VIETNAM NATIONAL UNIVERSITY HANOI
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Vo Van Hoang

ENHANCING INTRUSION DETECTION PERFORMANCE BY DATA AUGMENTATION, PARALLEL ENSEMBLE INFERENCE, AND FLOW SENSING STRATEGY

Major: Information Systems
Code: 9480104

SUMMARY OF THE PHD DISSERTATION
OF INFORMATION SYSTEMS

Supervisors:
Associate Professor Nguyen Ngoc Hoa
Associate Professor Nguyen Ngoc Tu

Ha Noi - 2025

The thesis was completed at: **University of Engineering and Technology, Vietnam National University, Hanoi**.

Supervisor:

- **Associate Professor, PhD Nguyen Ngoc Hoa**
- **Associate Professor, PhD Nguyen Ngoc Tu**

Reviewer: **Associate Professor, PhD Nguyen Linh Giang**

Reviewer: **Associate Professor, PhD Nguyen Long Giang**

Reviewer: **Professor, PhD Nguyen Hieu Minh**

The dissertation is going to be defended before a National University-level Committee in University of Engineering and Technology, Vietnam National University, Hanoi, on December, 2025.

PhD STUDENT **SUPERVISORS**

Vo Van Hoang Nguyen Ngoc Hoa Nguyen Ngoc Tu

CONFIRMATION OF THE TRAINING UNIVERSITY

Thesis can be found at:

- National Library of Vietnam.
- Library Information Center , Vietnam National University, Hanoi.

Introduction

Motivation

Cyberattacks are becoming more sophisticated, exposing the limitations of traditional signature- and rule-based systems, which struggle with zero-day exploits, polymorphic malware, and high false alarms. While ML and DL provide powerful alternatives, they face challenges such as noisy and imbalanced data, feature redundancy, and poor interpretability. Moreover, many models fail to meet real-time and scalability requirements in practice. To address these gaps, this dissertation proposes a roadmap unifying data balancing, feature refinement, model optimization, and multimodel inference to build accurate, resilient, explainable, and deployable AI-based intrusion and malware detection systems.

Research Challenges

This dissertation focuses on the following major challenges:

1. Challenge 1: Cybersecurity datasets are heavily imbalanced, with the vast majority of samples belonging to benign traffic or a few common attack types, while rare but dangerous threats (e.g., infiltration, exfiltration, and zero-day attacks) are underrepresented. This leads to biased model learning and poor detection of minority attacks.

2. Challenge 2: Achieving high accuracy and low false positive rates in AI-powered intrusion detection systems, while maintaining overall system performance and interpretability, remains a persistent challenge.

3. Challenge 3: For AI-powered intrusion detection systems to be operationally viable, they must process large volumes of traffic at wire speed with minimal delay. However, the computational complexity of machine learning models often hinders real-time deployment.

Research Objectives

- Objective 1: An overview of cyberattacks and the techniques used by hackers to carry out such attacks. Research intrusion and malware detection techniques and analyze the advantages and disadvantages of each method. Evaluate the results of the latest research related to the problem of intrusion detection.

- Objective 2: We propose a augmentation dataset method that aims to improve the quality of minority attack samples, select the most representative samples from the majority classes; minimize training noise by identifying important features within the dataset.

- Objective 3: Traditional intrusion detection methods often struggle with generalization and robustness against novel or adversarial attacks. This objective aims to integrate neural networks with boost models through soft voting and stacking strategies.

- Objective 4: AI-based detection systems often suffer from inference latency and limited scalability. This objective aims to design a lightweight, high-throughput detection architecture with support for flow-based sensing and parallel ensemble inference.

Research Scope

To achieve the objectives of this dissertation, we focus on the following key areas:

1. Research data structures and class imbalance in intrusion detection datasets and study machine learning and deep learning models for their effectiveness.
2. Research focuses on building lightweight high-throughput detection architectures suitable for real-time deployment in large-scale networks.

Research Methodologies

This dissertation employs a systematic and layered research methodology, as outlined below:

- **Theoretical Methodology:** We conduct a comprehensive survey, synthesis, and evaluation of previous research relevant to intrusion detection and malware classification.
- **Experimental Methodology:** The proposed frameworks and algorithms are empirically validated through extensive experiments on multiple benchmark datasets, including public and custom-prepared corpora.

Research Contributions

The key contributions are as follows:

1. We propose methods for augmentation dataset and feature set optimization. The approach integrates adversarial sample generation to enrich the minority class and employs filtering techniques to retain only semantically meaningful samples from the majority class.
2. We propose an integrated ensemble architecture that combines neural networks with boosting classifiers using both soft voting and stacking strategies. This hybrid framework leverages the complementary strengths of deep learning and tree-based models to enhance detection accuracy, robustness, and interpretability.
3. We design and implement NetIPS, a lightweight and real-time intrusion detection and prevention architecture optimized for large-scale network environments.

Thesis Structure

This dissertation is structured into four chapters:

- Chapter 1 This chapter presents essential background knowledge in intrusion and malware detection, with an emphasis on machine learning, deep learning, and ensemble techniques.
- Chapter 2 proposes augmentation dataset methods for machine learning, focusing on addressing the imbalance between minority and majority classes in the dataset.
- Chapter 3 focuses on improving machine learning models to enhance performance. The chapter proposes combining and mutually reinforcing different types of models to increase intrusion detection effectiveness and system robustness.
- Chapter 4 proposes a practical deployment approach for intrusion detection systems in large-scale networks. A comprehensive process for intrusion detection is introduced that integrates both signature-based and behavior-based analysis, along with execution and sampling strategies.

1 Preliminaries and Literature Reviews

1.1 Fundamental Concepts

1.1.1 Intrusion Detection System

Intrusion Detection Systems (IDS) are critical for monitoring network and host activities, with NIDS analyzing traffic flows and HIDS focusing on endpoint behavior. Detection approaches range from signature-based methods, effective only for known threats, to anomaly-based systems that detect novel intrusions but suffer high false positives. Recent ML/DL techniques improve adaptability and malware detection via static and dynamic analysis, yet challenges remain in data imbalance, obfuscation, and real-time deployment.

1.1.2 Common Types of Network Attacks

The summary of common network attack types show as Table 1.1.

1.1.3 Machine Learning in Cybersecurity

Machine learning (ML) has become a key enabler for modern cybersecurity by learning complex patterns and adapting to evolving threats, surpassing the limitations of traditional rule-based detection. ML techniques are used across tasks such as intrusion detection, malware classification, phishing detection, and behavioral analysis. Despite their strengths, ML/DL models face challenges: data imbalance, limited generalization, lack of interpretability, and real-time performance constraints.

1.1.4 Class Imbalance in Cybersecurity Dataset

Class imbalance is a major challenge in cybersecurity datasets, where benign samples vastly outnumber malicious ones, and rare but critical attack types are often underrepresented. This skews ML model performance, leading to poor recall on minority attack classes and high false negative rates.

1.1.5 Ensemble Learning in Intrusion Detection

Ensemble learning combines multiple base models to achieve better predictive performance, making it highly effective for cybersecurity where attack patterns are diverse and evolving. By integrating models through techniques like bagging, boosting, voting, and stacking, ensembles improve accuracy, generalization, and resilience to adversarial evasion.

1.2 Approaches to Threat Detection

1.2.1 AI-powered Intrusion Detection

AI-based intrusion detection leverages machine learning (ML) and deep learning (DL) models to classify network traffic flows as benign or malicious. Gradient boosting methods such as XGBoost and GBM have proven effective in this domain by sequentially minimizing prediction errors and modeling complex attack

Table 1.1: Summary of Common Network Attack Types

Attack Type	**Technique**	**Impact**	**Detection**
Denial-of-Service (DoS/DDoS)	Traffic floods, amplification	Service unavailability	Rate limiting, filtering
Scanning & Enumeration	Port/vulnerability scans	Reconnaissance	IDS, anomaly detection
Spoofing	IP/ARP/DNS falsification	Evasion, redirection	Authentication, ARP/DNS security
Man-in-the-Middle (MitM)	Interception, SSL stripping	Data theft, manipulation	Encryption, certificate pinning
Sniffing/ Eavesdropping	Passive/active traffic capture	Credential leakage	TLS, VPN
Replay/Session Hijacking	Packet replay, session ID theft	Unauthorized access	Token/session management, TLS
Malware Propagation	Worms, trojans, ransomware	Compromise, data loss	Antivirus, sandboxing
Phishing/Social Engineering	Deceptive messages, psychological tricks	Credential theft, initial access	User training, email filtering
SQLi/XSS/CSRF	Web input manipulation	Data theft, defacement	Input validation, WAF
APT	Multi-stage, stealthy infiltration	Espionage, long-term theft	Behavior analytics, EDR
Supply Chain	Third-party compromise	Widespread breach	Vendor management, code review
Insider Threat	Privileged misuse, data exfiltration	Confidentiality breach	Monitoring, least privilege, DLP

patterns. Deep neural networks (DNNs) excel at capturing nonlinear relationships in traffic data through multi-layer representations. Each ML/DL model offers unique strengths; combining them through ensemble learning improves detection accuracy and resilience against adversarial attacks.

1.2.2 AI-powered Malware Detection

For AI-powered malware detection, define D as the dataset composed of pairs (v, y), where v is the representation of a feature vector of a PE file and $y \in \{0, 1\}$ is the associated label (e.g., 0 for benign, 1 for malware), respectively. Thus, $D = (v_1, y_1), (v_2, y_2), ..., (v_M, y_M)$ with M representing the total number of samples. The problem is to train a generalized AI model $f : R^n \implies 0, 1$ on the dataset D such that, for any new PE file, its feature vector v is mapped to a predicted label $\hat{y} = f(v)$. The goal is to maximize the accuracy of f while generalizing well beyond the training dataset, thus enabling the reliable detection of malware in unseen PE files.

1.2.3 Handling Imbalanced Datasets

Most datasets suffer from severe class imbalance, with benign traffic dominating and attack flows underrepresented. This imbalance degrades both model training and prediction accuracy. To address this, data balancing techniques are employed such as undersampling the majority class and oversampling the minority class.

1.3 Related Work

1.3.1 Deep and Boosting Learning for Intrusion Detection

A summary of these methods and their performance is provided in Table 1.2.

Table 1.2: Summary of Related Works based Intrusion Detection

Method	Venue	Approach	Dataset	Acc(%)
RF+ miniVGGNet [?]	IEEE Access 2020	Combination of K-Means and ENN to balance dataset then RF+ miniVGGNet to detect intrusions.	NSL-KDD, CIC-IDS2018	82.84, 96.99
WGAN+ LightGBM [?]	Computer Science 2021	Applying WGAN-GP for data generation on minority class samples and using LightGBM for the classification.	NSL-KDD, CIC-IDS2018	99.00, 96.00
MMM-RF [?]	Computer & Security 2022	Use CFS to analyze network traffic, T-SNE to minimize data dimension, and SMOTE to imbalance the CSE-CIC-IDS2018 dataset.	CIC-IDS2018	99.98
CNN, DBNs, LSTM [?]	Computers and Electrical Engineering 2022	Transforms the traffic flow features into waves and utilizes advanced audio/speech recognition deep-learning-based techniques to detect intruders.	CIC-IDS2017, NSL-KDD	99.21, 84.82
CNN+LSTM [?]	Digital Communications and Networks 2023	Used SMOTE to balance abnormal traffic, CNN to extract deep features, then CNN-LSTM to detect intrusions.	UNSW.NB15, CIC-IDS2017, NSL-KDD	99.21, 99.32, 98.45
FFO+PNN [?]	Alexandria Engineering Journal 2023	Used the FFO technique to extract features and PNN to classify categories.	NSL-KDD	98.99
CNN+EQL [?]	Computer Communications 2023	Used CNN and the Attention mechanism mingle to form a CA Block focusing on local spatiotemporal feature extraction and EQL v2 to increase the minority class weight and balance the learning attention on minority classes.	UNSW.NB15, NSL-KDD, CIC-IDS2017, CIC-DDoS2019	89.39, 99.77, 99.88, 99.58
PIGNUS [?]	Computer & Security 2023	Use Auto Encoders to select optimal features and Cascade Forward Back Propagation Neural Network for classification and attack detection.	NSL-KDD	99.02

1.3.2 Deep and Boosting Learning for Malware Detection

A summary of these methods and their performance is provided in Table 1.3.

Table 1.3: Summary of Related Works based Malware Detection

Method	Venue	Approach	Dataset	Acc(%)
CNN [?]	Distributed Computing and Artificial Intelligence 2021	The method in this study converts binary files into grayscale images to detect malware. The model also integrates an attention mechanism to identify suspicious parts within the file.	EMBER 2018	94.00
DNN [?]	Procedia Computer Science 2022	This method builds an improved offensive generative model based on GANs to strengthen the current DNN-based system.	EMBER 2018	97.42
CNN [?]	International Journal of Computer Network and Information Security 2022	This method employs feature extraction, data standardization, and data cleaning techniques to address imbalances and impurities within the dataset.	EMBER 2017 & 2018	97.53, 94.09
EII-MBS [?]	Computers & Security 2022	This technique finds patterns in how instructions relate to each other and turns this information into vector representations to classify malware families.	BODMAS	99.29
XGB-CATB-EXT [?]	Computer, Material & Continua 2023	The technique in this study utilizes a model combining supervised and unsupervised learning to improve malware detection. Specifically, k-means clusters the data before a set of ML algorithms classifies it.	EMBER 2018	96.77
MD-ADA [?]	Computers & Security 2024	This approach combines CNN-based image embeddings and adversarial domain adaptation (using GANs) to classify malware.	BODMAS	99.29
FCG-MFD [?]	Journal of Network and Computer Applications 2025	This method uses function call graphs and node2vec along with ideas from NLP to help classify malware families.	BODMAS	99.28

1.3.3 Data Augmentation

To resolve imbalanced dataset, several techniques are employed such as undersampling the majority class and oversampling the minority class, etc.

1.4 Dataset Collection

This dissertation employs several widely used public datasets: The CSE-CIC-IDS2018; NSL-KDD; EMBER2017 and EMBER2018 and BODMAS datasets.

1.5 Evaluation Metrics

We use standard metrics computed from the confusion matrix, such as: Acc; $Prec$; Rec; etc.

1.6 Research Gaps and Approach Direction

- Research Gap 1: Most real-world intrusion detection datasets suffer from severe class imbalance, where minority attack classes are underrepresented and difficult to learn.

  Approach Direction (chapter 2): To address these limitations, we propose augmentation dataset methods that enhances both the quantity and quality of training dataset, optimizing feature space.

- Research Gap 2: Although recent studies have proposed various approaches to optimize machine learning models for intrusion detection, model optimization remains a persistent challenge in machine learning applications.

  Approach Direction (chapter 3): We design a mutual deep+boosting ensemble inference pipeline that leverages the complementary strengths of diverse models to enhance overall performance and reduce vulnerability to model poisoning.

- Research Gap 3: Despite recent advances, most IDS models remain unsuitable for high-throughput environments due to computational bottlenecks, static detection logic, and lack of adaptive flow control. Traditional detection frameworks are unable to meet real-time latency constraints or scale to modern enterprise or ISP-level networks.

  Approach Direction (chapter 4): We propose a scalable and low-latency intrusion prevention system called NetIPS, built upon parallelized deep and boosting models integrated with flow-sensing optimization and sandbox analysis.

1.7 Summary

In summary, this chapter has identified the key research challenges and objectives in intrusion and malware detection and outlined the main scientific contributions and research roadmap of the dissertation. The mapping between these contributions and the corresponding technical chapters has also been presented, providing a clear structure for the remainder of this work.

2 Enhancing AI-powered Intrusion Detection with Data Augmentation and Feature Optimization

2.1 Problem Statement

We introduces two complementary solutions: (i) an adaptive data augmentation pipeline that compresses majority classes and generates realistic minority samples to improve balance and diversity, and (ii) a SHAP-based Optimized Feature Set (OFS) method that prunes irrelevant features, enhances interpretability, and reduces computational overhead.

2.2 Approach Direction

To address the two major challenges of class imbalance and feature redundancy in intrusion detection datasets, we introduce augmentation dataset methods aimed at enhancing the learning capacity of AI models in practical cybersecurity contexts. The proposed approach is designed to simultaneously address the problem of insufficient dataset in minority classes, select high-quality samples from majority classes, and identify valuable features in datasets with large numbers of features, issues that are commonly encountered in real-world datasets.

2.3 Training Dataset Augmentation

2.3.1 Difficulty-Aware-based Data Augmentation

We propose a method based on the concept of the DSSTE algorithm proposed by [?] to augment the training dataset. Our algorithm is named AugDS and is shown in 2.1.

2.3.2 AWGAN-based Data Augmentation

To solve the issue of the unbalanced dataset in IDS, our augmented WGAN method, AWGAN, generates realistic samples for minority classes using WGAN, the AWGAN is depicted in Figure 2.1a and is described formally in 2.2.

2.4 Feature set Optimization

2.4.1 Feature Extraction and Cleaning

Feature extraction and cleaning are essential to reduce computational cost and avoid noisy or duplicate data that cause overfitting. Our approach removes null or duplicate records, ensuring only unique and relevant entries remain in the dataset.

2.4.2 Feature Vectorizing

Raw data, often in JSON format, must be transformed into numerical vectors for AI model training. To achieve this, we apply feature hashing, which maps tokens into fixed-length vectors while preserving data characteristics [?]. Using kernel and sign hash functions, features are vectorized, normalized, and stored

Algorithm 2.1 AugDS: Build the Augmented Dataset

Input: F - Raw Dataset, represented by a list of feature vectors; K - scaling factor
Output: T - Augmented Dataset;

$L \leftarrow ComputeLabels(F)$ ▷ Get all labels of dataset F
$F \leftarrow Normalize(F)$ ▷ normalizing all feature vectors
$ES = EditedNearestNeighbours(RT, |L|)$ ▷ determining the easy sets ES by finding L nearest neighbours samples
$DS = RT \setminus ES$ ▷ difficult set DS is the rest of RT
$Majors, Minors \leftarrow ComputeMajMin(DS)$
$S_{maj} \leftarrow \emptyset, S_{min} \leftarrow \emptyset$
for each $M \in Majors$ **do** ▷ Compression Step
  $C \leftarrow Clustering(M, |L|)$ ▷ computing the centroids C of $|L|$ clusters by using KMeans algorithm
  $M \leftarrow Compress(M, C, \tau)$ ▷ compressing majority samples using centroids C of L clusters
  $S_{maj} \leftarrow S_{maj} \cup M$
end for
for each $M \in Minors$ **do** ▷ Zooming Step
  for each $m \in range(K, K + \frac{number}{N_{S_{min}}})$ **do** ▷ Zooming Step, $N_{S_{min}}$ is number sample in S_{min}.
    $M \leftarrow Zoom(m)$ ▷ zoom range is $[1 - \frac{1}{K}, 1 + \frac{1}{K}]$ on both continuous and categorical features.
    $S_{min} \leftarrow S_{min} \cup M$
  end for
end for
$T = ES \cup S_{maj} \cup S_{min}$ ▷ synthese of new dataset T
return (T)

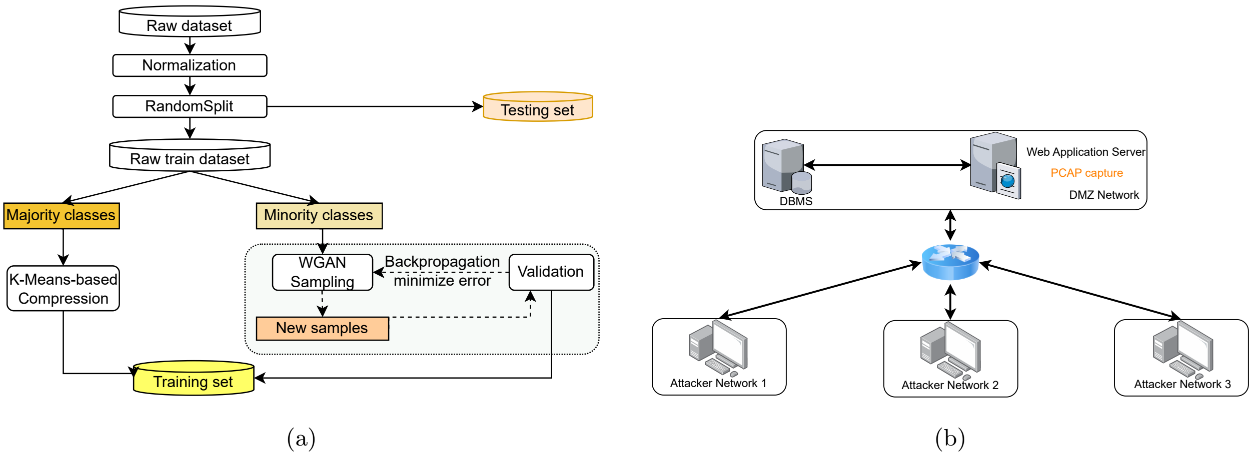


Figure 2.1: (a) AWGAN-based data augmentation framework; (b) SQL injection attack generation testbed.

in CSV. Finally, categorical attributes are encoded via label encoding and one-hot encoding (using Keras), ensuring compatibility with ML models such as GBM and neural networks.

2.4.3 Feature Normalization

Normalizing the features centers them around zero with a unit standard deviation, facilitating the learning process of the ML algorithm. This normalization technique helps speed up convergence and improve the model's overall performance.

2.4.4 SHAP-based Feature Set Optimization

Our method, Optimizing Features using SHAP (OFS), selects the most important subset of features from training data by combining model performance with explainability. The general pseudocode to optimize the set of features using SHAP is presented in 2.3.

2.5 Experiments and Evaluation

2.5.1 Dataset Preparation

- DS1:CSE-CIC-IDS2018 and NSL-KDD datasets.

  To augment SQL-injection detection, we also built a testbed system, as shown in Figure 2.1b, to add more detection ability. Finally, based on our dataset preparation process, we obtain two augmented

Algorithm 2.2 AWGAN: Create the Training & Testing Sets by Augmented WGAN

Input: F - Raw Dataset, represented by a list of feature vectors.
r - ratio between training and testing sets; default is 7:3.
τ - maximum samples in a label.
Output: T - Training Set; V - Testing Set.

$L \leftarrow GetLabels(F)$ ▷ Get all labels of dataset F.
$F \leftarrow Normalize(F)$ ▷ Normalize all feature vectors.
$(RT, V) \leftarrow SplitTrainTest(F, r)$ ▷ Split F randomly into the raw training set RT and testing set V with ratio of r.
$(S_{maj}, S_{min}) \leftarrow GetClasses(RT)$ ▷ Determine majority classes (S_{maj}) and minority classes (S_{min}) from RT
$T \leftarrow \emptyset$
for each $M \in S_{maj}$ **do** ▷ Compression each majority class
  $C \leftarrow Clustering(M, |L|)$ ▷ Compute the centroids C of $|L|$ clusters by using ENN
  $M \leftarrow Select(M, C, \tau)$ ▷ Compress majority samples using C of L clusters
  $T \leftarrow T \cup M$
end for
for each $M \in S_{min}$ **do** ▷ Generate samples for minority classes by WGAN
  while $|M| < \tau$ **do**
    $S \leftarrow WGAN\_Sampling(M)$ ▷ Generate new samples
    $M = Denoise(M, S)$ ▷ Eliminate noise samples
  end while ▷ Repeat until get enough samples τ.
  $T \leftarrow T \cup M$ ▷ Add realistic samples to T
end for
return (T, V)

Algorithm 2.3 OFS: **O**ptimizing **F**eature **S**et Using **SHAP**

Input: DS - dataset with the feature set F; M - m AI models; τ - threshold to drop features.

$X, y \leftarrow DS$ ▷ Get dataframes for features and labels
$X \leftarrow$ Normalize(X) ▷ Normalize all features to [0,1]
$FS \leftarrow \emptyset$ ▷ Init the feature set list.
for each $m \in M$ **do** ▷ Determine the feature importance for each AI model m.
  $AI \leftarrow m.fit(X, y)$ ▷ Train m using the dataset.
  if m is a boosting model **then**
    $shap\_values_m \leftarrow SHAP.TreeExplainer(m)$ ▷ Compute the SHAP values of all features based on decision tree model.
  else
    $shap\_values_m \leftarrow SHAP.DeepExplainer(m)$ ▷ Compute the SHAP values of all features based on DL model.
  end if
  $FS.push(shap\_values_M)$ ▷ Push the Shapley values of the model M into the list FS.
end for
$OFS \leftarrow \emptyset$
for each $f \in F$ **do**
  $shap\_values \leftarrow FS[f]$ ▷ Get SHAP values of feature f on all models M.
  if $shap\_values \geq \tau$ **then**
    $OFS \leftarrow OFS \cup f$ ▷ Consider f being important and add to OFS in the case of all its SHAP values $\geq \tau$.
  end if
end for

Output: OFS - Optimized Feature Set.

datasets DS1, illustrated in Table 2.1. Note that DS1 datasets will be comprehensively evaluated in chapter 3.

- DS2: We also selected CSE-CIC-IDS2018 and NSL-KDD to experimentally evaluate the effectiveness of 2.2. Finally, Table 2.1 summarizes the number of samples for each class of both datasets.
- DS3: EMBER2017, EMBER2018, and BODMAS, to experimentally evaluate the effectiveness of 2.3, the output constitute DS3.

  We use six thresholds: 0.1, 0.075, 0.05, 0.25, 0.01, and 0.001. For each threshold, features with SHAP values $\geq$ the chosen threshold are selected, shown as Figure 2.2 and Figure 2.4a. We found that the threshold of 0.025 gives the best result, shown as Figure 2.3.

2.5.2 Results and Evaluation

- S1: We thoroughly evaluate 2.1 on the DS1 to investigate its effectiveness in addressing class imbalance.
- S2: The AWGAN 2.2 is evaluated on DS2 to rigorously assess its ability to generate realistic and diverse synthetic samples for minority classes.
- S3: The OFS 2.3 is examined using the DS3, with a focus on static malware detection tasks.

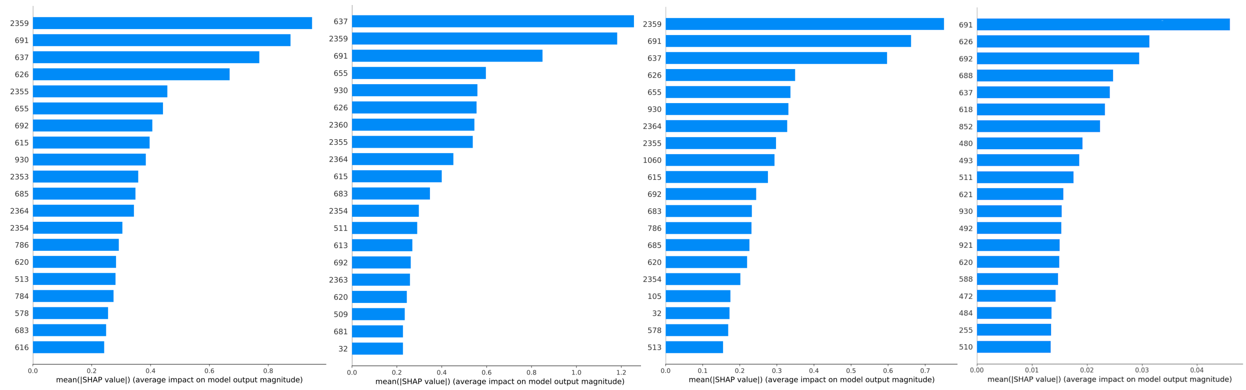
Table 2.1: Comparison of difficulty-aware augmentation (a) and AWGAN-based augmentation (b)

(a) Difficulty-Aware-based Data Augmentation

Class	Original	Train	Test
CSE-CIC-IDS2018			
Benign	4, 360, 029	20, 000	6, 000
Bot	282, 310	20, 000	6, 000
DDoS-HOIC	668, 461	20, 000	6, 000
DoS-GoldenEye	41, 455	20, 000	6, 000
DoS-Hulk	434, 873	20, 000	6, 000
Infiltration	160, 604	20, 000	6, 000
SQL-Injection	26, 797	20, 000	6, 000
DoS-SlowHTTPTest	19, 462	13, 623	4, 491
DoS-Slowloris	10, 285	14, 826	2, 373
DDoS-LOIC-UDP	1, 211	1, 588	279
BruteForce-Web	253	978	58
BruteForce-XSS	151	106	35
NSL-KDD			
Benign	61, 343	20, 000	6, 000
DoS	39, 927	20, 000	6, 000
Probe	8, 153	20, 000	1, 881
R2L	697	4, 467	161
U2R	36	36	8

(b) AWGAN-based Data Augmentation

Class	Original	Train	Test
CSE-CIC-IDS2018			
Benign	4, 360, 029	14, 000	6, 000
Infiltration	160, 604	14, 000	6, 000
Bot	282, 310	14, 000	6, 000
DDoS-HOIC	668, 461	14, 000	6, 000
DoS-GoldenEye	41, 455	14, 000	6, 000
DoS-Hulk	434, 873	14, 000	6, 000
DoS-SlowHTTPTest	13, 067	14, 000	4, 082
DoS-Slowloris	6, 977	14, 000	2, 093
DDoS-LOIC-UDP	1, 120	14, 000	336
BruteForce-Web	261	14, 000	78
BruteForce-XSS	97	14, 000	29
SQL-Injection	53	14, 000	17
NSL-KDD			
Benign	61, 343	14, 000	6, 000
DoS	39, 927	14, 000	6, 000
Probe	8, 333	14, 000	2, 500
R2L	637	14, 000	191
U2R	40	14, 000	12



(a) XGB Feature Important Score (b) GBM Feature Important Score (c) CBT Feature Important Score (d) CNN Feature Important Score

Figure 2.2: SHAP-based Feature Important Scores on EMBER2018 Dataset

S1 Results

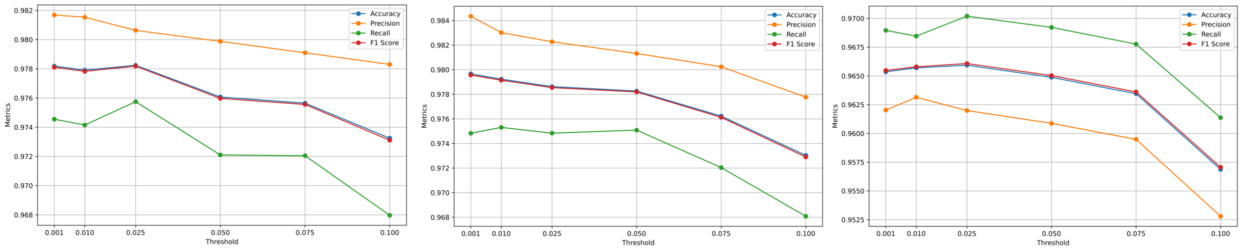
The summarized in Table 2.1. The effect is also visible in the t-SNE visualizations: before and after the balance shown in Figure 2.5a, and Figure 2.5b.

S2 Results

The Table 2.1 summarizes the number of samples per class in both datasets. The individual models shown as Table 2.2. Figure 2.6a show the original data before performing AWGAN-based augmentation, while Figure 2.6b illustrate the augmented training sets.

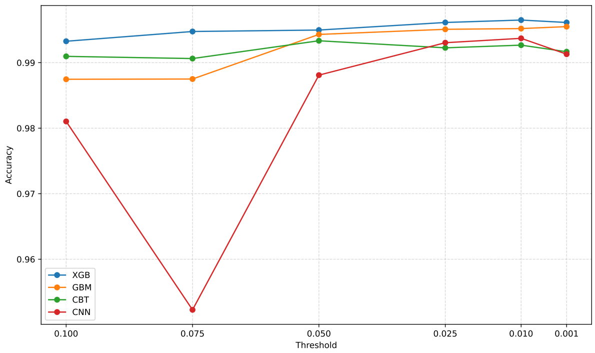
Table 2.2: Evaluation of AI models on WGAN-based Data Augmentation (%)

Metric	CSE-CIC-IDS2018					NSL-KDD				
	XGB	CBT	GBM	BME	DNN	XGB	CBT	GBM	BME	DNN
F1	99.77	99.92	**99.95**	99.77	97.75	**99.48**	99.21	**99.48**	**99.48**	98.00
Acc	99.76	99.92	99.96	**99.98**	97.54	99.49	99.22	**99.56**	99.43	98.07
Prec	99.83	99.93	99.96	**99.98**	98.20	**99.49**	99.21	**99.49**	99.41	98.03
Rec	99.76	99.92	99.96	**99.98**	97.54	**99.49**	99.22	**99.49**	99.43	98.07
FPR	**0**	**0**	0.03	**0**	0.13	0.67	1.27	**0.63**	0.77	1.22
FNR	**0**	0.01	**0**	**0**	1.37	0.37	0.39	**0.30**	0.32	2.26
AUC	**100**	**100**	99.99	99.99	98.69	**99.99**	99.98	**99.99**	99.89	99.85



(a) XGB (b) GBM (c) CBT

Figure 2.3: Threshold-based Performances on EMBER2018 Dataset



(a)

Metric	*CSE-CIC-IDS2018*			*NSL-KDD*		
	DNN	XGB	GBM	DNN	XGB	GBM
Acc	99.73	99.58	**99.74**	98.80	**99.66**	99.43
Prec	**99.80**	99.59	99.59	98.84	**99.66**	99.44
F1	**99.66**	99.58	99.58	98.80	**99.66**	99.43
Rec	**99.73**	99.58	99.58	**99.80**	99.66	99.43
AUC	99.96	**100**	**100**	99.84	**100**	99.92

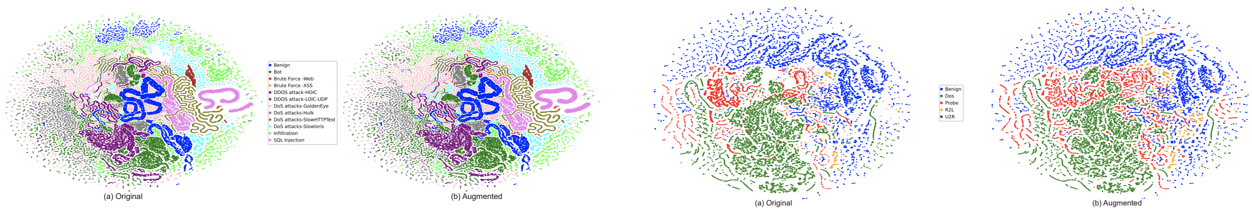
(b)

Figure 2.4: (a) Threshold-based performance on BODMAS dataset; (b) Evaluation results of AI models on difficulty-aware data augmentation.

S3 Results

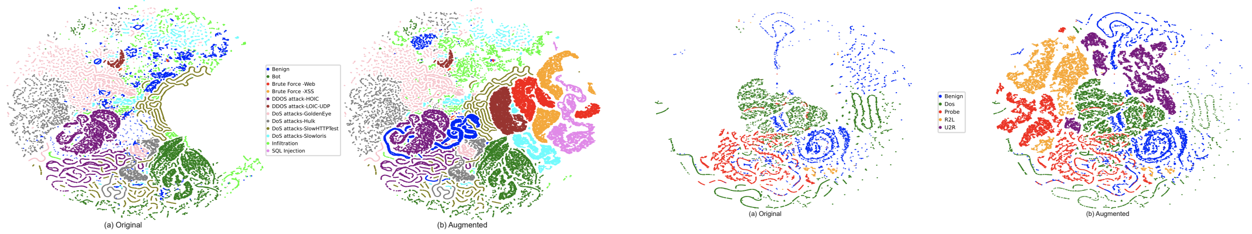
Table 2.3: Evaluation of AI models on Original Datasets(%)

Method	F1	Acc	Prec	Sens	FAR	FNR
EMBER2017 Evaluation						
XGB	99.16	99.16	99.16	99.16	0.84	0.84
CBT	**99.27**	**99.27**	**99.27**	**99.27**	**0.73**	**0.73**
GBM	98.67	98.67	98.67	98.67	1.33	1.33
CNN	95.95	96.04	93.72	95.95	3.35	4.05
EMBER2018 Evaluation						
XGB	97.63	97.63	97.63	97.63	2.37	2.37
CBT	97.19	97.19	97.19	97.19	2.81	2.81
GBM	**97.80**	**97.80**	**97.80**	**97.80**	**2.20**	**2.20**
CNN	94.03	94.02	94.16	94.02	5.97	5.98
BODMAS Evaluation						
XGB	98.71	98.69	99.68	97.75	0.32	2.25
CBT	**98.94**	**98.93**	99.88	**98.02**	0.12	**1.98**
GBM	98.90	98.89	**99.94**	97.88	**0.06**	2.12
CNN	98.90	98.89	99.87	97.96	0.13	2.04



(a) CSE-CIC-IDS2018 Training Set (b) NSL-KDD Training Set

Figure 2.5: Visualization results of Difficulty-Aware-based Data Augmentation



(a) CSE-CIC-IDS2018 Training Set (b) NSL-KDD Training Set

Figure 2.6: AWGAN-based visualization

Table 2.4: Evaluation of AI models based Features set Optimization (%)

Method	F1	Acc	Prec	Sens	FAR	FNR	F1	Acc	Prec	Sens	FAR	FNR
	BODMAS (4 features)						BODMAS (165 features)					
XGB	**89.76**	**90.78**	**85.19**	94.85	4.82	5.15	**99.28**	**99.39**	99.28	99.28	**0.61**	0.72
CBT	**89.76**	90.77	85.17	94.86	4.83	5.14	99.26	99.37	**99.29**	99.23	0.71	0.77
GBM	**89.76**	**90.78**	85.16	94.89	4.80	5.11	99.13	99.26	99.09	99.16	0.74	0.84
CNN	88.03	88.95	81.77	**95.34**	**4.66**	**4.66**	99.13	99.26	99.02	**99.24**	0.76	**0.74**
	EMBER2018 (170 features)						EMBER2018 (565 features)					
XGB	97.59	97.59	**97.84**	97.34	2.16	2.66	97.67	97.68	97.97	97.37	2.17	2.63
CBT	97.45	97.45	97.52	97.37	2.25	**2.53**	97.52	97.52	97.58	**97.46**	2.26	**2.54**
GBM	**97.85**	**97.86**	97.23	**97.47**	**2.13**	**2.53**	**97.88**	**97.89**	**98.34**	97.42	**2.16**	2.58
CNN	95.72	95.72	95.71	95.73	4.03	4.27	95.72	95.90	95.64	95.19	4.08	4.81

To evaluate the effectiveness of our feature optimization and data balancing strategies, we compare the model performance in the original datasets shown in Table 2.3 and in the optimized feature sets shown in Table 2.4 for EMBER2018 and BODMAS dataset.

2.6 Summary

These research results have been partially presented in published works, including three articles in respected journals (VVH-J2, VVH-J1, VVH-j3) and two conference paper (VVH-C2, VVH-C4), highlighting the novel and important contributions discussed in this chapter. Specifically, VVH-J2 presents an algorithm that addresses the challenge of class imbalance in network intrusion datasets through data compression and zooming techniques. VVH-J1 and VVH-C4 propose GAN-based methods capable of generating new samples to augment the minority class, thus mitigating data imbalance. VVH-j3 introduces a feature optimization approach to improve the quality of the dataset.

3 Enhancing AI-powered Intrusion Detection with Mutual Deep and Boosting Inference

3.1 Problem Statement

Single-model approaches often result in unstable performance and weak resilience to adversarial threats. To address this, we propose an ensemble framework combining deep and boosting models through soft voting and stacking, improving accuracy, robustness, and efficiency. This unified approach enhances detection of both common and sophisticated attacks, making it practical for real-world deployment.

3.2 Network Intrusion Detection via AI-Powered Deep Analysis

3.2.1 Direction Approach

We developed the SDAID solution, a comprehensive network intrusion detection approach that uses deep AI-powered analysis to identify anomalous behavior, as illustrated in Figure **??** and 3.1.

3.2.2 Network Traffic Flow Modeling

We propose using CICFlowMeter to perform the feature extraction task.

3.2.3 DNN-based Intrusion Detection Algorithm

Our DNN model is depicted in Figure **??**.

3.2.4 Boosting-based Intrusion Detection Algorithm

Boosting algorithms, such as XGBoost, build models sequentially where each new tree corrects the errors of the previous ones, achieving high accuracy and scalability. By tuning hyperparameters, we reduce overfitting and enhance generalization.

3.2.5 Hyperparameter Optimization

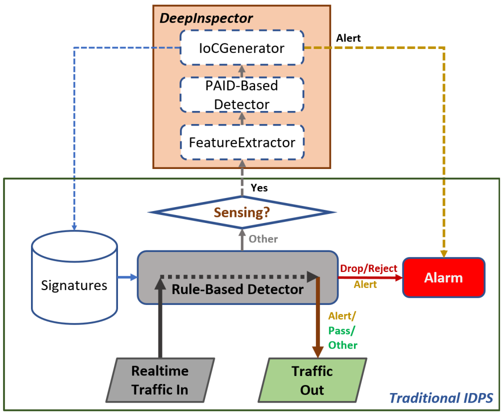
We select the model parameters based on a technique called Hyperparameter Optimization [**?**]. The optimal values are also illustrated in Table 3.1.

3.2.6 Experiments and Evaluation

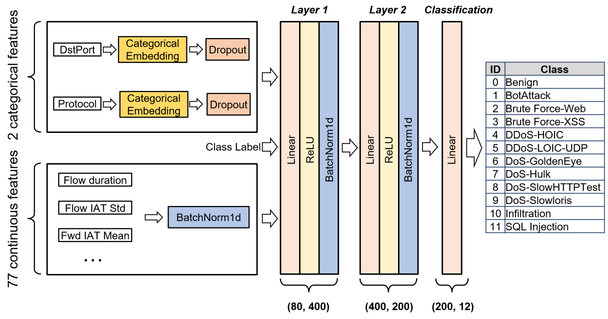
For the experimental environment, we use the setup presented in section 2.5.

To evaluate PAID 3.1, we perform the two scenarios described as follows:

- Scenario $S1$: CSE-CIC-IDS2018 dataset to train PAID.
- Scenario $S2$: We evaluate the PAID based on NSL-KDD.



(a)



(b)

Figure 3.1: (a) Network intrusion detection using AI-powered deep analysis; (b) DNN-based intrusion detection.

Algorithm 3.1 PAID: Perform an Ensemble Learning for AI-powered Intrusion Detection

Model: XGB - XGB trained model; DNN - DNN trained model; GBM - GBM trained model
Input: f - traffic flow.
Output: (msg, IoC) - (alert message; generated new IoC)

$R \leftarrow \emptyset$
$F \leftarrow CICFlowMeter(f)$ ▷ extract 83 features of traffic flow f
$Fin \leftarrow F \setminus [FlowID, SrcIP, SrcPort, Label]$ ▷ remove 4 unused features
$Cats \leftarrow [DstPort, Protocol]$ ▷ Categorical Variables
$Conts \leftarrow Fin \setminus Cats$ ▷ 77 Continuous Variables
Perform three processes P1,P2,P3:
P1: $dnn\_preds \leftarrow DNN.predict(Cats, Conts)$ ▷ perform the prediction using DNN model
P2: $xgb\_preds \leftarrow XGB.predict(Cats, Conts)$ ▷ perform the prediction using XGB model
P3: $gbm\_preds \leftarrow GBM.predict(Cats, Conts)$ ▷ perform the prediction using GBM model
Wait P1, P2, P3 finished.
$avgs \leftarrow (xgb\_preds + dnn\_preds + gbm\_preds)/3)$
$FC \leftarrow avgs.argmax(axis = 1)$ ▷ get the flow labels from 0 to 11
if $FC! = 0$ **then** ▷ classified as network attacks
    $msg \leftarrow Alert(FC)$ ▷ constitute an alert by using metadata from the flow f; set alert category being as label
    $R \leftarrow IoCGenerator(FC)$ ▷ generate a new IoC to handle the next similar flows
end if
return $msg; IoC$

S1 Results

The confusion matrix illustrates the results of our experiment performed with the PAID method, shown in Table 3.2a and the first part of Table 3.3.

S2 Results

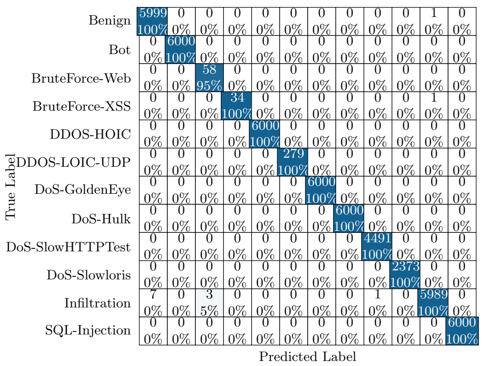
We consequently indicate this experiment results for the PAID as the confusion matrix shown in Table 3.2b and the second part of Table 3.3.

3.2.7 Comparison with SOTAs

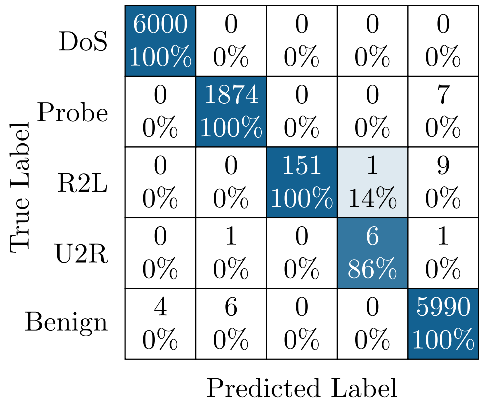
The comparison of intrusion detection performance between PAID and SOTA is summarized in Table 3.4.

Table 3.1: Hyperparameter Optimization

Model	Hyperparameter	Value	Optimal
DNN	Learning rate	[0.001, 1.0]	0.003
	Batch size	[16, 32, 48, 64, 96, 128]	64
	Epochs	[1, 2, ..., 15, 16]	5
	Layers	[[200, 100], ..., [1000, 500]]	[400, 200]
XGB	Learning rate	[0,1]	0.01
	n_estimators	[1,∞]	30
	max_depth	[0,∞]	6
GBM	Learning rate	[0,1]	0.02
	min_samples_leaf	[1,∞]	30
	max_depth	[0,∞]	9



(a) Confusion matrix of S1 evaluation.



(b) Confusion matrix of S2 evaluation.

Table 3.2: Confusion matrices of malware detection evaluations: (a) S1 dataset and (b) S2 dataset.

3.3 Malware Detection via Mutual Deep and Boosting Ensemble Learning

3.3.1 Approach Direction

We apply ensemble learning, including soft voting and stacking, to build binary classification models for malware detection. The method we propose in this study is called MDOB, an acronym for *"Enhancing Resilient and Explainable AI-Powered **M**alware **D**etection using Feature **O**ptimization and **M**utual **D**eep+**B**oosting Ensemble Learning."* Figure 3.2a illustrates the comprehensive architecture of our MDOB method.

3.3.2 Mutual Deep and Boosting Learning

We propose a mutual learning that integrates deep learning (DL) and gradient boosting models (GBM) for malware detection, leveraging AutoGluon for model selection, tuning, and optimization. By combining both, our system enhances accuracy, robustness, and adaptability against evolving threats.

Table 3.3: Performance Evaluation based Network Intrusion Detection

Metric	S1 (CSE-CIC-IDS2018)				S2 (NSL-KDD)			
	DNN	XGB	GBM	**PAID**	DNN	XGB	GBM	**PAID**
Acc	99.73	99.58	99.74	**99.97**	98.80	99.66	99.43	**99.69**
Prec	99.80	99.59	99.59	**99.97**	98.84	99.66	99.44	**99.69**
F1	99.66	99.58	99.58	**99.97**	98.80	99.66	99.43	**99.69**
Rec	99.73	99.58	99.58	**99.97**	99.80	99.66	99.43	**99.69**
AUC	99.96	**100**	**100**	**100**	99.84	**100**	99.92	99.99

Table 3.4: Comparison of PAID with other SOTA methods

Method	Acc	Prec	F1	Rec
CSE-CIC-IDS2018-based Evaluation				
PAID (our)	**99.97**	**99.97**	**99.97**	**99.97**
WGAN+IDR [**?**]	–	99	98	97
RANet [**?**]	96.73	–	96.59	96.73
Adaboost [**?**]	99.69	99.70	99.70	99.69
Autoencoder [**?**]	99.20	95.00	-	98.90
AUE [**?**]	97.90	98.00	98.00	98.00
DSSTE + miniVGGNet [**?**]	96.99	97.46	97.04	96.97
LSTM + AM + SMOTE [**?**]	96.20	96.00	93.00	96.00
NSL-KDD-based Evaluation				
PAID (our)	**99.69**	**99.69**	**99.69**	**99.69**
Autoencoder [**?**]	99.20	-	-	99.27
Multiple LSTM [**?**]	98.94	-	-	99.23
SMO [**?**]	96.20	-	-	-
RANet [**?**]	83.23	–	82.57	83.23
DNN [**?**]	78.50	81.00	76.50	78.50

3.3.3 Combination of Voting and Stacking Ensemble Learning

Algorithm 3.2 VSEL: Combination of Voting and Stacking Ensemble Learning

Input: $TD = \{(X^i, y^i)\}_{i=1}^N$ - training dataset with optimized features; $MS = \{M_1, M_2, ..., M_m\}$ - set of m base models; $model\_params$ - optimized hyperparameters of m AI models; K - number of folds for building meta training dataset (MTD).

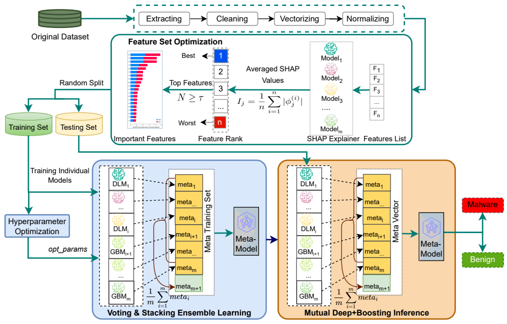
$MTD \leftarrow \emptyset$ ▷ Init MTD
$\{TD_1, TD_2, ..., TD_K\} \leftarrow Split(TD, K)$ ▷ Split the training dataset into K folds
for each fold $k \in 1..K$ **do**
  $TD_{\text{train}} \leftarrow TD \setminus TD_k$; $TD_{\text{val}} \leftarrow TD_k$ ▷ Use $K-1$ folds for training and 1 fold for validation
  for each $M_i \in MS$ **do**
    $M_i \leftarrow \text{Train}(M_i, TD_{\text{train}}, model\_params[M_i])$
  end for
  for each $(X, y) \in TD_{\text{val}}$ **do**
    $meta \leftarrow \emptyset$; $vote\_sum \leftarrow 0$ ▷ Create the meta-feature vector
    for each $M_i \in MS$ **do**
      $p_i \leftarrow M_i(X)$ ▷ Predict the probability for X using the trained base model M_i
      meta.push(p_i)
      $vote\_sum \leftarrow vote\_sum + p_i$
    end for
    $p_{\text{vote}} \leftarrow vote\_sum/m$ ▷ Calculate soft voting prediction from all base models
    meta.push(p_{vote}) ▷ Add soft voting result as an additional feature m+1 in the meta-layer
    $MTD.push(meta, y)$ ▷ Add the meta-feature vector and corresponding label to MTD
  end for
end for
$MM \leftarrow \text{AutoML.SelectBestModel}(MTD)$ ▷ Perform AutoML on MTD to select the best as the meta model
for each $M_i \in MS$ **do**
  $M_i \leftarrow \text{Train}(M_i, TD, model\_params[M_i])$ ▷ Retrain all base models on the whole training dataset to be used in final prediction
end for

Output: MS - n trained AI models; MM - trained meta model.

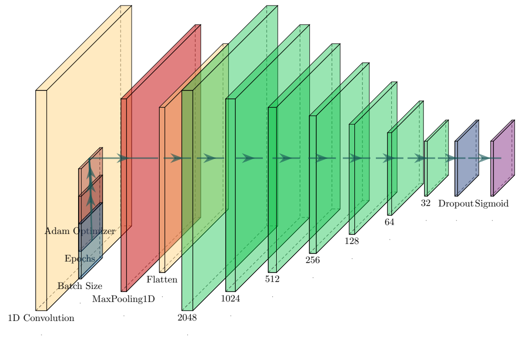
Our approach integrates voting and stacking learning to construct a more robust model using multiple AI-based classifiers. This process is illustrated in 3.2.

3.3.4 Hyperparameter Optimization

To optimize ML models in our approach, such as training individual models, we use Optuna [**?**]. This work is done through 3.3.



(a) Architecture of MDOB-based malware detection.



(b) Architecture of CNN model.

Figure 3.2: Model architectures for malware detection: (a) overall MDOB-based malware detection framework; (b) CNN-based malware classification module.

Algorithm 3.3 Hyperparameter Optimization using Optuna

Input: $model$ - AI model; $D_{train} = (X_{train}, y_{train})$ - training set; $D_{test} = (X_{test}, y_{test})$ - testing set; N_{trials} - number of trials; $T_{timeout}$ - optimization timeout; $params$ - list of hyperparameters.

function OBJECTIVE($trial$)
    $model\_params \leftarrow \{p_1, p_2, p_3, \ldots, p_n\}$ ▷ Initialize dictionary of hyperparameters for the model
    for $p \in params$ **do** ▷ Use Optuna to suggest hyperparameter values for each parameter p
        $model\_params[p] \leftarrow$ trial.suggest_⟨parameter_type⟩(“p", ⟨min_value⟩, ⟨max_value⟩)
    end for
    $clf \leftarrow$ model(**model_params) ▷ Instantiate model with current parameters
    $clf.fit(X_{\text{train}}, y_{\text{train}})$ ▷ Train model on training data
    $preds \leftarrow clf.predict(X_{\text{test}})$ ▷ Make predictions on testing data
    $metric \leftarrow$ performance_metric($y_{\text{test}}, preds$) ▷ Compute evaluation metric
    return metric
end function
Initialize an empty dictionary $opt\_params = \emptyset$
Optimize the objective function using Optuna:
$study \leftarrow optuna.create\_study(direction = \text{“}maximize\text{”})$
study.optimize(*objective*, n_trials=N_{trials}, timeout=T_{timeout})
$trial \leftarrow$ study.best_trial
$opt\_params \leftarrow trial.params$ ▷ Get optimized model parameters from the best trial
return opt_params

Output: opt_params - optimized hyperparameters.

3.3.5 Experiments and Evaluation

We conducted two scenarios to evaluate MDOB, as detailed below.

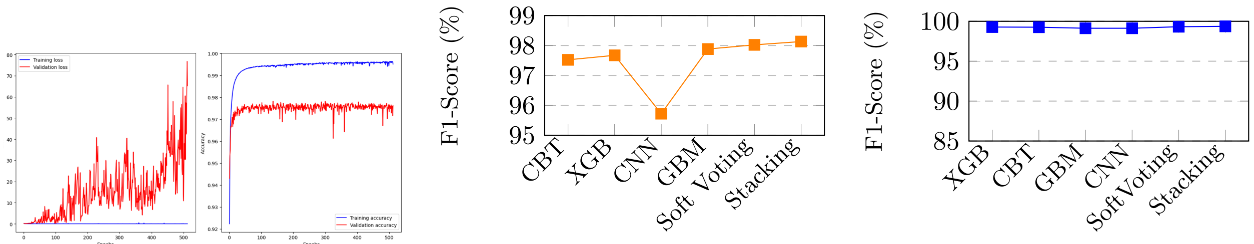
- Scenario $S1$: The focus is on using the EMBER2018 dataset to evaluate our proposed MDOB method.
- Scenario $S2$: We evaluated our proposed MDOB method using the BODMAS dataset.

S1 Results

Figure 3.3a shows the fine-tuning of the CNN model. Figure 3.3b compares the F1-score of different models on the EMBER2018 dataset using 565 features.

S2 Results

The results, summarized in Table 3.5. Figure 3.3c presents the F1-score performance on the BODMAS dataset using 165 features.



(a) (b) (c)

Figure 3.3: Comparative performances of CNN and ensemble models: (a) CNN training performance on EMBER2018 (565 features); (b) EMBER2018-based performance (565 features); (c) BODMAS-based performance (165 features).

Table 3.5: Evaluation of AI models based Malware Detection (%)

Learning	Method	F1	Acc	Prec	Sens	FAR	FNR	F1	Acc	Prec	Sens	FAR	FNR
		BODMAS (165 features)						EMBER 2018 (565 features)					
Baseline	XGB	99.28	99.39	99.28	99.28	0.61	0.72	97.67	97.68	97.97	97.37	2.17	2.63
	CBT	99.26	99.37	99.29	99.23	0.71	0.77	97.52	97.52	97.58	97.46	2.26	2.54
	GBM	99.13	99.26	99.09	99.16	0.74	0.84	97.88	97.89	98.34	97.42	2.16	2.58
	CNN	99.13	99.26	99.02	99.24	0.76	0.74	95.72	95.90	95.64	95.19	4.08	4.81
Mutual DLM+GBM	Voting	99.32	99.42	99.34	99.30	0.66	**0.70**	98.02	97.89	98.38	97.65	2.03	2.35
Mutual Voting+Stacking	**MDOB**	**99.37**	**99.46**	**99.48**	**99.26**	**0.54**	0.74	**98.13**	**98.14**	**98.58**	**97.68**	**1.93**	**2.32**

Table 3.6: Comparison of MDOB with SOTA Methods (%)

Method	Venue	Acc	Prec	F1	Sens
EMBER2018					
MDOB (our)	-	**98.14**	**98.58**	**98.13**	**97.68**
AutoML [?]	Computers & Security 2024	95.80	–	95.80	–
dualFFNN k-medoids [?]	Computers & Security 2023	98.02	–	–	–
Consensus [?]	CMC 2023	96.77	–	96.77	–
DL [?]	Telecom 2023	95.57	–	–	–
MLMD [?]	CAI 2023	97.42	–	–	–
DNN [?]	IJNIS 2022	94.09	90.14	88.66	88.85
BODMAS					
MDOB (our)	-	**99.46**	**99.48**	**99.37**	**99.26**
EII-MBS [?]	Computers & Security 2022	99.29	98.26	94.23	98.07
MD-ADA [?]	Computers & Security 2024	99.29	–	99.13	–
FCG-MFD [?]	JNCA 2025	99.28	–	99.14	–

3.3.6 Comparison with SOTAs

The comparison of malware detection implementations between MDOB and SOTA is summarized in Table 3.6.

3.4 Summary

In this chapter, we focus on improving the performance and robustness of intrusion and malware detection systems through ensemble learning and mutual interaction among machine learning models. Building on the enhanced datasets developed in chapter 2, this chapter addresses the limitations of individual models and proposes a unified framework that takes advantage of the complementary strengths of both deep learning and modern boosting algorithms.

4 Holistic Large-Scale AI-powered Intrusion Prevention with Flow Sensing Strategy and Parallel Ensemble Inference

4.1 Problem Statement

Traditional signature or standalone DL models are limited by latency and adaptability, often underperforming against evolving attacks in large-scale networks. To overcome these issues, we propose NetIPS, a proactive intrusion prevention system that integrates flow sensing, parallel inference, and lightweight user-space architecture.

4.2 Proposed Holistic Intrusion Detection Framework

4.2.1 Approach Direction

Our comprehensive intrusion detection approach uses deep AI-powered analysis to identify anomalous behavior and signatures of previous intrusions, namely APELID, as illustrated in Figure 4.1 and 4.1.

4.2.2 Parallel Ensemble Inference-based Intrusion Detection

Two ideas motivated our intrusion detection method: the ensemble learning approach and parallel computing. 4.2 shows our PELID algorithm.

4.2.3 Strategy for AI-powered real-time intrusion detection

For large-scale network traffic, the deep analysis certainly causes the stuck of IDPS. Therefore, we propose an efficient strategy to sense the traffic flows. Thus, we control the periodic deep analysis sampling strategy using 6 variables: DI_Cycle, DIC_Min, DIC_Max, and DI_Window, DIW_Min, DIW_Max.

4.2.4 Hunting Malware by Sandbox Approach

In order to improve the capability to detect malicious files transferred over the network, our proposed APELID solution is integrated with a *MalwareAnalyzer* based on a sandbox approach, as illustrated in Figure 4.1. 4.3 illustrates our strategy to analyze and identify this malware file.

4.3 Experiments and Evaluation

1. RQ1: Does combining multiple AI models of PELID, both traditional ML and DL, allow enhancing the performance of network intrusion detection and reducing analysis time?
2. RQ2: When deploying an IDPS inline system in an intranet with large-scale network traffic, is it fast enough to conduct a deep analysis of network flows for intrusion detection with the AI model generated by the APELID method to ensure that network flows are handled in real time?
3. RQ3: Is it possible to implement malware file detection in the inline IDPS system combined with deep analysis based on the AI model?

4.3.1 Experimental Results

CSE-CIC-IDS2018-based Results

The detailed results of the CSE-CIC-IDS2018 experiment are illustrated in the first part of Table 4.2 and the confusion matrix shown in Table 4.1a.

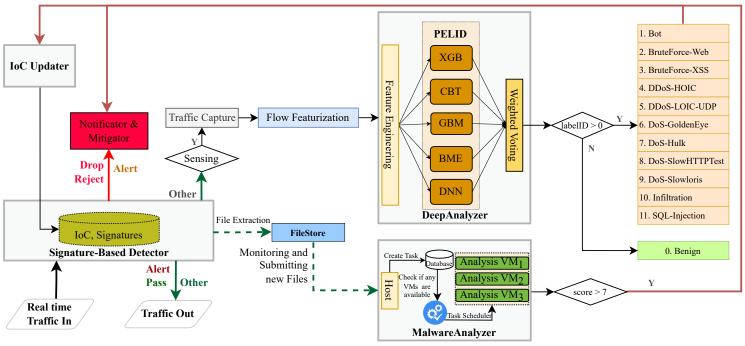


Figure 4.1: Architecture of Holistic Intrusion Detection

Algorithm 4.1 Holistic intrusion Detection by flow sensing strategy and deep analysis

Input: f - Traffic In Flow, S - Signature Set, $Sensing$ - perform AI-powered deep analysis or not, F - Files that transfer between network.
Output: f, msg, S - (Traffic Out Flow; Alert Message; Updated Signature Set)

$action \leftarrow RuleBasedDetector(f, S)$
$IoC_{\text{set}} \leftarrow \emptyset$
if $action = Drop/Reject$ **then** ▷ Drop/Reject flow due of a detected critical attack
  Drop/Reject(f)
  $msg \leftarrow' CriticalAttack'$
  return $(none, msg, S)$
else if $action = Alert$ **then** ▷ Generate an alert
  $msg \leftarrow' Alert\_based\_on\_Signature'$
else if $action = Pass$ **then** ▷ Stop further inspection of the flow
  $msg \leftarrow None$
else if $Sensing = True$ **then** ▷ f does not match any rules, then AI-powered deep analysis is triggered by the sensing mechanism
  $(msg_{deep}, IoC_{deep}) \leftarrow DeepAnalyzer(f)$ ▷ Inspect F deeply by PELID and return a message and new IoC if an intrusion attack is detected.
  $IoC_{set} \leftarrow IoC_{set} \cup IoC_{deep}$ ▷ Update IoC_{set} with new indication of compromise IoC_{deep}
end if
for each $t \in F$ **do**
  $(msg_t, IoC_t) \leftarrow MalwareAnalyzer(t)$ ▷ Analysis t deeply by Sandbox return a message and new IoC_t if an malware file is detected.
  $IoC_{set} \leftarrow IoC_{set} \cup IoC_t$ ▷ Update IoC_{set} with new indication of compromise IoC_t
end for
$S \leftarrow S \cup IoC_{set}$ ▷ Update S with new indication of compromise IoC_{set}
return (f, msg, S)

NSL-KDD-based Results

The second part of Table 4.2 shows the experimental results by using NSL-KDD dataset, and Table 4.1b presents the PELID model's confusion matrix.

Malware Hunting Results

This scenario includes two completely separate networks: DMZ Network (including Web server (HTTP and FTP), Mail Server (SNMP), and Attacks-Network), shown as Table 4.3a. We compared the experimental results with Virus Total (VT), shown in Table 4.3b.

4.3.2 Evaluation

Efficacy of PELID in Intrusion Detection

Compared with individual AI models, as illustrated in Table 4.2. These results privilege us to respond to **RQ1**: combining multiple AI models of PELID allow for improved network intrusion detection.

Efficacy of PELID in Time Consumption

Figure 4.2b shows that the average time the PELID prediction, **RQ2 RQ3** has been resolved by all these experimental results show more in Table 4.2.

4.3.3 Comparison with SOTAs

?? demonstrates that APELID outperforms SOTA and achieves the greatest scores across all evaluation metrics to answering **RQ1.**

Algorithm 4.2 PELID: Parallel Ensemble Learning-based Intrusion Detection

Model: XGB, GBM, CBT, BME, DNN - XGB, GBM, CBT, BME and DNN trained model, and their ensemble weight ω_i where $\sum_{i=1}^{5} \omega_i = 1$.
Input: f - traffic flow.
Output: (msg, R) - (alert messages; new generated rules)

$R \leftarrow \emptyset$
$F \leftarrow Featurize(f)$ ▷ Extract features of traffic flow f.
$Fin \leftarrow Normalize(F)$ ▷ Perform the feature engineering: remove unused features and normalize the rest.
$Cats \leftarrow [DstPort, Protocol]$ ▷ Categorical variables
$Conts \leftarrow Fin \setminus Cats$ ▷ Continuous variables
Perform in parallel five processes P1, P2, P3, P4, P5:
P1: $pXGB \leftarrow XGB.predict(Cats, Conts)$ ▷ Perform the prediction using XGB.
P2: $pGBM \leftarrow GBM.predict(Cats, Conts)$ ▷ Perform the prediction using GBM.
P3: $pCBT \leftarrow CBT.predict(Cats, Conts)$ ▷ Perform the prediction using CBT.
P4: $pBME \leftarrow BME.predict(Cats, Conts)$ ▷ Perform the prediction using BME.
P5: $pDNN \leftarrow DNN.predict(Cats, Conts)$ ▷ Perform the prediction using DNN.
Wait P1, P2, P3, P4, P5 finished.
$scores \leftarrow (pXGB * \omega_1 + pGBM * \omega_2 + pCBT * \omega_3 + pBME * \omega_4 + pDNN * \omega_5)$
$FC \leftarrow scores.argmax(axis = 1)$ ▷ Get the flow predicted label.
if $FC! = 0$ **then** ▷ Classified as network attacks
  $msg \leftarrow Alert(FC, f)$ ▷ Generate an alert by using metadata from the flow f; set alert category being as predicted label.
  $R \leftarrow RuleGenerator(FC, f)$ ▷ Generate a new signature based on its indicator of compromise.
end if
return $msg; R$

Algorithm 4.3 Malware Detection

Input: F - New files transferred in network and accumulated in $FileStore$ folder.
Output: (msg, R) - (Alert Message, New Rules generated based malware detected files).

$Ready \leftarrow Wait\_Sandbox\_Ready$ ▷ Blocking-function until Sandbox is ready.
$IngestFiles(F)$ ▷ Send F in the *FileStore* folder to Sandbox
$score = HybridAnalyzer(F)$ ▷ Determine the overall score of both static and dynamic analysis.
if $score > 7$ **then** ▷ Critical suspicious file
  $R \leftarrow RuleGenerator(F)$ ▷ Update the rule to block connection.
  $msg \leftarrow `Detected\_Malware\_Files'$
  return msg, R
end if

4.4 NetIPS: Deployment of Network Intrusion Detection and Prevention

4.4.1 Deployment Model

The architecture is illustrated in Figure 4.2c and divided into three layers. The lower layer is the network hardware, including SmartNIC (network accelerator) and traditional network interfaces, used to analyze traffic and manage the NetIPS.

4.4.2 Hypermatching for Signature-based Detector

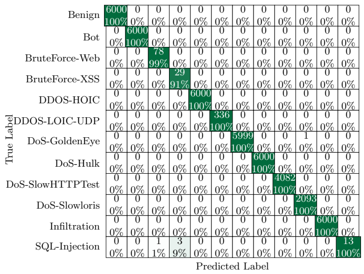
In the Rule-based Detector, the Hyperscan technique is utilized to enhance the efficacy of the ruleset matching procedure. It matches more effectively than other methods (such as Aho-Corasick, Boyer-Moore).

4.4.3 Accelerating AI-powered Intrusion Detection in User Space

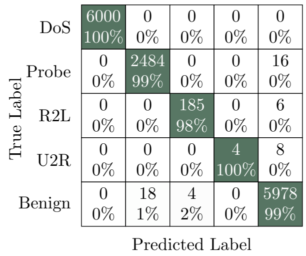
In NetIPS, packet handling is optimized by replacing traditional NICs with a Napatech SmartNIC and leveraging the DPDK library to bypass kernel overhead, reducing context switching and latency.

4.5 Summary

Chapter 4 addresses the critical challenge of deploying AI-powered intrusion detection and prevention systems in large-scale, real-world environments, where requirements for real-time performance, scalability, and operational reliability are paramount. Building upon the data enhancements and ensemble modeling innovations developed in previous chapters, this chapter introduces and evaluates a comprehensive architecture for practical, high-throughput network defense.



(a)

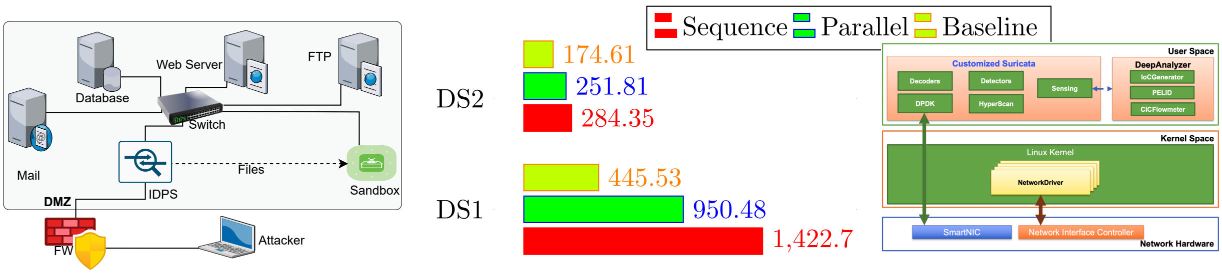


(b)

Table 4.1: Confusion matrices of PELID model: (a) CSE-CIC-IDS2018 and (b) NSL-KDD.

Table 4.2: Evaluation of AI models based PELID (%)

Metric	CSE-CIC-IDS2018						NSL-KDD					
	XGB	CBT	GBM	BME	DNN	**PELID**	XGB	CBT	GBM	BME	DNN	**PELID**
F1	99.77	99.92	99.95	99.77	97.75	**99.99**	99.48	99.21	99.48	99.48	98.00	**99.63**
Acc	99.76	99.92	99.96	99.98	97.54	**99.99**	99.49	99.22	99.56	99.43	98.07	**99.65**
Prec	99.83	99.93	99.96	99.98	98.20	**99.99**	99.49	99.21	99.49	99.41	98.03	**99.65**
Rec	99.76	99.92	99.96	99.98	97.54	**99.99**	99.49	99.22	99.49	99.43	98.07	**99.65**
FPR	**0**	**0**	0.03	**0**	0.13	**0**	0.67	1.27	0.63	0.77	1.22	**0.37**
FNR	**0**	0.01	**0**	**0**	1.37	**0**	0.37	0.39	0.30	0.32	2.26	**0.34**
AUC	**100**	**100**	99.99	99.99	98.69	**100**	**99.99**	99.98	**99.99**	99.89	99.85	**99.99**



(a) (b) (c)

Figure 4.2: (a) malware hunting scenario, (b) parallel vs sequential processing, and (c) APELID-based NetIPS architecture.

(a)

N	Malware	Type	Hash	VT	APELID
1	QuasarRAT	.exe	832ab3a898d188426d3541e1533b55f9	56/68	Yes
2	Loki	.xlsx	5b6aec60c3be4724f7980a659206531a	29/58	Yes
3	STRRAT	.jar	2199150e7d79d0e831cda314c7ce6f56	28/62	Yes
4	AsynRAT	.doc	da6419e4d4e4528990898bcfdaa85e01	32/60	Yes
5	SnakeKeylogger	.exe	715b0f6390ba4387a4155c1d59a3669c	49/69	Yes
6	AgentTesla	.exe	5c590fcb32aedec16532aa857eec28b5	40/66	Yes
7	OskiStealer	.xlsx	6a9203346218dded19d0a8a1dee24023	20/59	Yes
8	NanoCore	.exe	4bae18ac4a73ff38f7ed718365e6c2b2	41/67	Yes
9	DanaBot	.exe	5f4731a4ef7d1484893213caaf6a6685	42/69	Yes
10	DCRAT	.exe	ea800644b9dfd027807447fdd98241aa	50/68	Yes
11	YellowCockatoo	.dll	df7b2ece343c52df774d72e12ea09009	51/69	Yes
12	RemoteManipulator	.exe	4c5649e9b9a2d9997ac2600a804e0aeb	41/68	Yes
13	Pony	.exe	ab468a5b5cd9470c0895097efa2a687f	63/71	Yes
14	Stealc	.exe	cea30f806e644cebe48399eefa345e51	47/71	Yes
15	njRat	.exe	b17414d6949c2e013de14fdc268cfc89	65/71	Yes
16	RedLineStealer	.exe	8a61e10948c23a9a5c353d28b8738490	35/71	Yes
17	Guildma	.zip	8a61e10948c23a9a5c353d28b8738490	35/71	Yes
18	Gozi	.js	1df2e7a13459223b2cc55b93744add77	24/71	Yes
19	DarkTortilla	.exe	1c354a83f81063dc75612a9a7bd51225	54/71	Yes
20	VectorStealer	.xlsx	5b47098a17ecd534de15df03b12beacb	40/71	Yes

(b)

Method	Acc	Prec	F1	Rec
CSE-CIC-IDS2018				
APELID (ours)	**99.99**	**99.99**	**99.99**	**99.99**
MMM-RF	99.98	–	–	–
GAN+RF	99.83	98.68	95.04	92.76
KNN-MQBHOA	99.78	99.56	99.65	99.87
HDLNIDS	98.90	98.63	99.03	99.14
CNN	98.17	95.00	94.00	95.00
AUE	97.90	98.00	98.00	98.00
miniVGGNet	96.99	97.46	97.04	96.97
NSL-KDD				
APELID (ours)	**99.65**	**99.65**	**99.63**	**99.65**
KNN-MQBHOA	99.00	99.00	97.00	98.00
FFO-PNN	98.99	96.97	96.97	96.97
DLNID	90.73	86.38	89.65	93.17
GMM-WGAN-IDS	86.59	88.55	86.88	86.59
Adaptive-Ensemble	85.20	86.50	86.50	85.20
CAFE-CNN	83.34	85.35	82.60	83.44

Table 4.3: (a) Malware hunting results detected by APELID in the wild; (b) Comparison of APELID performance with state-of-the-art intrusion detection methods on CSE-CIC-IDS2018 and NSL-KDD datasets.

Conclusions and Future Work

Contribution Highlights

- Propose a machine learning pipeline with data augmentation and feature optimization (WGAN-powered augmentation + SHAP-based feature optimization) to balance and enhance the quality of training datasets, thereby improving the detection capability for minority-class attacks.
- Introduce a deep and boosting mutual inference framework that strengthens the accuracy and resilience of intrusion and malware detection systems.
- Propose a solution to address data bottlenecks in large-scale network intrusion prevention through a time-interval and frequency-based flow sensing strategy, combined with parallelized inference of deep and boosting mutual inference models.
- Integrate the proposed methods into the NetIPS real-time intrusion detection and prevention system, which leverages AI-based models at the user level to process high-volume traffic (on a large scale), making it suitable for enterprise and ISP networks.

Dissertation Limitations

- All tests were performed using fixed datasets that were prepared in advance, which means that we cannot see how well the model would adapt to real-life situations or when the data change over time.
- The NetIPS component has not yet been extensively validated in various real-world scenarios. In particular, comprehensive evaluations of hardware performance and deployment feasibility have not been conducted in large-scale production networks.
- The current experimental design does not include ablation studies to quantify the contribution of individual components or techniques to the overall performance. Such evaluations could provide more details on the effectiveness of the system and guide future optimizations.
- The models were trained primarily on structured network or PE data. More complex attack vectors, such as encrypted traffic, multistage malware, etc.., were not within the scope of this study.

Future Research Directions

- Online and continuous learning: Integrating online learning methods and incremental retraining into detection pipelines could allow models to adapt to evolving threats and handle dynamic environments more effectively.
- Future systems could use different types of data, such as how hosts behave, process trees, user activities, and patterns in encrypted traffic, all within a single detection framework.
- Automated response and defense integration: Improving detection systems with immediate actions, like automatically blocking threats, updating rules, or prioritizing alerts, can connect simple detection with active defense.
- Making it easier to understand decisions: Creating simple and user-friendly tools that explain how AI systems work, particularly for endpoint systems, can build trust and help security analysts work better with AI tools.

Personal Publications

Journals

VVH-J1 **Hoang V. Vo** and Hanh P. Du and Hoa N. Nguyen, AI-powered intrusion detection in large-scale traffic networks based on flow sensing strategy and parallel deep analysis, *Journal of Network and Computer Applications* 220 (2023) 103735. DOI: 10.1016/j.jnca.2023.103735; (IF 8.0, SCI-E, top 2% Q1-Scopus)

VVH-J2 **Hoang V. Vo** and Hanh P. Du and Hoa N. Nguyen, APEPID: Enhancing real-time intrusion detection with augmented WGAN and parallel ensemble learning, *Computers and Security* 136 (2024) 103567. DOI: 10.1016/j.cose.2023.103567; (IF 5.4, SCI-E, top 7% Q1-Scopus)

VVH-J3 **Hoang V. Vo** and Hanh P. Du and Hoa N. Nguyen, MDOB: Enhancing Resilient and Explainable AI-Powered Malware Detection Using Feature Set Optimization and Mutual Deep+Boosting Ensemble Inference. *Journal of Information Security and Applications* 2025 93 (2025) 104175. DOI: 10.1016/j.jisa.2025.104175; (IF 3.7, SCI-E, top 8% Q1-Scopus)

Conferences

VVH-C1 **Hoang V. Vo**, Hoa N. Nguyen, Tu N. Nguyen, Hanh P. Du, *SDAID: Towards a Hybrid Signature and Deep Analysis-based Intrusion Detection Method*, in: GLOBECOM 2022 - 2022 IEEE Global Communications Conference, 2022, pp. 2615–2620. DOI: 10.1109/GLOBECOM48 099.2022.10001582. (WoS, Scopus)

VVH-C2 **Hoang V. Vo**, Duong H. Nguyen, Tuyen T. Nguyen, Hoa N. Nguyen, Duan V. Nguyen, *Leveraging AI-Driven Realtime Intrusion Detection by Using WGAN and XGBoost*, in: Proceedings of the 11th International Symposium on Information and Communication Technology, Association for Computing Machinery, New York, NY, USA, 2022, p. 208–215. DOI: 10.1145/3568562.3568660. (WoS, Scopus)

VVH-C3 **Hoang V. Vo**, Phong H. Nguyen, Hau T. Nguyen, Duy B. Vu, Hoa N. Nguyen, *Enhancing AI-Powered Malware Detection by Parallel Ensemble Learning*, in: 2023 RIVF International Conference on Computing and Communication Technologies (RIVF), 2023, pp. 503–508. DOI: 10.1109/RIVF60135.2023.10471855. (WoS)

VVH-C4 **Hoang V. Vo**, Hanh P. Du and Hoa N. Nguyen, *AWDLID: Augmented WGAN and Deep Learning for Improved Intrusion Detection*, 2024 1st International Conference On Cryptography And Information Security (VCRIS), Hanoi, Vietnam, 2024, pp. 1-6, DOI: 10.1109/VCRIS63677.2024.10813392. (WoS)