mutated attacks as compared to the ones it was previously trained on. In other words, this process provides an evaluation of the ability of an AI-based IDS model to correctly identify attacks with different feature profiles. This is done to ensure that the models will be successful when deployed in a real environment and avoid overfitting. Overfitting occurs when AI models make accurate predictions on training and validation data while making inaccurate predictions on test data (i.e., data that was never used in training). This step is crucial as it provides an unbiased performance evaluation for these IDS models.

D. HYPERPARAMETER OPTIMIZATION

Defensive AI development constitutes training of the model, validation, and generalization of its results, and hyperparameter optimization until the model performance on test data (constituting atypical attacks) shows no more improvement. The performance is measured based on the True Positive Rate (TPR) for attack and True Negative Rate (TNR) for benign flows. The optimization process is performed when the models do not generalize well on atypical attacks in test data with TPR less than a predefined threshold value. In such a case, the model continues retraining until the TPR no longer improves.

Algorithm 1 represents a generic overview of our proposed methodology for synthesizing and identifying atypical attacks.

E. NOVELTY AND ADVANTAGES

To the best of our knowledge, the proposed Defensive AI approach is unique and highly accurate against atypical attack flows. This approach outperforms the state-of-the-art techniques for intrusion detection and network security in several ways. Our novel feature selection technique plays a very important role in making the AI interpretations easy since it picks the most relevant feature subset and minimizes the training loss (which can otherwise be high due to a large number of less significant features). Our training methodology is not based on using subsets from the same benchmark dataset for training and generalization as compared to most of the current researches in the field of cyberattack detection using AI [5]–[7], [10]–[12], [18], [19], [53], [54]. Since these data subsets may have the same probabilistic distribution and feature profiles, it leads to overfitting of AI models [9]. The used atypical attacks are synthesized in a virtual environment to make better generalizations for the AI models. Our approach is unique, compared to other AI cybersecurity [10], [11], [14], [16], [17], [19], in that it improves model generalization

TABLE 3. Best 20 features selected using the HSFE technique.

S.No.	Feature name	Chi2	RFE	LR	RFC	L-GBM	ETC	L-SVC	# votes
1	pkt len var	✓	✓	✓	✓	✗	✓	✓	6
2	pkt len std	✓	✓	✓	✓	✓	✓	✗	6
3	max pkt len	✓	✓	✓	✓	✗	✓	✓	6
4	fwd pkts/s	✓	✓	✓	✗	✓	✓	✓	6
5	fwd IAT max	✓	✓	✓	✗	✓	✓	✓	6
6	flow IAT max	✓	✓	✓	✓	✓	✓	✗	6
7	init win bytes fwd	✓	✗	✗	✓	✓	✓	✓	5
8	fwd pkt len mean	✓	✓	✓	✓	✗	✓	✗	5
9	fwd IAT std	✓	✓	✗	✓	✗	✓	✓	5
10	fwd IAT mean	✓	✓	✓	✗	✓	✗	✓	5
11	flow IAT std	✓	✓	✓	✗	✓	✗	✓	5
12	pkt len mean	✓	✗	✗	✓	✗	✓	✓	4
13	min pkt len	✓	✓	✓	✗	✗	✗	✓	4
14	flow IAT mean	✓	✓	✗	✗	✓	✗	✓	4
15	flow duration	✓	✗	✓	✗	✓	✓	✗	4
16	avg pkt size	✓	✗	✓	✓	✗	✓	✗	4
17	fwd pkt len std	✗	✓	✓	✗	✗	✗	✓	3
18	fwd pkt len min	✓	✓	✗	✗	✗	✗	✓	3
19	fwd pkt len max	✗	✓	✗	✓	✗	✓	✗	3
20	fwd IAT tot	✓	✗	✗	✗	✓	✓	✗	3

TABLE 4. Different feature profiles for Slow Httptest DoS attack.

Atypical Attack	Mode	Duration (seconds)	Number of Flows	c	i	r	l	t	x	p	s	w	y	n	z	k
Atypical Attack-1	H	240	1473	1000	10	300	240	GET	24	3	-	-	-	-	-	-
Atypical Attack-2	H	2475	8550	5000	25	500	10800	GET	24	10	-	-	-	-	-	-
Atypical Attack-3	B	7200	17185	5000	120	500	7200	POST	10	10	4096	-	-	-	-	-
Atypical Attack-4	X	1800	9205	10000	10	450	1800	GET	32	5	-	512	1024	5	32	3

using hyperparameter optimization and retraining to produce unbiased classifiers.

VI. EXPERIMENTS AND PERFORMANCE EVALUATION

This section explains in detail the training and validation of all ML/DL classifiers using CICIDS2017 data and their performance measurement. The machine used in our research to conduct the experiments consists of the following configurations: Intel Xenon CPU E5-2630 v3@ 2.40GHz, 480GB SSD, 64-bit Windows 10 Pro, and NVIDIA Quadro K2200. The Hyperparameter optimization step for various AI models was implemented on a server installed with 4 GPUs (GeForce GTX 1080 Ti with a compute capability of 6.1).

A. CICIDS2017 DATASET DETAILS

CICIDS2017 benchmark dataset is used only for training and validation of the AI models against typical attacks and benign flows. Fig. 6 shows the CICIDS2017 attack versus benign flows. This benchmark dataset contains the most recent attacks that resemble real-world attacks [51], [52]. The total percentages of DoS Hulk, DDoS, DoS GoldenEye, DoS Slowloris and DoS SlowHttptest attack flows are 22.04%, 12.21%, 0.98%, 0.55% and 0.52% respectively. While the total percentage of benign flows in this dataset is 63.70%.

The dataset consists of 75 features. These features are generated by feeding the packet captures that resemble real-world data into the CICFlowmeter. A detailed explanation of all the features can be found in [55]. Table 3 shows the best

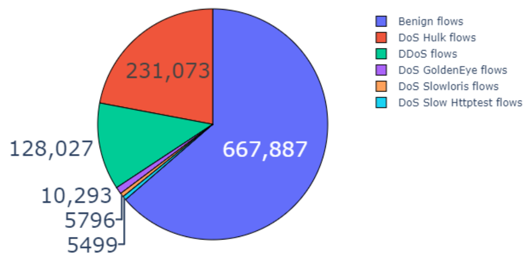


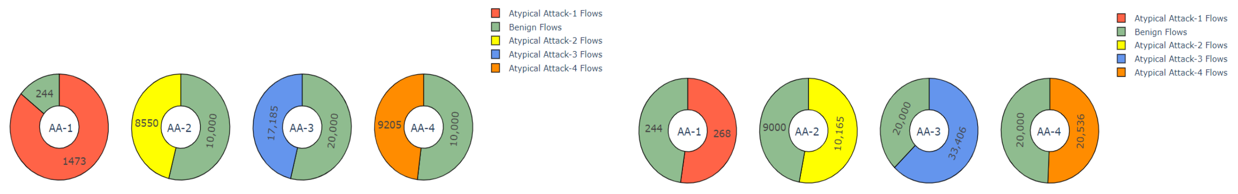
FIGURE 6. **CICIDS2017 attack versus benign flows.**

20 features selected using the proposed feature selection technique based on their respective rank in the voting score.

B. ATYPICAL DATA DETAILS

We synthesized 8 atypical attacks, 4 each for two low-rate DoS attack classes, Slowloris and Slow Httptest DoS. These attacks have similar features as that of training data, but having a different range of values of features and are synthesized in different network settings as compared to training data.

Slow Httptest attack is launched in 3 different modes namely Slow Header (H), Slow Body (B), and Slow Read (X) to create different feature profiles for the attack. The features used for synthesizing Slow Httptest atypical attacks are explained in detail in Table 5. We mutate these tool-based features from their default values to random values to generate new feature profiles. The synthesized atypical attacks with different feature profiles are represented in Table 4.



(a) Slow Httptest Atypical Data (b) Slowloris Atypical Data

FIGURE 7. Atypical attack and benign flows. "AA" is used to depict feature profiles for atypical attack (1-4) both for Slow Httptest and Slowloris attack classes.

TABLE 5. Features for synthesizing Slow Httptest DoS attacks.

Feature	Description
c	number of target connections
i	followup data interval in seconds
r	number of connections per second
l	length of attack in seconds
t	request verb (GET/POST)
x	maximum length of followup data per tick
p	timeout to wait for http response in seconds
s	size of content length header in bytes
w	start range of advertised window in bytes
y	end range of advertised window in bytes
n	interval between read operations in recv buffer in seconds
z	bytes to slow read from recv buffer with single read call
k	number of times to repeat same request in the connection

The CICflowmeter (flow analyzer) is used to generate flows from raw attack packet captures. The number of attack flows generated depends on the time duration of the attack. If the attack time duration is higher, a greater number of attack flows will be generated.

As shown in Fig. 7(a), the number of attack flows for Slow Httptest atypical attack-1, attack-2, attack-3 and attack-4 are 1473, 8550, 17185 and 9205 respectively. We would like to indicate that in our case, the number of atypical attack flows is expected to be small since our intention is precisely to examine how a defensive AI engine can deal with mutated attacks that are not present in the training data. We have selected benign flows randomly from CICIDS2017 test (hold-out) data to match the number of attack flows for analyzing AI models using different performance metrics. The total number of random benign flows for Slow Httptest DoS atypical attack-1, 2, 3, and 4 are 244, 10000, 20000, and 10000 respectively.

For Slowloris DoS, we use target port number (p), number of sockets (s), and random user agents (ua) features to generate atypical attacks. The random user agents feature is used to randomize user agents for each request. The details of atypical slowloris attacks with different feature profiles are given in Table 6. As shown in Fig. 7(b), Slowloris atypical attack-1 consists of 268 flows while atypical attack-2 consists of 10165 attack flows, attack-3 and attack-4 consist of 33406 flows and 20536 flows respectively. For Slowloris DoS atypical attack-1, 2, 3, and 4, the number of randomly selected benign flows are 244, 9000, 20000, and 20000 respectively.

TABLE 6. Different feature profiles for Slowloris DoS attack.

Atypical Attack	Duration (seconds)	Number of Flows	p	s	ua
Atypical Attack-1	900	268	80	265	No
Atypical Attack-2	3600	10165	80	500	No
Atypical Attack-3	10800	33406	80	1000	No
Atypical Attack-4	7200	20536	80	5000	Yes

C. PERFORMANCE METRICS

Multiple statistical measures are used to evaluate the AI models. These metrics are used to select the best model for binary classification that classifies input data into 'Attack' or 'Benign'. The performance metrics used to evaluate the AI models are:

- **Accuracy**
  The fraction of the number of correctly identified attack and benign flows with respect to the total number of input flows.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{4}$$

- **Precision**
  The fraction of the number of correctly classified attack flows among all the flows classified as attacks. It is also known as Positive Predictive Value.

$$Precision = \frac{TP}{(TP + FP)} \tag{5}$$

- **Recall/Sensitivity/True Positive Rate (TPR)**
  The fraction of the number of correctly classified attack flows among all the attack flows in the data. It is also known as Hit rate.

$$Recall = \frac{TP}{(TP + FN)} \tag{6}$$

- **Specificity/Selectivity/True Negative Rate (TNR)**
  The fraction of the number of correctly classified benign flows among all the benign flows in the data.

$$TNR = \frac{TN}{(TN + FP)} \tag{7}$$

- **False Alarm Rate (FAR)/ False Positive Rate (FPR)**
  The fraction of the number of benign flows wrongly classified as attack with respect to the total number of

benign flows.

$$FAR = \frac{FP}{(FP + TN)} \tag{8}$$

- **F1 score**
  It measures the robustness of the model. It is represented by the harmonic mean of precision and recall. The F1 score ranges between 0 to 1 and the higher the value, the better the classifier is in detecting abnormal traffic.

$$F1\ Score = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \tag{9}$$

Here, TN, TP, FN, and FP represent True Negatives, True Positives, False Negatives, and False Positives respectively.

D. TRAINING AND VALIDATION PHASES

After the feature selection phase, we train and validate different ML/DL classifiers such as Deep Neural Network (DNN), Multi-Layer Perceptron (MLP), K-Nearest Neighbor (KNN), Bernoulli Naïve Bayes (BNB), Gaussian Naïve Bayes (GNB), Logistic Regression (LR), Linear Support Vector Classifier (L-SVC), Linear Discriminant Analysis (LDA), Decision Tree Classifier (DTC) and Random Forest Classifier (RFC) on the CICIDS2017 data. All ML classifiers are trained with their default hyperparameter configurations on 75% train data.

To train a DNN model, 20 input dimensions are used (equal to the number of selected features). The model has 3 Dense layers with Rectified Linear Unit (ReLU) activation functions, each has 60 neurons. Each Dense layer is followed by Dropout with a value of 0.2 to avoid overfitting. The last Fully Connected (FC) layer along with Sigmoid activation provides output probabilities for attack or benign traffic. The model is trained for 200 epochs, but early stopping is applied to reduce overfitting. The batch size is set to 1024 and the learning rate is 0.0001.

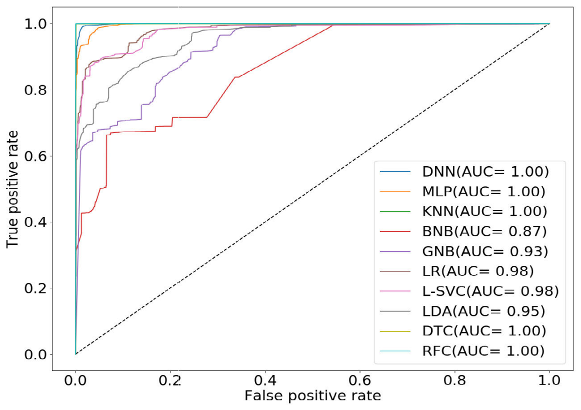


FIGURE 8. ROC comparison of AI models on CICIDS2017 test data (typical attack and benign flows).

Fig. 8 shows a comparison between various ML/DL models based on their AUC values on CICIDS2017 test data which consists of typical/known attacks. This hold-out data consists of 262143 flows (both attack and benign) and constitutes 25% of the total CICIDS2017 data flows. AUC measures the ability of a classifier to distinguish between classes by calculating the area under the Receiver Operating Characteristic (ROC) graph. The ROC graph represents the plot between TPR and FPR at various threshold values. AUC values lie in the range from 0 to 1. The higher the AUC value, the better the model performance. Fig. 8 shows that the best AUC is achieved by models such as DNN, MLP, KNN, DTC, and RFC. Other models such as LR and L-SVC also achieve AUC values as high as 0.98. Models such as BNB, GNB, and LDA also show similar improvements.

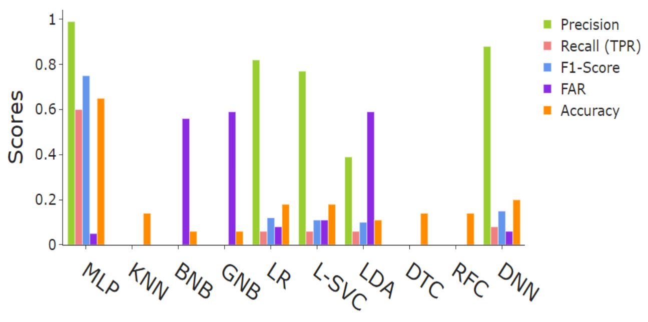
E. MODEL GENERALIZATION AND EVALUATION

The pre-trained ML/DL models are evaluated against synthesized atypical attacks to improve model generalization. We conduct this evaluation before HPO to analyze which AI model can identify these attacks with higher detection rates. We compare the performances of these models using one atypical attack (atypical attack-1) from both the attack classes namely, Slow Httptest and Slowloris. Other synthesized atypical attacks are used to evaluate multiple AI models after the hyperparameter optimization phase. The TNR for benign flows and TPR for attack flows are separately measured as shown in Table 7. Although most models provide high TNR on CICIDS2017 test data, they generally have poor performance on the atypical attack-1 from both Slow Httptest and Slowloris attack classes. LR, L-SVC, and DTC can detect only Slowloris atypical attack-1 with 98.88% TPR and benign data with TNR of 91.80%, 89.34%, and 100% respectively. All other models perform poorly on both attacks.

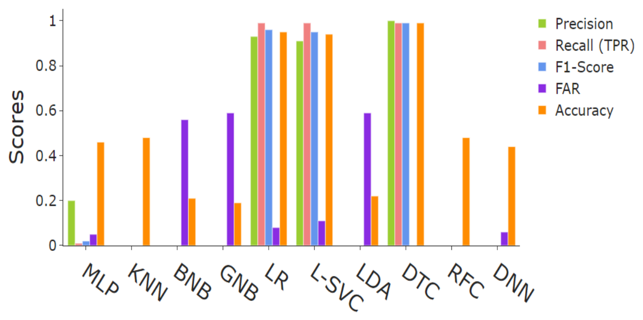
TABLE 7. Performance of AI models on atypical attack-1 & benign flows.

Atypical Attack-1			
Model	**TNR**	**TPR (Slow Httptest)**	**TPR (Slowloris)**
DNN	93.44%	8.35%	0%
MLP	95.08%	59.88%	1.12%
KNN	100%	0%	0%
BNB	43.85%	0%	0%
GNB	40.98%	0%	0%
LR	91.80%	6.31%	98.88%
L-SVC	89.34%	5.97%	98.88%
LDA	45.49%	5.70%	0%
DTC	100%	0%	98.88%
RFC	100%	0%	98.88%

A comparative analysis of different AI models using multiple performance metrics including precision, recall, F1 score, FAR, and accuracy is shown in Fig. 9. The figure shows that LR, L-SVC, and DTC have a better performance on Slowloris atypical attack-1 while MLP performs better on Slow Httptest atypical attack-1 only. This signifies that the studied AI models are unable to perform well in identifying both classes of atypical attacks. The reason for this poor performance on atypical attacks is due to the overfitting problem of these models. Therefore, we subject the AI classifiers



(a) Slow Httptest Atypical Attack-1



(b) Slowloris Atypical Attack-1

FIGURE 9. Evaluation of AI models on atypical attack and benign flows.

to further optimization of hyperparameters to improve their generalization on atypical attacks.

VII. HYPERPARAMETER OPTIMIZATION

Since the TPR of the AI models employed for evaluation on atypical attack-1 is less than the predefined threshold value (in our case, 80%), these models need HPO to further improve their performance. For simplicity, we selected the models that could identify at least one of the synthetic atypical attack classes during evaluation to undergo further optimization. These models are L-SVC, DNN, and DTC. Other AI models discussed in this paper can also be optimized using this technique but they are not examined further in this paper. We also point out that our approach is generic and can be applied to any AI-specific model.

For the hyperparameter optimization phase, AI models are trained with the CICIDS2017 train data using multiple hyperparameter (HP) settings. Then, their performances are evaluated against atypical attack-1 data (both Slowloris and Slow Httptest DoS) to select the best set of hyperparameters. Further evaluation of these optimized AI models is done using multiple atypical attacks discussed in section VII.

A. HPO AND GENERALIZATION FOR DNN

We apply Manual Search Hyperparameter Optimization (MS-HPO) on the DNN model to improve the TPR and TNR for attack and benign flows, respectively. The input dimensions for model architecture are 20 since our feature selection technique uses 20 features. The model has 3 Dense layers with Rectified Linear Unit (ReLU) activation functions. The first two layers have 1024 neurons while the third dense layer has 512 neurons. Each Dense layer is followed by Dropout with a value of 0.3 to avoid overfitting. The last Fully Connected (FC) layer along with Sigmoid activation provides output probabilities of synthesized test data being ‘attack’ or ‘benign’. These parameters are kept constant during the MS-HPO process. We have only utilized batch size (bs), number of epochs, and learning rate (lr) for HPO purposes. We conducted the MS-HPO for the DNN model with different batch sizes, number of epochs, and learning rates for a total of 10 iterations out of which only the best 5 are discussed further in this paper.

Table 8 shows the results of hyperparameter tuning on the DNN model. We observe that the last two hyperparameter configurations provide better results than others. We select the last configuration with a learning rate of 0.0005, batch size of 16, and 25 epochs. Our goal is to improve the TPR of attack flows as well as TNR for benign flows, thereby reducing both FNR and FPR. After employing our proposed HPO technique for the DNN model, the TNR for Slow Httptest atypical attack-1 improves by 51.26% and by 100% for Slowloris atypical attack-1 as compared to the DNN model discussed in Table 7.

TABLE 8. Effect of MS-HPO on DNN model.

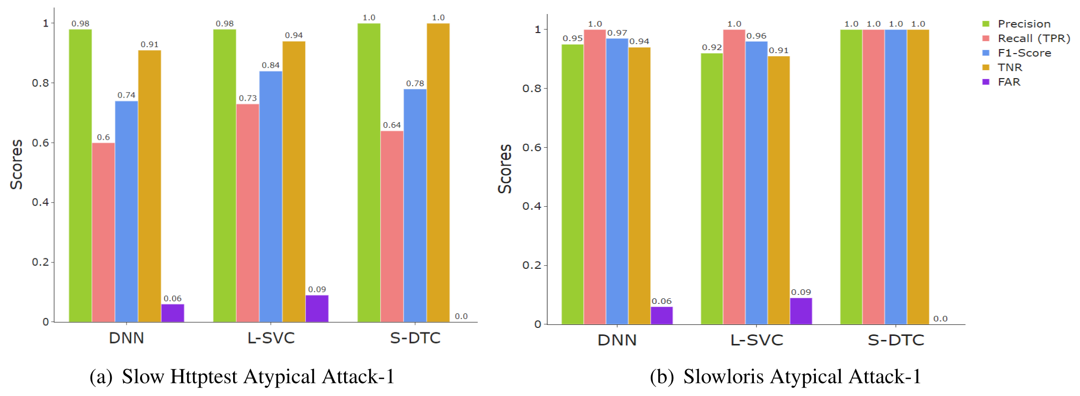
	Atypical Attack-1		
HP	**TNR**	**TPR (Slow Httptest)**	**TPR (Slowloris)**
lr-0.001 bs-1 Epochs-12	76.23%	0.00%	0.00%
lr-0.001 bs-6 Epochs-15	97.95%	6.31%	0.00%
lr-0.0008 bs-8 Epochs-10	97.95%	58.04%	1.11%
lr-0.001 bs-8 Epochs-15	**32.79%**	**69.93%**	**100%**
lr-0.0005 bs-16 Epochs-25	**93.85%**	**59.61%**	**100%**

B. HPO AND GENERALIZATION FOR LINEAR-SVC

Since L-SVC is the best performing model among linear classifiers after the feature engineering phase, we select it for further performance improvement using GS-HPO. The CICIDS2017 data is divided into two datasets, 70% for training and 30% for validation. The model is evaluated on Slow Httptest Atypical Attack-1 and Slowloris Atypical Attack-1. Table 9 discusses the hyperparameters (HP) used in GS-HPO for the L-SVC model.

TABLE 9. Hyperparameter optimization for Linear classifiers.

Model	HP	Description	Values
L-SVC	C	Regularization	[0.8,1.0,1.5,2.0]
	tol	Tolerance for stopping	[0.001,0.01,0.1,1]



(a) Slow Httptest Atypical Attack-1 (b) Slowloris Atypical Attack-1

FIGURE 10. **Evaluation of AI models after HPO on atypical attack-1 and benign flows.**

The L-SVC model is retrained based on the best hyperparameters selected by Grid Search Hyperparameter Optimization (GS-HPO) which are $C = 1.5$ and $tolerance = 1$. Here C represents the regularization parameter while *tolerance* represents the tolerance for stopping criteria [56]. It is observed from Fig. 10 that after the HPO phase, the performance of L-SVC for benign flows improves by 1.64% whereas, for Slow Httptest Atypical Attack-1, TPR improves by 67.01% and for Slowloris Atypical Attack-1, it improves by only 1.12% as compared to the L-SVC results depicted in Table 7.

C. HPO AND GENERALIZATION FOR TREE-BASED MODELS

We use the Tree-Based Optimization Tool (TPOT) [57], which is based on Evolutionary Search Hyperparameter Optimization (ES-HPO) explained in the background section of this paper, to find the right set of hyperparameters for building the AI model. This optimization process took 48 hours with a population size of 50 and 5 generations to find the right hyperparameter set for Stacked Decision Tree Classifier (S-DTC). The *population size* in this case is a positive integer that depicts the number of individuals in a population. *Generations* is also a positive integer representing the number of iterations to run the pipeline optimization process [57]. The resultant S-DTC combines the power of two DTCs stacked together in the ensemble. The first classifier has a maximum depth of 10, minimum leaf samples of 4, and minimum split samples of 20. While the second classifier has a maximum depth of 10, minimum leaf samples of 10, and minimum split samples of 16.

Fig. 10 represents the performance evaluation of DNN and L-SVC, and S-DTC models after HPO in terms of multiple performance metrics such as precision, recall, F1 score, and FAR. The results show an improvement in the performance metrics for these models as compared to Fig. 9. For example, the recall value for L-SVC on Slow Httptest is improved from 5.97% (before HPO) to 72.98% (after HPO). For DNN, the recall value has improved from 8.35% (before HPO) to 60% (after HPO). As seen in the figure, the AI models perform well when evaluated on Slowloris atypical attack-1 with all the performance metric values equal to or above 92%. It is observed that L-SVC has a better performance in terms of TPR, and F1 Score as compared to DNN, and S-DTC. However, the S-DTC gives the lowest FAR of almost 0% as compared to other AI models.

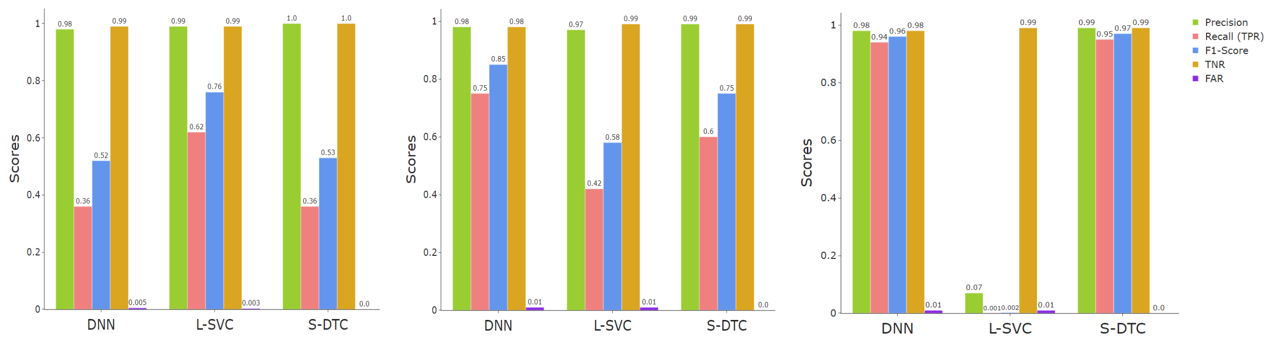
TABLE 10. **A comparison between the proposed approach and current research for typical, Atypical attacks and benign flows. Here, Typical attack represents the hold-out attack from CICIDS2017. ShttpAA1 represents Slow Httptest atypical attack-1 and SlowAA1 represents Slowloris atypical attack-1.**

Model	TNR *Benign*	TPR *Typical*	TNR *Benign*	TPR *ShttpAA1*	TPR *SlowAA1*
L-SVC [our approach]	98.69%	75.70%	90.98%	**72.98%**	**100%**
DNN [our approach]	96.25%	96.07%	93.85%	59.67%	**100%**
S-DTC [our approach]	**99.92%**	**99.90%**	**100%**	64.02%	**100%**
L-SVC [58]	98.66%	77.93%	92.62%	5.70%	98.88%
DNN [18]	98.57%	80.75%	94.67%	57.50%	1.11%
CNN [20]	99.76%	97.11%	100%	0%	0%
LSTM [21]	98.64%	98.57%	97.13%	18.67%	0%

Table 10 shows a comparison of L-SVC, DNN, and S-DTC models trained using our approach with current research for both typical and atypical data based on TPR and TNR values. We train the models presented in [18], [20], [58], and [21] with their default hyperparameters and evaluate their performance using typical attacks, atypical attack-1 and benign flows. Stable results of the AI models trained using our approach on both typical as well as atypical attacks compared to other state-of-the-art models indicate the first-rate performance of our approach.

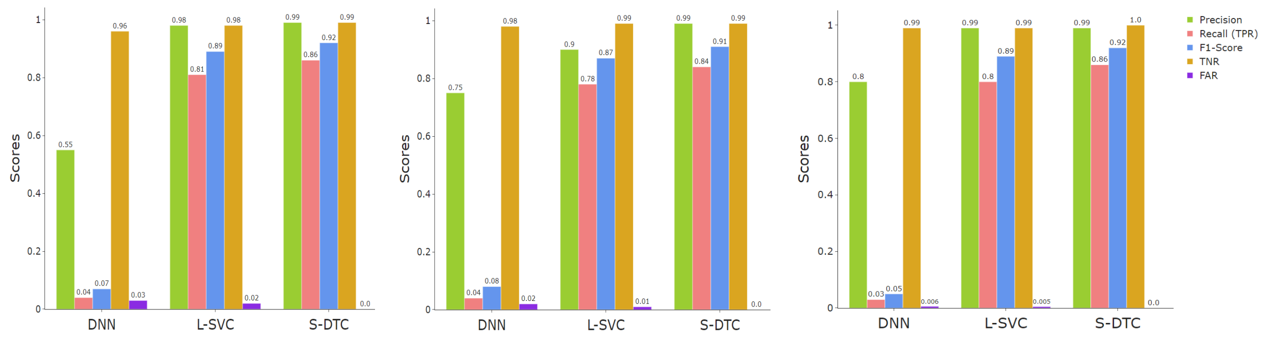
D. EVALUATION AND COMPARISON OF AI MODELS ON ATYPICAL ATTACKS

This section provides an evaluation of the performance of AI models namely DNN, L-SVC, and S-DTC against atypical attacks. The atypical attack and benign flows (2 to 4) shown in Fig. 7(b) and Fig. 7(a) are used for further comparison of the AI models' detection capability.



(a) Atypical Attack-2 (b) Atypical Attack-3 (c) Atypical Attack-4

FIGURE 11. **Comparison of AI models on Slow Httptest atypical attack and benign flows after HPO.**



(a) Atypical Attack-2 (b) Atypical Attack-3 (c) Atypical Attack-4

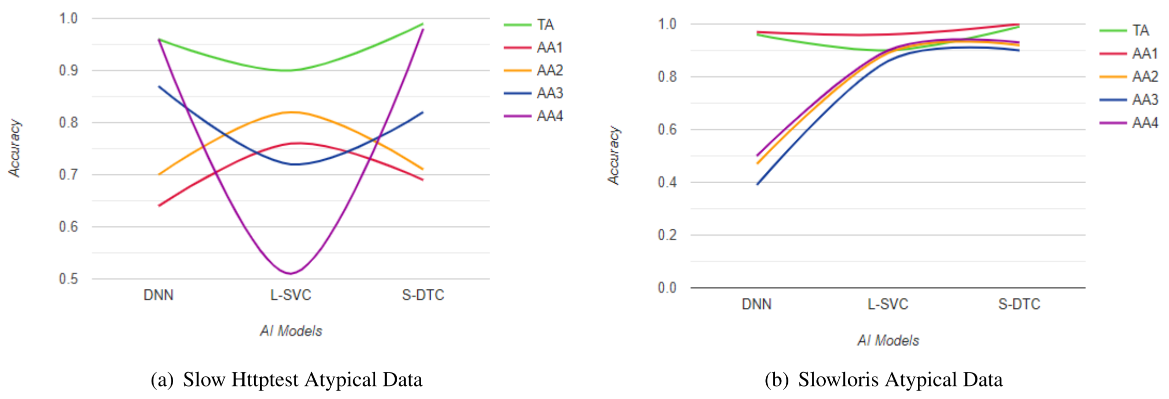
FIGURE 12. **Comparison of AI models on Slowloris atypical attack and benign flows after HPO.**

The performance comparison of DNN, L-SVC, and S-DTC on different atypical attacks belonging to the Slow Httptest DoS attack class is shown in Fig. 11. Although all 3 AI models achieve high precision, high TNR, and very low FAR scores indicating their superior performance on benign flows. Yet their performance (Recall) on atypical attack flows is lower and lies in the range from 36% to 95%. For atypical attack-2, L-SVC gives the best performance with 62% recall. The DNN recall value for atypical attack-3 is 75% which is the highest among all the models. For atypical attack-4, DNN and S-DTC have high recall values of 94% and 95% respectively whereas L-SVC was not successful in identifying this attack. The Precision and F1 score values for all these models also vary according to their performances in identifying attack and benign flows. We observe that none of the models can identify all 3 attacks with high recall values. DNN and S-DTC achieve higher recall rates as compared to L-SVC which performs poorly on atypical attack-3 and 4.

Fig. 12 compares the performances of DNN, L-SVC, and S-DTC on Slowloris atypical attacks-2, 3, and 4. The results depict a similar performance on all three attacks since all the attacks are launched in a slow header mode. These attacks have similar features to the typical attacks in the training dataset but have a mutated feature profile. In comparison, Slow Httptest DoS atypical attacks in Fig. 11, depict different results. One of the reasons is that these atypical attacks are launched in three different modes namely, Slow header, Slow body, and Slow read. Therefore, they belong to three different categories of slow-rate DoS Attacks launched using the same attack tool.

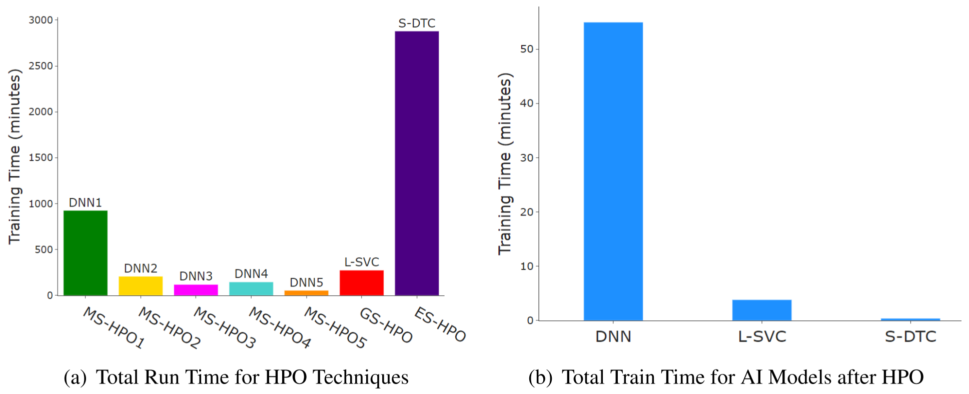
For Slowloris DoS Atypical attacks in Fig. 12, DNN, L-SVC, and S-DTC achieve high TNR and a very low FAR for all the cases which imply that benign flows can be accurately identified. From the results on atypical attack-2, 3, and 4, DNN is the worst performing model for the identification of these attacks with recall values of 4%, 4%, and 3% respectively. Whereas L-SVC and S-DTC achieve higher recall values on all three attacks indicating that for these cases, they provide better atypical attack identification as compared to DNN. For L-SVC, the recall values for atypical attack-2, 3, and 4 are 81%, 78%, and 80% respectively. Results on atypical attack-2, 3, and 4 indicate that for Slowloris attack class, S-DTC is the best performing model with recall values of 86%, 84%, and 86% respectively.

We provide an analysis of the overall accuracy of the AI models on typical attacks, multiple atypical attacks from Slow Httptest DoS and Slowloris DoS attack classes as well as benign flows in Fig. 13(a) and Fig. 13(b) respectively. All the models, DNN, L-SVC, and S-DTC achieve higher accuracy on typical attack data as compared to atypical attack data. For Slow Httptest DoS, the overall accuracy is lower as compared to Slowloris DoS attacks. The accuracy for L-SVC



(a) Slow Httptest Atypical Data (b) Slowloris Atypical Data

FIGURE 13. Accuracy comparison of AI models after HPO on typical and atypical data. Note that typical attacks (TA) represent the attacks in the hold-out CICIDS2017 data and AA represents an atypical attack.



(a) Total Run Time for HPO Techniques (b) Total Train Time for AI Models after HPO

FIGURE 14. Time complexity analysis for Hyperparameter Optimization (HPO) techniques and selected AI classifiers after HPO. DNN(1-5) represent different hyperparameter configurations for DNN model (discussed in Table 8). Here, MS-HPO represents Manual Search HPO, GS-HPO represents Grid Search HPO, and ES-HPO represents Evolutionary Search HPO.

on Slow Httptest Atypical attack-4 is the lowest around 50% approximately. For DNN and S-DTC, the accuracy is equal to or greater than 65%. For Slowloris DoS, L-SVC, and S-DTC in general have a higher accuracy on atypical attack data as compared to DNN which is the worst-performing model on most of the atypical attacks.

Although we observe some enhancement of model performances against atypical attacks after the HPO phase, yet further optimization and retraining are needed to meet the pre-defined threshold values for recall. Overall results indicate that S-DTC performs relatively better than all the models in comparison except on Slow Httptest atypical attack-2 where its recall value is only 36%. The other two models in comparison, DNN, and L-SVC also have lower recall values (36% and 62% respectively) on this attack. We believe that the higher misclassification rate for atypical attacks may be due to the under-representation of this attack feature profile in the training data. We plan to investigate this poor performance issue, re-optimization of AI models, and retraining phases in our future work using an incremental learning approach.

We follow the approach in [4] to provide a complexity analysis for the hyperparameter techniques employed in this research as well as for training the AI models in Fig. 14. As shown in Fig. 14(a), we compare the total run time required to train a DNN model with different sets of hyperparameters for identifying atypical attacks (MS-HPO) with GS-HPO for L-SVC, and ES-HPO for S-DTC. After analysis, the results indicate that ES-HPO for the S-DTC takes the longest time (2880 minutes) to identify the hyperparameters for atypical attack detection. We train the AI models DNN, L-SVC, and S-DTC with the set of hyperparameters selected during the HPO process for each model. Our results highlight that after the HPO phase, the training complexity for DNN, L-SVC, and S-DTC is reduced as indicated in Fig. 14(b). S-DTC takes the least time to train among all the models whereas DNN takes the longest time of 54.98 minutes owing to the complex deep learning model structure.

VIII. CONCLUSION, CHALLENGES AND FUTURE DIRECTIONS

This paper proposes a novel attack detection approach that employs two-fold feature selection and hyperparameter optimization techniques to defend against atypical attacks. We provide an extensive experimental analysis based on the proposed Defensive AI Engine to compare and evaluate multiple AI models such as DNN, L-SVC, and S-DTC against atypical attacks. These AI models are trained using benchmark data and evaluated using synthesized atypical attacks with different feature profiles. Our experiments demonstrate that the performance of the AI models trained using our approach shows significant improvement after the hyperparameter optimization phase against typical as well as atypical attacks. As compared to state-of-the-art approaches, the AI models trained using our approach achieve better TPR and TNR on atypical attack-1 and benign flows. The comparative performance analysis demonstrates that overall S-DTC outperforms the other two contenders (DNN and L-SVC) for atypical attack identification.

We highlight several technical challenges identified during this research and provide some insights to deal with them. Our main motivation behind this work is the lack of an extensive evaluation of IDS models against atypical and completely unknown attacks. To enhance the current knowledge of dynamically changing attacks, we propose our novel defensive AI engine to build and analyze IDS models against such attacks. Some important factors to consider while building an AI model for IDS are the size of available training data, the number of features, and the training time required. The first important step for our methodology is selecting the most relevant features to minimize training loss and to make AI interpretations easier.

We employ multiple performance metrics such as precision, recall, F1 score, TNR, FAR, and accuracy to evaluate the AI models against atypical attacks. Relying only on a single metric such as accuracy can be misleading and can add more bias to the classification of new attacks especially when there is a huge class imbalance problem [59]. For example, if 90% of the instances in our train data are benign (majority class) and all these benign instances are correctly identified, then the overall accuracy of the system will be 90% even though 0% attacks (minority class) were identified.

We provide an analysis of time complexity for the hyperparameter optimization phase as well as the training phase of IDS models. This step is crucial especially in a dynamically changing attack environment, where there is a need for continuous retraining of the IDS as new attack data is generated [4]. From our investigation through this research, we deduce that the selection of optimal hyperparameters is a very important step to build enhanced IDS models that can face evolving attack strategies. But this process can be very intensive, time-consuming, and complex. Although we see improvements in the performances of our selected AI models over multiple atypical attacks, they need more optimization of hyperparameters and further retraining to meet predefined TPR thresholds. These re-optimization and retraining phases are left for future research.

Extensive improvements in the results may be expected in the future especially with the recent advances in adversarial learning and other semi-supervised approaches. These techniques use adaptive algorithms to generate network behavior that has never been seen before. They are aware of the network dynamics and can predict what will happen within the next time frame. Therefore, as future work, we plan to explore adversarial semi-supervised methods to defend against atypical attacks.

TABLE 11. Common acronyms used in this paper with their description in alphabetical order.

Acronym	Description
AA	Atypical Attack
BNB	Bernoulli Naïve Bayes
bs	Batch Size
CNN	Convolutional Neural Network
DL	Deep Learning
DNN	Deep Neural Network
DTC	Decision Tree Classifier
ES-HPO	Evolutionary Search Hyperparameter Optimization
ETC	Extra Tree Classifier
FAR	False Alarm Rate
GNB	Gaussian Naïve Bayes
GS-HPO	Grid Search Hyperparameter Optimization
HFSE	Heterogenous Feature Selection Ensemble
HP	Hyperparameter
HPO	Hyperparameter Optimization
IDS	Intrusion Detection System
KNN	K-Nearest Neighbor
LDA	Linear Discriminant Analysis
L-GBM	Light Gradient Boosting Machine
LR	Logistic Regression
lr	Learning Rate
LSTM	Long short-term memory
L-SVC	Linear Support Vector Classifier
ML	Machine Learning
MLP	Multi- Layer Perceptron
MS-HPO	Manual Search Hyperparameter Optimization
RFC	Random Forest Classifier
RFE	Recursive Feature Elimination
S-DTC	Stacked Decision Tree Classifier
TA	Typical Attack
TNR	True Negative Rate
TPOT	Tree-Based Optimization Tool
TPR	True Positive Rate

Our aim through this research is to identify attacks with different feature profiles. We have kept aside a subset of training data for testing against typical attack flows, as it is common practice in security analysis. Our focus is particularly on attacks that can mutate themselves into atypical attacks, i.e., starting from a known attack and then modifying into feature profiles that would evade the IDS. We employed two classes of slow-rate DoS attacks, Slowloris and Slow Httptest to synthesize atypical attacks for IDS evaluation. Such attacks may be difficult to trace since the attacker may not send any malicious content when sending multiple requests to overwhelm the target server. In our future research, to evaluate the IDS we plan to include other classes of atypical attacks as well. We further plan to investigate the identification of completely unknown network attacks by evaluating the IDS models trained using one dataset against other currently available attack datasets such as ISCX2012, UNSW-NB15, and DDoS 2019. Finally, we intend to analyze the trained IDS models in a real network testbed.

APPENDIX A ABBREVIATIONS

Table 11 lists and describes the commonly used acronyms in this paper in alphabetical order.

REFERENCES

[1] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *Proc. Int. Conf. Platform Technol. Service (PlatCon)*, Feb. 2016, pp. 1–5.

[2] C. Liu, Y. Liu, Y. Yan, and J. Wang, "An intrusion detection model with hierarchical attention mechanism," *IEEE Access*, vol. 8, pp. 67542–67554, 2020.

[3] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "FS-Net: A flow sequence network for encrypted traffic classification," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2019, pp. 1171–1179.

[4] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescape, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 2, pp. 445–458, Jun. 2019.

[5] M. Z. Alom, V. Bontupalli, and T. M. Taha, "Intrusion detection using deep belief networks," in *Proc. Nat. Aerosp. Electron. Conf. (NAECON)*, Jun. 2015, pp. 339–344.

[6] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[7] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48231–48246, 2018.

[8] H. Nguyen and D. Choi, "Network anomaly detection: Flow-based or packet-based approach?" 2010, *arXiv:1007.1266*. [Online]. Available: http://arxiv.org/abs/1007.1266

[9] U. Sabeel, S. S. Heydari, H. Mohanka, Y. Bendhaou, K. Elgazzar, and K. El-Khatib, "Evaluation of deep learning in detecting unknown network attacks," in *Proc. Int. Conf. Smart Appl., Commun. Netw. (SmartNets)*, Dec. 2019, pp. 1–6.

[10] A. R. Syarif and W. Gata, "Intrusion detection system using hybrid binary PSO and K-nearest neighborhood algorithm," in *Proc. 11th Int. Conf. Inf. Commun. Technol. Syst. (ICTS)*, Oct. 2017, pp. 181–186.

[11] R. Shah, Y. Qian, D. Kumar, M. Ali, and M. Alvi, "Network intrusion detection through discriminative feature selection by using sparse logistic regression," *Future Internet*, vol. 9, no. 4, p. 81, Nov. 2017.

[12] K. Peng, V. C. M. Leung, and Q. Huang, "Clustering approach based on mini batch Kmeans for intrusion detection system over big data," *IEEE Access*, vol. 6, pp. 11897–11906, 2018.

[13] D. Aksu, S. Üstebay, M. A. Aydin, and T. Atmaca, "Intrusion detection with comparative analysis of supervised learning techniques and Fisher score feature selection algorithm," in *Proc. Int. Symp. Comput. Inf. Sci.* Cham, Switzerland: Springer, 2018, pp. 141–149.

[14] A. Ahmim, L. Maglaras, M. A. Ferrag, M. Derdour, and H. Janicke, "A novel hierarchical intrusion detection system based on decision tree and rules-based models," in *Proc. 15th Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, May 2019, pp. 228–233.

[15] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, and A. Abuzneid, "Features dimensionality reduction approaches for machine learning based network intrusion detection," *Electronics*, vol. 8, no. 3, p. 322, Mar. 2019.

[16] T. Ma, F. Wang, J. Cheng, Y. Yu, and X. Chen, "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks," *Sensors*, vol. 16, no. 10, p. 1701, Oct. 2016.

[17] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.

[18] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[19] M. Roopak, G. Y. Tian, and J. Chambers, "An intrusion detection system against DDoS attacks in IoT networks," in *Proc. 10th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2020, pp. 0562–0567.

[20] M. Azizjon, A. Jumabek, and W. Kim, "1D CNN based network intrusion detection with normalization on imbalanced data," in *Proc. Int. Conf. Artif. Intell. Inf. Commun. (ICAIIC)*, Feb. 2020, pp. 218–224.

[21] M. D. Hossain, H. Ochiai, D. Fall, and Y. Kadobayashi, "LSTM-based network attack detection: Performance comparison by hyper-parameter values tuning," in *Proc. 7th IEEE Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)/6th IEEE Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, Aug. 2020, pp. 62–69.

[22] S. Nayyar, S. Arora, and M. Singh, "Recurrent neural network based intrusion detection system," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Jul. 2020, pp. 136–140.

[23] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescape, "A hierarchical hybrid intrusion detection approach in IoT scenarios," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–7.

[24] S. Zavrak and M. Iskefiyeli, "Anomaly-based intrusion detection from network flow features using variational autoencoder," *IEEE Access*, vol. 8, pp. 108346–108358, 2020.

[25] K. Sadaf and J. Sultana, "Intrusion detection based on autoencoder and isolation forest in fog computing," *IEEE Access*, vol. 8, pp. 167059–167068, 2020.

[26] Z. Zohrevand and U. Glässer, "Should I raise the red flag? A comprehensive survey of anomaly scoring methods toward mitigating false alarms," 2019, *arXiv:1904.06646*. [Online]. Available: http://arxiv.org/abs/1904.06646

[27] K. Al Jallad, M. Aljnidi, and M. S. Desouki, "Anomaly detection optimization using big data and deep learning to reduce false-positive," *J. Big Data*, vol. 7, no. 1, pp. 1–12, Dec. 2020.

[28] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," 2018, *arXiv:1802.09089*. [Online]. Available: http://arxiv.org/abs/1802.09089

[29] I. Benmessahel, K. Xie, and M. Chellal, "A new evolutionary neural networks based on intrusion detection systems using multiverse optimization," *Int. J. Speech Technol.*, vol. 48, no. 8, pp. 2315–2327, Aug. 2018.

[30] J. Ku, B. Zheng, and D. Yun, "Intrusion detection based on self-adaptive differential evolutionary extreme learning machine," in *Proc. Int. Conf. Comput. Netw., Electron. Autom. (ICCNEA)*, Sep. 2017, pp. 94–100.

[31] J. Gao, S. Chai, B. Zhang, and Y. Xia, "Research on network intrusion detection based on incremental extreme learning machine and adaptive principal component analysis," *Energies*, vol. 12, no. 7, pp. 1–17, 2019.

[32] X. Yuan, R. Wang, Y. Zhuang, K. Zhu, and J. Hao, "A concept drift based ensemble incremental learning approach for intrusion detection," in *Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Social Comput. (CPSCom), IEEE Smart Data (SmartData)*, Jul. 2018, pp. 350–357.

[33] S. Paul. *Beginner's Guide to Feature Selection in Python*. Accessed: Feb. 18, 2021. [Online]. Available: https://www.datacamp.com/community/tutorials/feature-selection-python

[34] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Jan. 2003.

[35] U. Khurana, H. Samulowitz, and D. Turaga, "Feature engineering for predictive modeling using reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3407–3414.

[36] S. H. To. *Chi-Square Statistic: How to Calculate it/Distribution*. Accessed: Feb. 19, 2021. [Online]. Available: https://www.statisticshowto.com/probability-and-statistics/chi-square/

[37] T.-T. Nguyen, J. Z. Huang, and T. T. Nguyen, "Unbiased feature selection in learning random forests for high-dimensional data," *Sci. World J.*, vol. 2015, pp. 1–18, Dec. 2015.

[38] N. Titov. *Lightgbm*. Accessed: Feb. 20, 2021. [Online]. Available: https://lightgbm.readthedocs.io/en/latest/Features.html#references

[39] I. Ilievski, T. Akhtar, J. Feng, and C. A. Shoemaker, "Efficient hyperparameter optimization for deep learning algorithms using deterministic RBF surrogates," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1–8.

[40] M. Feurer and F. Hutter, *Hyperparameter Optimization*. Cham, Switzerland: Springer, 2019, pp. 3–33.

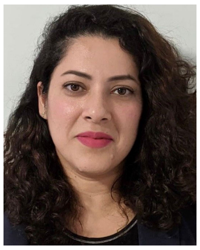
[41] I. Loshchilov and F. Hutter, "CMA-ES for hyperparameter optimization of deep neural networks," 2016, *arXiv:1604.07269*. [Online]. Available: http://arxiv.org/abs/1604.07269

[42] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you? Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1135–1144.

[43] D. Civin. *Explainable AI Could Reduce the Impact of Biased Algorithms*. Accessed: Feb. 21, 2021. [Online]. Available: https://venturebeat.com/2018/05/21/

[44] G. Brown, "Ensemble learning," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA, USA: Springer, 2010, pp. 312–320, doi: 10.1007/978-0-387-30164-8_252.

[45] V. Bolón-Canedo and A. Alonso-Betanzos, "Ensembles for feature selection: A review and future trends," *Inf. Fusion*, vol. 52, pp. 1–12, Dec. 2019.
[46] University of Virginia Library. *Understanding Q-Q Plots*. Accessed: Mar. 1, 2021. [Online]. Available: https://data.library.virginia.edu/understanding-q-q-plots/
[47] Codecademy. *Normalization*. Accessed: Mar. 1, 2021. [Online]. Available: https://www.codecademy.com/articles/normalization
[48] A. Bhandari. *Feature Scaling for Machine Learning*. Accessed: Mar. 2, 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/04/page/3/
[49] *Preprocessing Data*. Accessed: Mar. 2, 2021. [Online]. Available: https://scikit-learn.org/stable/modules/preprocessing.html
[50] *Slowhttptest Package Description*. Accessed: Mar. 10, 2021. [Online]. Available: https://tools.kali.org/stress-testing/slowhttptest
[51] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, 2018, pp. 108–116.
[52] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features," in *Proc. 3rd Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, 2017, pp. 253–262.
[53] P. Toupas, D. Chamou, K. M. Giannoutakis, A. Drosou, and D. Tzovaras, "An intrusion detection system for multi-class classification based on deep neural networks," in *Proc. 18th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2019, pp. 1253–1258.
[54] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, "BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset," *IEEE Access*, vol. 8, pp. 29575–29585, 2020.
[55] *CICFlowmeter (Formerly ISCXFlowMeter)*. Accessed: Jun. 18, 2019. [Online]. Available: http://www.netflowmeter.ca/netflowmeter.html
[56] *Sklearn.svm.linearSVC*. Accessed: Mar. 20, 2021. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC
[57] EpistasisLab at UPenn. *A Python Automated Machine Learning Tool That Optimizes Machine Learning Pipelines Using Genetic Programming*. Accessed: Apr. 10, 2021. [Online]. Available: https://github.com/EpistasisLab/tpot
[58] N. Bakhareva, A. Shukhman, A. Matveev, P. Polezhaev, Y. Ushakov, and L. Legashev, "Attack detection in enterprise networks by machine learning methods," in *Proc. Int. Russian Automat. Conf. (RusAutoCon)*, Sep. 2019, pp. 1–6.
[59] B. J. M. Abma, "Evaluation of requirements management tools with support for traceability-based change impact analysis," M.S. thesis, Dept. Elect. Eng., Univ. Twente, Enschede, The Netherlands, 2009.



ULYA SABEEL received the Bachelor of Technology degree in information technology from Bharath University, India, in 2011, and the Master of Technology degree in computer science and engineering from Amity University, India, in 2013. She is currently pursuing the Ph.D. degree in computer science with the University of Ontario Institute of Technology (Ontario Tech University), Canada.

Since 2018, she has been working as a Research Assistant with the Advanced Networking Technology and Security (ANTS) Research Laboratory and a Teaching Assistant for the Faculty of Business and IT, Ontario Tech. She is currently working as a part-time Professor in business analytics and insights with the Centennial College, Toronto. Prior to this, she has more than five years research and teaching experience, as an Assistant Professor with Amity University. She has authored multiple international conference and journal articles in her research field. Her research interests include network security, computer networks, applied machine learning, and deep learning.

Ms. Sabeel is an active member of many international professional organizations. Her Ph.D. research work has been awarded the prestigious Ontario Graduate Scholarship, in 2019, 2020, and 2021, consequently.

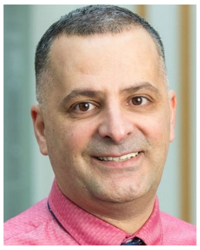
SHAHRAM SHAH HEYDARI (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in electronic engineering from the Sharif University of Technology, Iran, the M.A.Sc. degree from Concordia University, Montreal, and the Ph.D. degree from the University of Ottawa, Canada.

He is currently an Associate Professor with the Faculty of Business and Information Technology, University of Ontario Institute of Technology (Ontario Tech University), Canada, and the Co-Director of the Ontario Tech Advanced Networking Technology and Security (ANTS) Research Laboratory. Before joining Ontario Tech, in 2007, he was a System Designer and member of the Scientific Staff at Nortel Networks, where he worked on element management in ultrahigh-speed IP/MPLS routers, performance modeling of automatically switched optical networks (ASON), and proprietary voice-over-IP transport control protocols. His main research interests include network design and planning, software-defined networking, applications of artificial intelligence in network management, and network Quality of Experience (QoE).

KHALID ELGAZZAR (Senior Member, IEEE) received the B.Sc. degree in computer and communication engineering from Alexandria University, Egypt, in 1995, the M.Sc. degree in computer engineering from Arab Academy for Science and Technology, Egypt, in 2007, and the Ph.D. degree in computer science from Queen's University, Canada, in 2013.

He is currently the Canada Research Chair in Internet of Things (IoT) and an Assistant Professor with the Department of Electrical, Computer, and Software Engineering, Ontario Tech, where he is also the Founder and Director of the IoT Research Laboratory. Before joining Ontario Tech, he was an Assistant Professor with the University of Louisiana at Lafayette and a Research Associate with the Carnegie Mellon School of Computer Science. He is an expert in the areas of the Internet of Things (IoT), computer systems, real-time data analytics, and mobile computing.

Dr. Elgazzar received many research awards and several recognitions and best paper awards at top international venues. He is an active volunteer in technical program committees and organizing committees in both IEEE and ACM events. He is an associate editor for several ACM/IEEE journals in the areas of mobile computing and the IoT.



KHALIL EL-KHATIB (Member, IEEE) received the bachelor's degree in computer science from the American University of Beirut, the Master of Computer Science degree from McGill University, Montreal, Canada, and the Ph.D. degree from the University of Ottawa, Canada.

He is currently working as a Professor in information security and the Director of the Faculty of Business and Information Technology, Institute for Cybersecurity and Resilient Systems, Ontario Tech. Before joining Ontario Tech, he worked as an Assistant Professor with the University of Western Ontario. His research interests include big data and security analytics, security and privacy issues in wireless sensor networks, smart cities and communities, mobile wireless *ad hoc* networks and vehicular networks, smart grid security, biometrics, and ubiquitous computing.