This is the title

[a], , , , ,

Abstract

The rapid evolution of cyberattack techniques has opened up a novel challenge known as multiclass novelty detection (MND) within the realm of cyber intrusion detection. Despite numerous successful studies in addressing the multi-class classification or the detection of novel attacks, the emergence of the MND challenge has posed a gap in Intrusion Detection Systems (IDS) research. Motivated by this challenge, this study proposes a novel single-model algorithm, named neighbour Null Foley-Sammon Transformation (nNFST), for integrating into IDS to address the MND challenge. The nNFST algorithm applies a novel technique based on the inverse nearest neighbour algorithm to compute within-class and between-class variation. This technique preserves both the local distribution structure within each class and the global distribution structure across classes, thereby mitigating the impact of singular points on the algorithm and enhancing accuracy when handling complex data. Additionally, we employ the kernel trick to ensure algorithm accuracy and leverage sparse matrix multiplication to calculate distance matrices, thereby reducing the training cost. The nNFST algorithm is evaluated across four public datasets and archives high accuracy on different tasks: multiclass classification task (Accuracy from 97.12% to 99.56%), detecting novel attack task (Accuracy from 94.53% to 99.33%) and a high Matthews correlation coefficient score (from 0.825 to 0.975) in addressing the MND challenge. These experimental results indicate the superiority of our proposed algorithm compared to another baseline competitor in solving different tasks with high accuracy.

Keywords: Intrusion Detection System, Multiclass Nolvety Detection, Null Foley-Sammon Transformation, Inverse Nearest Neighbour

1. Introduction

The fast-moving Internet technology has revolutionized modern lifestyles and moved the wheel of development in various industries. However, this process also presents potential threats to everyday internet users. In efforts to strengthen network security, anomaly-based intrusion detection systems have emerged as an essential solution, which is superior to traditional approaches by applying machine learning models (1; 2). However, despite their success showing in previous studies, IDSs now face new challenges in keeping up with the swift evolution of attack techniques. Beyond simply categorizing known attacks or detecting anomalies, the new challenges for next-generation IDSs lay in handling both known attack classifications and the identification of novel, previously unseen attacks. This new challenge, known as Multiclass Novelty Detection (3; 4), has been researched in the computer vision field for years but has not received much attention in the network security domain.

1.1. Multiclass Novelty Detection for IDS

The Multiclass Novelty Detection (MND) problem in the context of network intrusion detection refers to the task of enabling models to classify trained attacks and identify unseen attacks (often termed novel attacks). There are three main approaches to addressing the MND problem: using super-class, employing multiple single-class models, and using a single model. In the IDS context, the super-class approach is impractical as it requires combining both attack and normal data into a single label. Conversely, employing multiple single-class models is a possible approach, but it complicates the model as the number of attacks increases, potentially reducing the system performance. On the other hand, using a single model, while practical, requires the model to classify and detect new attack types — a task typically handled by two different types of models. Consequently, while there have been applications in computer vision, there remains a noticeable research gap in addressing the MND problem within the field of network intrusion detection.

Acknowledging the research gap mentioned above, this study introduces an algorithm called neighbour Null Foley-Sammon Transformation (nNFST), specifically designed to address the multiclass novelty detection challenge within the realm of network intrusion detection. Building upon the solid theoretical foundation of the Null Foley-Sammon Transformation (NFST), nNFST presents its suitability for integrating into Intrusion Detection Systems. To the best of our knowledge, this is a pioneering study in addressing the challenge of MND in the domain of network intrusion detection using a single model.

1.2. Null Foley-Sammon Transformation

The Null Foley-Sammon Transformation (NFST) (3; 5) is among the few single models capable of addressing MND with the core concept revolving around maximizing the Fisher criterion. The Fisher criterion's primary objective is to enhance the separation between the means of different classes while minimizing the spread (variance) within each class. This is achieved

by finding a projection direction in the new feature space that maximizes the ratio of between-class variance to within-class variance. The NFST model maximizes the Fisher criterion by employing the Null Foley-Sammon Transformation, which maps samples within each class to a single point while ensuring that the between-class variance remains greater than zero, therefore, maximizing the Fisher criterion to infinitive. The NFST has demonstrated outstanding performance in addressing the MND task within the realm of computer vision, but its application in the network intrusion detection domain presents certain limitations that may reduce its performance:

- The first and foremost weakness of NFST lies in its approach to calculating within-class variance and selecting representative samples for between-class variance calculation. NFST calculates the distance between classes and within-class points based on class level, assuming that samples in the same class belong to the same distribution. Consequently, the NFST simply chooses the representative sample by calculating the mean of all samples within a class and computes the between-class variance via the representative samples. While this method is straightforward and fast, it overlooks many cases in practice where samples from the same classes belong to different distributions, resulting in the mixing of several subclasses or clusters within a class. As a result, the original assumptions cause NFST to fails to capture the local structure of each class, leading to two important consequences: (1) Vulnerability to Singular Points: NFST is highly sensitive to the presence of singular points, particularly when the data is sparse or do not come from a single distribution. (2) Reduced Accuracy with Mixed Labels: The model's accuracy is reduced when dealing with labels that consist of mixed local distributions. The theory of these weakness is introduced in research (6; 7). We explain more detail and give an illustration of these weaknesses in Section 3.

- The second notable weakness of NFST lies in the expensive computation, mainly attributed to the complicated matrix computations involved, which scale with the number of input samples. The computational process requires many complex calculations on vector matrices, typically based on pairwise distances within-class samples and between classes. As a result, the computational complexity grows quadratically with the number of samples. Consequently, for large datasets, the computation of these matrices becomes computationally intensive, leading to significant processing times and resource requirements. This limitation poses a challenge to the model training process, particularly in real-world application scenarios with large datasets.

Recognizing the aforementioned weaknesses significantly impacts the applicability of NFST in the context of network intrusion detection. In this study, we propose an algorithm named the neighbor Null Foley-Sammon Transformation (nNFST) to address the existing weaknesses and apply the new model to solve the Multiclass Novelty Detection (MND) problem in network intrusion detection. The general idea behind the nNFST model to address the first weakness of Reduced Accuracy with Mixed Labels and Vulnerability to Singular Points is to introduce a novel representative point selection method based on the inverse nearest neighbor technique to accurately represent the local distribution of each class. Additionally, we observe that the original model's matrix operations often involve calculations with sparse matrices. Therefore, we propose to apply sparse matrix multiplication techniques to enhance the computational efficiency of the model.

1.3. Main contribution

The swift evolution of cyberattack techniques has opened up a novel challenge known as multiclass novelty detection within the realm of cyber intrusion detection. Despite numerous successful studies addressing individual aspects, such as classification problems or the detection of new attacks, the emergence of the MND problem has posed a gap in IDS research. Motivated by this challenge, this study proposes a novel algorithm designed as a single model to be applied in IDS to address the MND challenge. The main contributions of this research are summarized as follows:

- We propose the neighbor Null Foley-Sammon Transformation (nNFST) algorithm as a unified IDS model. This novel approach enables dual capabilities: accurate classification of known attacks and detection of novel attacks, representing the first attempt to tackle the MND problem with a single model in intrusion detection.

- Our algorithm introduces an innovative method based on an inverse nearest neighbour to compute within-class and between-class variation. This technique preserves both the local distribution structure within each class and the global distribution structure across classes, thereby mitigating the impact of singular points on the model and enhancing accuracy when handling complex data.

- Additionally, we employ the kernel trick to ensure algorithm accuracy and leverage sparse matrix multiplication to calculate distance matrices, thereby reducing the training cost.

- Finally, we conduct extensive experiments on various public datasets with intrusion detection domain. The experimental results consistently demonstrate the superiority of the proposed algorithm compared to other baseline competitors in various attack scenarios and in different tasks.

The rest of this paper is presented as follows. Section 2 reviews related work on network intrusion detection based on the anomaly approach. Section 3 presents the problem formulation, while Section 4 introduces in detail our proposed method. Sections 5 and 6 present our experiments setup, datasets, and the results. Finally, Section 7 gives our concluding remarks.

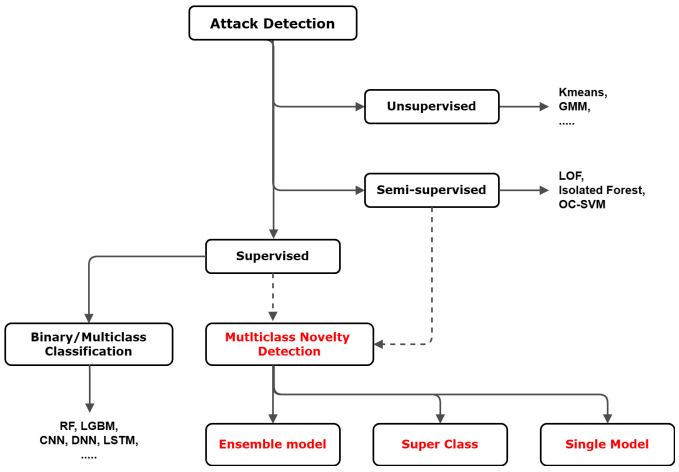

Figure 1: The illustration of IDS solution's approach.

2. Related works

The network intrusion detection system has proven itself as a state-of-the-art solution for safeguarding network systems against cyber threats. With advances in technology, the current trend in NIDS development is to integrate machine learning and deep learning to build anomaly-based NIDS. Survey researchs (8; 9; 10) indicates that the anomaly-based approaches offer superior accuracy and efficiency compared to conventional NIDS, which is fixed on static analysis and struggles to adapt to novel attacks. Based on the training sources (described in Figure 1), the current AI application in NIDS consists of three main approaches: unsupervised, semi-supervised, and supervised learning. In which, supervised learning can further be categorized into binary and multiclass tasks. However, to adapt to the evolution of cyber threats, a novel task for NIDS has emerged known as Multiclass novelty detection (MND), which is quite challenging as it requires NIDS to perform both tasks of Semi-supervised and Multiclass-supervised models. Because of its difficulty and novelty, hence, despite its potential to be a significant security enhancement, there is a limited number of solutions following this approach. In what follows, we discuss in sequence the solution approaches and the corresponding outstanding research.

Unsupervised approach: In the context of intrusion detection, the unsupervised-based method is related to systems that operate without requiring labelled input data. This characteristic stands as a significant advantage of unsupervised-based systems since the labelling process demands considerable time and expertise from security experts. Consequently, the unsupervised-based NIDS can immediately detect novel attacks without waiting for datasets to be annotated by experts. Several notable solutions adopt this approach, such as using cluster models or applying active learning for training models. Table 1 provides an overview of research that proposes a system based on this approach (11; 12; 13). However, despite the advantage of being label-free, unsupervised-based systems face limitations in classifying attack types. Some research proposed using active learning to address this problem, but it requires analysis from security experts. Furthermore, the unsupervised-based NIDS is typically less accurate compared to other approaches and usually has a high false alarm rate.

Semi-supervised approach: Semi-supervised NIDSs are systems that require partly labelled data for model training. In the context of intrusion detection, these systems typically use labelled normal network data for training to detect anomaly traffic from normal traffic. Such systems often address tasks such as one-class detection, novelty detection, or outlier detection. One notable advantage of this approach is that systems can detect novel attacks without requiring extensive resources for data labelling. Moreover, by training with a portion of the labelled data, the systems can reduce the false alarm rate. Table 1 showcases several noteworthy projects demonstrating the efficiency of this approach (14; 15; 16), which has garnered significant attention in the field. However, models based on this approach face challenges in accurately classifying known (trained) attack types.

Supervised approach: In contrast to other approaches, supervised-based Network Intrusion Detection Systems (NIDS) require labelled data to train models that can classify malicious activities or threats on a network. This approach offers high accuracy in detecting known attack patterns, as the system learns from previous examples. Furthermore, they can easily classify attack types and normal traffic. However, a notable drawback is its limited ability to detect novel or unknown attacks not present in the training dataset. Additionally, supervised-based NIDS require a substantial amount of labelled data, which can be costly and time-consuming to prepare. Numerous research projects have been proposed and achieved success in this domain, particularly with the advent of deep learning , which has propelled development to new heights.

In the binary classification task, the NIDS focused on classifying between normal traffic and other attacks, which were grouped into one superclass. The multiclass classification task is an extension of the binary task, where the NIDSs are trained with full-labelled datasets and focus on classifying multiclass. Various research efforts addressing this task are introduced in Table 1, such as reseach (17; 18; 19; 20). They have made significant strides in solving these tasks. These NIDS archive high accuracy while minimizing false alarm rates. Furthermore, they reach a high processing rate, which is a significant advantage when applied to resource-constrained devices.

Multiclass novelty detections task: Recently, to keep pace with the evolution of cyber attacks, a new task has appeared named Multiclass Novelty Detections. This task requires Network Intrusion Detection Systems (NIDS) to classify known attack types while simultaneously detecting novel classes. Although research on MND has achieved success in the field of computer vision and inspired its application in other domains, it remains relatively unexplored in network intrusion detection. In general, there are three main approaches to addressing MND (described in Figure 1): Super class, Ensemble model and Single model.

In the domain of intrusion detection, the super-class approach seems impractical since it requires grouping both normal and attack traffic into superclasses, which goes against the

goal of detecting attacks in network traffic. In contrast, the second approach has gained little attention. Notable research, such as (21), has explored this direction. This study proposed a two-layer NIDS, which combined a deep learning model named SOCNN for classifying known attacks and two outlier detection models (LoF and iNNE) for detecting novel attacks. However, the second approach has significant drawbacks since it combines many single models, making the detection process more complex and then slowing down the detection process. Finally, the third approach, employing a single model, appears to be the most promising. It has garnered attention in the computer vision domain, particularly through models based on null transformation. However, despite its potential, our survey indicates that no existing research has applied this single-model approach for solving MND tasks within the intrusion detection domain.

Recognizing the importance of addressing the Multiclass Novelty Detection (MND) task in the realm of intrusion detection systems (IDS), this research proposes a novel single-model approach for NIDSs. The significance of this work lies in its potential to enhance network security by effectively detecting novel attacks while ensuring high accuracy in detecting known attacks, thereby keeping pace with the rapid evolution of cyber threats.

3. Problem fomulation

3.1. The main task in intrusion detection domain

The research and development of intrusion detection systems have been recognized as important work for a long time. However, the rapid evolution of attack techniques poses a challenge to conventional IDSs, which cannot detect new attacks that appear every day. Recognizing this challenge, we introduce a novel research direction named Multiclass Novelty Detection (MND), which requires IDSs to identify novel attacks while classifying known attack types. Numerous studies have been proposed to address the MND task in the computer vision domain; but in the domain of IDS development, it still lacks of pioneering research and specific definitions. Therefore, in this section, based on definitions from previous studies, we redefine the three main challenges that IDSs must address: multiclass classification, one-class novelty detection, and multiclass novelty detection.

Multiclass classification: This is the traditional problem that most IDS solutions focus on solving. Herein, IDSs are required to classify the types of data within the testing set with a note that the data types in the training and testing sets are the same. From a mathematical point of view, given the set Z of network traffic, dataset $S = (X, Y)$ represents the given network traffic.

$$Z = B \cup A \tag{1}$$

where:
$B = \{b_i \mid i \in [1, 2, \ldots, N_B]\}$ denotes the normal (benign) traffic.
$A = \{a_i \mid i \in [1, 2, \ldots, N_A]\}$ represents for N_A attacks.

$$S = \{(x_i, y_i) \mid x_i \in \mathbb{R}^D, y_i \in Z, i \in [1, 2, \ldots, n]\} \tag{2}$$

where:
n denotes the number of the observed sample.
m denotes the number of features of each sample.
x_i denotes for a network traffic session.
y_i denotes coresponding labels.

From the labeled dataset S, an IDS is trained with an objective function $f(\cdot)$ using the training set $S_{train} = \{(x, y) \mid x \in \mathbb{R}^D, y \in Z\}$. Subsequently, for each testing sample x in $S_{test} = \{(x, y) \mid x \in \mathbb{R}^D, y \in Z\}$, the model predicts its labels via the trained $f(\cdot) : y = f(x \mid S_{test}) : \mathbb{R}^D \rightarrow Z$. Notably, the binary classification task is a special case when $N_A = N_B = 1$.

One-class novelty detection: Also known as one-class classification or outlier detection, this task aimed at identifying observations that are significantly different from the trained data. Widely researched in the IDS development domain, in this task, the IDS is usually trained with normal traffic and detects instances significantly different from the trained traffic, typically indicative of attack traffic. Unlike multiclass classification task where data points belong to one of the multiple trained classes, here the IDS is towards separating anomalies from the normal traffic.

$$Z_N = \{0, 1\} \tag{3}$$

where:
$Z_N = 0$ denotes normal traffic.
$Z_N = 1$ denotes anomaly traffic.

$$S_N = \{(x_i, y_i) \mid x_i \in \mathbb{R}^D, y_i \in Z_N, i \in [1, 2, \ldots, n]\} \tag{4}$$

An IDS is trained with an objective function $f(\cdot)$ using the training set $S^N_{train} = \{(x, y) \mid x \in \mathbb{R}^D, y = 0\}$. Subsequently, for each testing sample x in $S^N_{test} = \{(x, y) \mid x \in \mathbb{R}^D, y \in Z_N\}$, the model predicts its labels via the trained $f(\cdot) : y = f(x \mid S^N_{test}) : \mathbb{R}^D \rightarrow Z_N$.

Multiclass novelty detection: This is a novel task and has no specific definition in the field of IDS development. In general, the Multiclass Novelty Detection (MND) task aims to build a model capable of detecting novel classes while classifying trained multiclass data. Particularly in the IDS domain, the MND task combines aspects of both aforementioned tasks, where IDS is required to perform classifying attacks on trained data and perform novelty detection to detect novel attacks.

$$Z_{MN} = Z \cup \{-1\} \tag{5}$$

where:
$Z_{MN} = -1$ represents for novelty attacks.

$$S_{MN} = \{(x_i, y_i) \mid x_i \in \mathbb{R}^D, y_i \in Z_{MN}, i \in [1, 2, \ldots, n]\} \tag{6}$$

The goal of solving MND task is to trained An IDS with an objective function $f(\cdot)$ using the training set $S^{MN}_{train} = \{(x, y) \mid x \in \mathbb{R}^D, y \in Z\}$. Subsequently, for each testing sample x in $S^{MN}_{test} = \{(x, y) \mid x \in \mathbb{R}^D, y \in Z_{MN}\}$, the model predicts its labels via the trained $f(\cdot) : y = f(x \mid S^{MN}_{test}) : \mathbb{R}^D \rightarrow Z_{MN}$.

Table 1: The overview of recent research on IDS

ID	Tittle	Year of release	Summary	Training approach	Tasks	Model	Datasets	Performance
1	Traffic Anomaly Detection Model Using K-Means and Active Learning Method(11)	2022	Proposed using K-means for detecting attack and Active learning for boosting model accuracy	Unsupervised	Novelty Detection	Kmeans + Active learning	CICDDoS2019	ACC >90.20% F1 >94.90% FPR <0.01%
2	Autoencoder-Based Unsupervised Intrusion Detection Using Multi-Scale Convolutional Recurrent Networks(12)	2022	Proposed a deep learning-based ensemble model for detecting novel attacks	Unsupervised	Novelty Detection	MSCNN-LSTM-AE + Isolation Forest	NSL-KDD, UNSW-NB15, CICDDoS2019	AUC: 0.96 - 0.97
3	Robust Unsupervised Network Intrusion Detection With Self-Supervised Masked Context Reconstruction(13)	2023	Proposed a self-supervised learning scheme for building a robust unsupervised NIDS	Unsupervised	Novelty Detection	RUIDS	KDD, UNSW-NB15, CIC-IDS-2017	ACC: 92-99%, F1: 95-99%, AUC: 88-99
4	Realguard: A Lightweight Network Intrusion Detection System for IoT Gateways(17)	2022	Proposed a lightweight NIDS for IoT network based on deep neuron network	Supervised	Multiclass Classification	DNN	CIC-IDS-2017	ACC = 99.93%, TPR = 99.57%, FPR = 0.04%
5	A Deep Learning Technique for Intrusion Detection System Using a Recurrent Neural Networks based Framework(18)	2022	Proposed a IDS based on Recurrrent Neural Networks	Supervised	Multiclass Classification	RNN/LSTM/GRU	NSL-KDD, UNSW-NB15	ACC: 74-85%
6	DCNNBiLSTM: An Efficient Hybrid Deep Learning-based Intrusion Detection System(19)	2023	Proposed a IDS based on a hybrid model combining BiLSTM and DCNN	Supervised	Multiclass Classification	CNN + BiLSTM + DNN	CIC-IDS-2018, Edge_IIoT	ACC = 99%
7	Deep Residual Convolutional Neural Network: An Efficient Technique for Intrusion Detection System(20)	2024	Proposed an IDS based on Deep Residual Convolutional neuron network and optimized by the proposed Improved Gazelle optimization algorithm	Supervised	Multiclass Classification	DCRNN	UNSW-NB15, CIC-IDS-2017, CiCDDoS2019	ACC: 99% TPR: 98%-99% FPR: 0.78%-1.49%
8	Fed-ANIDS: Federated Learning for Anomaly-based Network Intrusion Detection Systems(14)	2023	Proposed an IDS based on autoencoder and federated learning scheme for securing a distributed network	Semi-supervised	Novelty Detection	Autoencoder + Federated Learning	USTC-TFC2016, CIC-ID-S2017, CIC-IDS-2018	ACC:92-99%, FPR: 0.46-1.75%, F1: 88-99%
9	Unsupervised Anomaly Detectors to Detect Intrusions in The Current Threat Landscape(15)	2021	An comprehensive survey research on applying semisupervised and unsupervised models on the intrusion detection domain	Semi-supervised/ Unsupervised	Novelty Detection	17 models	11 public datasets	MCC: 0.10 - 0.89
10	An Adaptable Deep Learning-based Intrusion Detection System to Zero-Day Attacks(16)	2021	Proposed a novelty-based framework and a deep novelty-based classifier named DOC++ to detect zero-day attacks	Semi-supervised	Multiclass Novelty Detection	Ensemble model	CIC-IDS-2018, CIC-IDS-2017	ACC: 42-99%
11	Robust Detection of Unknown Dos/Ddos Attacks in Iot Networks Using a Hybrid Learning Model(21)	2023	Proposed a two-stage NIDS combined of SOCNN and outlier detectors to classify DoS/DDoS attacks and detect their novel variants	Supervised/ Semi-supervised	Multiclass Novelty Detection	SOCNN + LoF + iNNE	BoT-IoT, CIC-IDS-2018, CIC-IDS-2017	ACC = 99% F1 = 99% FPR <0.16% UDR: 71-99%
12	Ours	N/a	Proposed a novel IDS based on nNFST to solve the multiclass novelty detection task	Supervised/ Semi-supervised	Multiclass Classification, Novelty Detection, Multiclass Novelty Detection	nNFST	BoT-IoT, ToN-IoT, N-BaIoT, CIC-IoT2023	MCC: 0.87 - 0.98 ACC: 97%-99% TPR: 85%-98% FPR: 0.27%-1.73% F1: 86%-98%

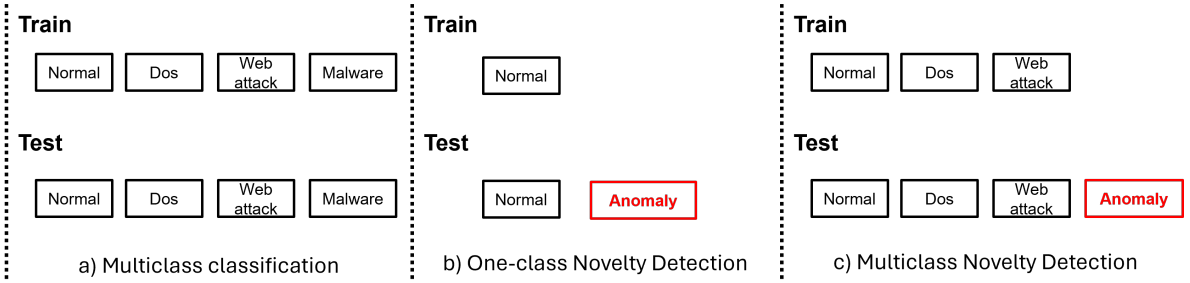

Figure 2: The illustration of challenges of IDSs.

In the scope of this reseach, we proposed a model named neighbor Null Foley-Sammon Transformation to address the MND task in the context of network intrusion detection.

3.2. *Null Foley-Sammon Transformation briefly and its Weakness*

Before looking into the details of NFST and our proposed version, we recap the general idea of NFST and its weakness for reader to better understand.

Null Foley-Sammon Transformation is a classifier based on the Fisher criterion. The model aim is to project the data from the current space to a null space that would map all samples of the same class into a single point in new space. Let X_j denotes the dataset of class j; the dataset $X = \{X_j \mid j = 1, \ldots, c\}$ denotes the training set; $N = \sum_1^c n_j$ is the total sample of X, and n_j denotes for the total sample of class j. The Fisher criterion is formulated as follows:

$$F(v) = \frac{v^\top M_b v}{v^\top M_w v} \tag{7}$$

$$M_b = \sum_{j=1}^{c} n_j (\sigma_j - \sigma)^\top (\sigma_j - \sigma) \tag{8}$$

$$M_w = \sum_{j=1}^{c} \sum_{i=1}^{n_j} (x_{ij} - \sigma_j)^\top (x_{ij} - \sigma_j) \tag{9}$$

$$M_t = \sum_{j=1}^{c} \sum_{i=1}^{n_j} (x_{ij} - \sigma)^\top (x_{ij} - \sigma) \tag{10}$$

where:
M_b represents for the scatter matrix between classes.
M_w represents for the within-class scatter matrix.
M_t represents for the total-scatter matrix.
$\sigma = \frac{1}{N} \sum_{j=1}^{c} \sum_{i=1}^{n_j} x_{ij}$ denotes the mean point of all training sample.
$\sigma_j = \frac{1}{n_j} \sum_{i=1}^{n_j} x_{ij}$ denotes the mean point of class j $v \in \mathbb{R}^D$ denotes the project direction.

The algorithm achieves the best separability when the Fisher criterion is maximized. Hence, the NFST algorithm proposed to find the direction that satisfies the following conditions:

$$v^\top M_b v > 0 \text{ and } v^\top M_w v = 0 \tag{11}$$

Obviously, the discriminant vectors v in the equation 11 safistfy the $F(v) = \infty$, which means the training samples was best separated on the direction v following by Fisher criterion.

The equation 11 equivale to equation 12 and the direction v is called as null projection direction (NPD).

$$v^\top M_t v > 0 \text{ and } v^\top M_w v = 0 \tag{12}$$

Following the equations provided above, the NFST algorithm expects to project data into new space, where samples within a class are mapped into a single point and the distance between classes is maximized. However, it is obvious that equations 11 and 12 only capture the global data structure between classes and lose the local structure within classes by considering only a single mean point for each class. This poses a weakness when a class has a complex local structure, such as building up from rather than one local cluster. Furthermore, research (6) has demonstrated that the existence of crossover points between classes can result in the collapse of both classes into a singular point in null space. Figures 3 and 4 give examples to illustrate these shortcomings of the NFST algorithm.

Let three classes be mapped into null space, with class two building up from two clusters. In the case of Figure 3, assuming that classes two and three have a crossover point, employing the NFST algorithm would result in all samples from classes two and three being mapped to a singular point and lead to misclassification. In the case of Figure 4, the testing sample x is mapped to point x' and expected to be closest to class 3 in null space. However, since the NFST considers point a to belong to class 2, hence class 2 is mapped to point $C^{2\prime}$ and closer to point x' than point $C^{3\prime}$, leading to the misclassification of point x.

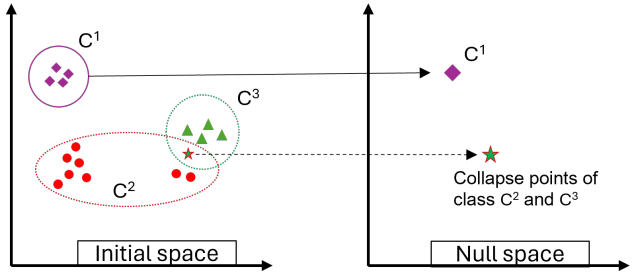

Figure 3: The illustration of collapsing singular points.

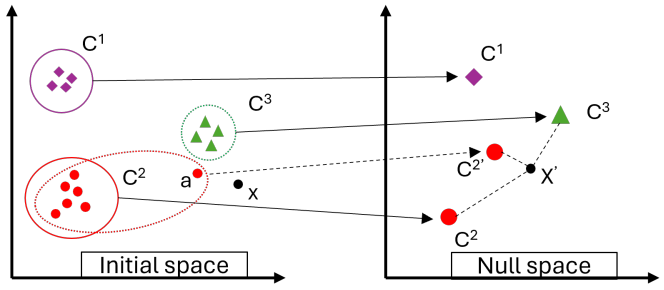

Figure 4: The illustration of weakness in NFST.

In this research, we introduce our robust version of NFST with the use of a neighborhood-based algorithm named inverse nearest neighbor to maintain the local structure of data, thereby, migrating the effect of collapsing singular points and improving the accuracy of the algorithm.

Table 2: The summary of notation used in this research.

Notation	Describe
	In the Problem fomulation section
Z	The set of network traffic types
$S = (X, Y)$	The given network traffic
B	The set of N_B normal (benign) traffic.
A	The set of N_A attack traffic
D	The number of features
n	The number of sample data
	In the Proposed Method section
X	The set of N data sample
D	The number of features
N	The number of sample data
$NN_r(p)$	The set of r closest points to p
$INN_r(p)$	The set of points that are inverse nearest neighbors level r of p
$d(x_1, x_2)$	The euclidean distance between x_1 and x_2
$G = (V, E)$	The graph represent for V sample and E connection between x and $INN_r(x)$
$SCCs^r$	The set of Q SCC^r in the graph.
SCC^r	A strongly connected component in the graph, represented as a set of nodes.
$SCCs^r_{Y_j}$	The set of Q_j SCC^r in the graph with nodes within class j.
Q	The number of strongly connected component in graph G
Y_{scc}	The set of label for each SCC
M_{bscc}	The between-SCCs scatter matrix
σ_j	The mean of the samples in SCC^r_j
σ	The mean of all training samples
M_{wscc}	The within-SCCs scatter matrix
$F(v)$	The Fisher criterion
M_{tscc}	The total-SCCs scatter matrix
v	The null projection direction (NPD)
Z_t	The null space of M_{tscc}
Z_w	The null space of M_{wscc}
B	An orthonormal basis of the subspace W_{tscc}
ϕ	The solution of Equation 29
$k(x_i, x_j)$	The RBF kernel
K	The matrix formed by the inner products within mapped training data
$Nolvety\_scores(x_i)$	The novelty score of sample x_i

4. Proposed Method

4.1. The overview of proposed method

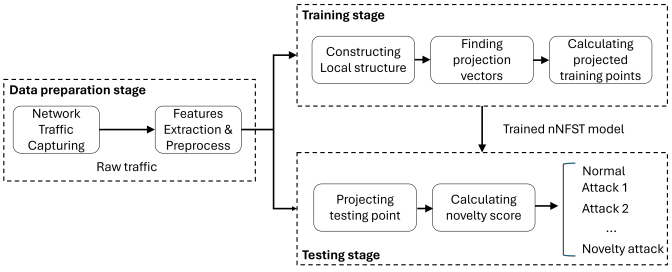

Figure 5: The overview workflow of our proposed NIDS.

In this section, we begin by outlining the workflow of our proposed Network Intrusion Detection System (NIDS) and then go into the details of the proposed nNFST algorithm integrated into NIDS to address the MND task. Figure 5 presents an overview of our proposed NIDS with three main stages:

- **The Data preparation stage**: This stage is responsible for monitoring, capturing, and extracting data for classification models. Initially, popular tools such as Wireshark and Tshark are used to monitor and capture all inbound and outbound traffic on the network. Subsequently, public network feature extraction techniques like CICFlowMeter are employed to aggregate, analyze, and extract valuable information from raw network traffic. This process transforms raw traffic into an executable format suitable for classification model processing. Finally, the extracted features are then normalized to ensure a consistent representation of network behavior before being forwarded for classification.

- **The training stage**: This stage is responsible for training our proposed nNFST algorithm. It first constructs the local structure of the data through the Inverse r-nearest neighbor algorithm and relabels following based on cluster-ID assignments. Then we apply the NFST algorithm to find the projection vector and calculate the base point of each class in the null space. Further details of the algorithm are presented in the following sub-section.

- **The testing stage**: This stage involves making predictions for incoming network traffic. After passing through the data preparation stage, the testing data is then projected into the null space and assigned a novelty score based on the distance to base points. Using the novel score, the algorithm determines whether the testing data is novel or belongs to one of the trained classes.

Since our main contribution in this research focuses on solving the MND task in the network intrusion detection domain, hence, we do not go into the details of the data preparation stage. Instead, we use publicly available datasets to evaluate the performance of our system. Next, we mainly discuss the details of the proposed algorithm nNFST and its application in detecting network intrusions.

4.2. Theory analysis of nNFST

This subsection provides a comprehensive analysis of our proposed algorithm, nNFST. It first describes the algorithm for determining the local structure of training data and then provides the theory of the discriminative algorithm. Additionally, some side techniques are introduced to enhance the algorithm's accuracy and reduce the computational costs.

4.2.1. Inverse r-nearest neighbor

As mentioned above, one of the weaknesses of NFST is its selection of the class representation point, which loses the inner data structure. To overcome this limitation, our research introduces a novel approach for selecting the representation point and regulating the optimization function by using the inverse nearest neighbor (INN). The main idea of INN is to group data points with strong relationships into clusters by using bi-directional connection. Inspired by the idea of K-nearest-neighbor, the INN connects two samples that are both top-r neighbors of each other, hence, making the connection within the cluster stronger. For more details, we present the INN formulation as below.

Definition 1 (Inverse r-nearest neighbor set): Given the set $X = \{x_i \in \mathbb{R}^D \mid i = 1, 2, \ldots, N\}$, we denote $NN_r(p)$ as the set of r closest points to p. The group of points that are inverse nearest neighbors level r of p is denoted as $INN_r(p)$ which satisfies the following conditions:

$$x_i \in INN_r(x_p) \iff \begin{cases} x_i \in NN_r(x_p) \\ x_p \in NN_r(x_i) \end{cases} \tag{13}$$

For a predefined value of r, the $INN_r(\cdot)$ could be empty, contain fewer than, equal to, or more than r samples. Figure 7 illustrates an example of INN on six samples with $r = 2$. It can be seen that x_1, x_2, and x_3 belong to the $NN_2(\cdot)$ set of each other and form a cluster. Additionally, sample x_5 has both x_4 and x_6 in its $NN_2(x_5)$ set; however, both x_4 and x_6 only have x_5 in their NN_2 set since they are not in $NN_2(x_3)$, making only a one-way connection to x_3.

In this research, we present each data sample as a point on a hyperplane and use the euclidean distance to calculate the distance between two samples. The formulation of distance calculation between two given samples is as follows:

$$d(x_1, x_2) = \sqrt{\sum_{i=1}^{D} (x_{1i} - x_{2i})^2} \tag{14}$$

Definition 2 (Strongly Connected Components based r - $SCCs^r$): Given the r value and dataset $S = (X, Y)$, with $X = \{X_j \mid j = 1, \ldots, c\}$ is the set of samples and $Y = \{Y_j \mid j = 1, \ldots, c\}$ is the associated label set. The X_j and Y_j denote the sample and label set for class j, respectively. Let $G = (V, E)$ be an undirected graph, where V is the set of nodes representing points, and E is the set of directed edges representing the INN level r relationship between points. The $SCCs^r$ in G can be represented as a set of sets, where each set contains nodes that form a strongly connected component:

$$SCCs^r = \cup_{j=1}^{c} SCCs^r_{Y_j} \tag{15}$$

where:
$SCCs^r$ is the set of Q SCC^r in the graph.
$SCCs^r_{Y_j}$ is the set of Q_j SCC^r in the graph with nodes within class j.
$Q = \sum_{j=1}^{C} Q_j$ denotes the total of SCC^r in the graph.
SCC^r is a strongly connected component in the graph, represented as a set of nodes.
Each SCC^r satisfies the property that for every pair of nodes (u, v) in SCC^r, there exists a directed path from u to v in the graph G, considering the edges representing the INN level r relationship. Their classes $y_{scc} \in Y_{scc}$ are assigned to the corresponding Y_j

Based on the general definitions, Algorithm 1 outlines the detailed implementation of our algorithm to construct $SCCs^r$. The algorithm begins by calculating $INN_r(\cdot)$ for each sample $x_i \in X_j$. After that, the algorithm builds the connection graph

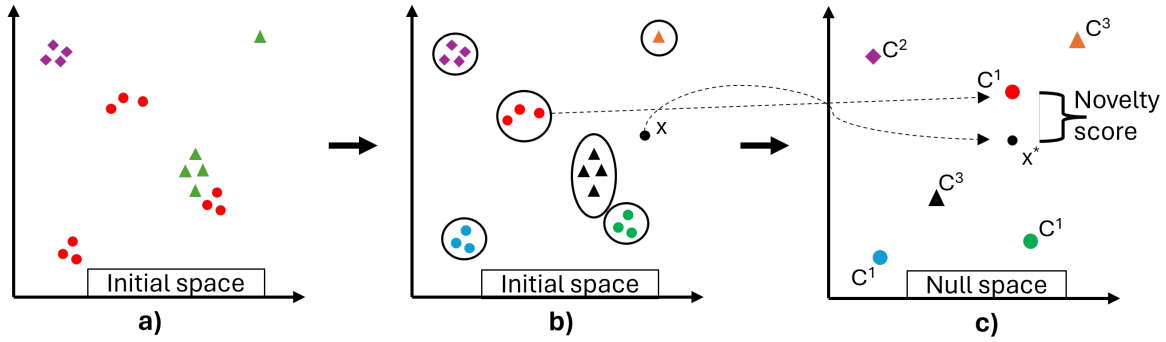

Figure 6: The overview of our proposed method. The sample of each class is grouped into sub-cluster and then mapped to a single point in null space.

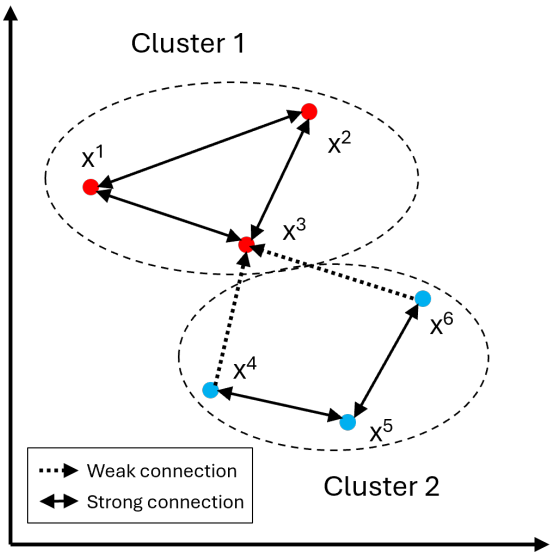

Figure 7: The illustration of Inverse nearest neighbour approach with R = 2.

G between samples within each class to prepare for constructing the local distribution. Each sample x_{ij} is considered as a node, and the connection between x_{ij} and its $INN_r(x_{ij})$ samples is an edge. After building the connection graph, the algorithm proceeds to group nodes with strong connections by applying the depth-first search technique. We named the newly formed group Strongly Connected Components (SCC). Samples within each SCC are then relabeled with the corresponding SCC-ID. This process iterates for each class j to reconstruct the training datasets into Q strongly connected components, with Q not assigned before.

4.2.2. *Neighbor Null Foley-Sammon Transformation*

After constructing the local structure data, the aim of the algorithm now is to map all samples within an SCC^r into a single point in discriminative space. To archive that, we perform the NFST algorithm based on the definition of the within-SCCs scatter matrix and between-SCCs scatter matrix as follows:

Algorithm 1: The localization data distribution algorithm

Input: S: the training dataset, r: the r value, C: the number of class
Output: S_N: the new training set

Initialize $NN_r = []$, $INN_r = []$, $X_{New} = []$, $y_{New} = []$
`/* Constructing local cluster for each class */`
for j in C **do**
  for i in N_j **do**
    $NN_r[i]$ = Find_Top_R_Nearest(X, i, r)
  end for
  for i in N_j **do**
    for k in $NN_r[i]$ **do**
      if $i \in NN_r[k]$ **then**
        $INN_r[i].append(k)$
      end if
    end for
  end for
  `/* Building graph and group sample into local cluster */`
  $Graph$ = Building_Graph(X_j, INN)
  X_{SCC}, SCC_{id} = Deep_First_Search$(Graph)$
  $X_{New}.append(X_{SCC})$
  $y_{New}.append(SCC_{id})$
end for
return X_{New}, y_{New}

Definition 3 (the between-SCCs scatter matrix):

$$M_{bscc} = \sum_{j=1}^{Q} Q_j(\sigma_j - \sigma)^\top(\sigma_j - \sigma) \tag{16}$$

where
$\sigma_j = \frac{1}{Q_j}\sum_{x \in SCC_j^r} x$ is the mean of the samples in SCC_j^r
$\sigma = \frac{1}{N}\sum_{x \in X} x$ is the mean of all training samples
Definition 4 (the within-SCCs scatter matrix):

$$M_{wscc} = \sum_{j=1}^{Q} \sum_{x \in SCC_j^r} (x - \sigma_j)^\top(x - \sigma_j) \tag{17}$$

Based on the Equation 16 and 23, the nNFST algorithm's goal is to find the discriminative features for the best separability by maximizing the Fisher criterion:

$$F(\nu) = \frac{\nu^\top M_{bscc} \nu}{\nu^\top M_{wscc} \nu} \tag{18}$$

By introducing the between-SCCs and the within-SCCs scatter matrix, the constraint of ν for $F(\nu)$ maximized is rewritten as:

$$\begin{aligned} \nu^\top M_{tscc} \nu > 0 \\ \nu^\top M_{wscc} \nu = 0 \end{aligned} \tag{19}$$

where:
$M_{tscc} = M_{bscc} + M_{wscc}$

$$M_{bscc} \nu = \lambda M_{wscc} \nu \tag{20}$$

Solving Equation 18 with the constraint defined in Equation 19 is a generalized eigenvector problem, which can be achieved by solving Equation 20. Here, ν is called the null projection direction (NPD).

In what follows, we specify when NPD exists, the number of them and how to calculate them. Let define the null space of M_{tscc} and M_{wscc} as:

$$\begin{aligned} Z_t &= \{z \in \mathbb{R}^D \mid S_{tscc}\, z = 0\} \\ Z_w &= \{z \in \mathbb{R}^D \mid S_{wscc}\, z = 0\} \end{aligned} \tag{21}$$

The respective orthogonal complement space is $Z_t^\perp$ and $Z_w^\perp$. It is obvious that:

$$\nu \in (Z_{tscc}^\perp \cap Z_{wscc}) \Rightarrow \{\nu^1, \nu^2, \ldots, \nu^k\} \in (Z_{tscc}^\perp \cap Z_{wscc}) \tag{22}$$

Hence, let us rewrite M_{wscc} and M_{tscc} under form of a real symmetric positive semi-definite matrix:

$$\begin{aligned} M_{wscc} &= \frac{1}{N} W_{wscc} W_{wscc}^\top \\ M_{tscc} &= \frac{1}{N} W_{tscc} W_{tscc}^\top \end{aligned} \tag{23}$$

where:
$W_{wscc} = (\bar{x_1^1}, \ldots, \bar{x_{N_1}^1}, \bar{x_1^j}, \ldots, \bar{x_{N_1}^j}, \bar{x_1^Q}, \ldots, \bar{x_{N_1}^Q})$ is the matrix consisting of the column vector centered by means of the corresponding classes.
$W_{tscc} = (\bar{x_1}, \ldots, \bar{x_N})$ is the matrix consisting of the column vector centered by the mean of data.

Consider the general assumption of NFST:

Assumption 1. *The given data samples $\{x_1, \ldots, x_N\} \in \mathbb{R}^D$ are considered to be in general positions. This implies that when the number of samples N is less than the dimensionality D ($N < D$), all samples are linearly independent. Moreover, when $N \geq D$, vectors in $\mathbb{R}^D$ have nonzero variance along every vector in subspace R^D.*

Assumption 2. *The total samples for each class of X are less than the dimensionality: $N_i < D, \forall i = 1, \ldots, Q$*

Denote $dim(\cdot)$ for the dimension; we have the following lemmas proven in (6):

Lemma 1. $dim(Z_{tscc}^\perp \cup Z_{wscc}) = rank(W_{tscc}) - rank(W_{wscc})$

Lemma 2. *Given the set $\{x_1, \ldots, x_N\} \in \mathbb{R}^D$ is linearly independent, then the column matrix of centered data has rank N-1: $rank(W_{tscc}) = N - 1$.*

Lemma 3. *Given the set $\{x_1, \ldots, x_N\} \in \mathbb{R}^D$ has non-variance with any vectors in subspace R^D, then the column matrix of centered data W_t has non-variance with any vectors in subspace R^D.*

From the Lemma 1, let the W_{wscc} expressed as: $W_{wscc} = (W_{wscc}^1, \ldots, W_{wscc}^Q)$, where W_{wscc}^j is the column matrix of data of class j centered by class mean. We have:

$$rank(W_{wscc}) \leq rank(W_{wscc}^1) + \cdots + rank(W_{wscc}^Q) \tag{24}$$

Based on the Assumption 2 and Lemma 2, we have $rank(W_{wscc}^j) = n - 1$, hence:

$$rank(W_{wscc}) \leq \sum_{Q}^{j=1} (N_j - 1) = N - Q \tag{25}$$

In the case $N \leq D$, we have that all the samples are linearly independent based on the Assumption 1, then $rank(W_{tscc}) = N - 1$ followed by Lemma 2. Based on Lemma 1 and Equation 26, we have:

$$dim(Z_{tscc}^\perp \cup Z_{wscc}) = rank(W_{tscc}) - rank(W_{wscc}) = Q - 1 \tag{26}$$

In the case $N > D$, following the Assumption 1 and Lemma 3, since we have $dim(W_t) = D$, then $rank(W_t) = D$. Based on Lemma 1 and Equation 26:

$$\begin{aligned} dim(Z_{tscc}^\perp \cup Z_{wscc}) &= rank(W_t) - rank(W_{wscc}) \\ &\geq rank(W_t) - Max(rank(W_{wscc})) \\ &= D - (N - Q) \end{aligned} \tag{27}$$

Based on the above proof, we can have the following corollary:

Corollary 1. *When $N < D + Q$, it always exists at least $Q - 1$ NPDs, which are the orthonormal basis of subspace $Z_{tscc}^\perp \cup Z_{wscc}$.*

To find the $\nu \in (Z_{tscc}^\perp \cap Z_{wscc})$, we first need to identify the basis vector of Z_{tscc} and then determine which vectors also belong to Z_{wscc}. We have $Z_{tscc}^\perp$ is the same as row space of M_{tcc}. Since M_{tcc} can be represented as W_{tcc}, then the row space of M_{tcc} is

also the row space of $W_{tscc}^{\top}$, which in turn is the column space of W_{tscc}. From here, we can guaranteed the vector $\nu \in Z_{tscc}^{\perp}$ by:

$$\nu = B\phi \tag{28}$$

with $B = \{b_i \mid i = 1, 2, \ldots, n; n \leq N\}$ is an orthonormal basis of the subspace W_{tscc}, which can be easily obtained using standard PCA or Gram-Schmidt orthonormalization. From now on, the problem of solving $\nu^{\top} M_{wscc} \nu = 0$ in Equation 19 is equivalent to computing the vector ϕ that satisfies Equation 29.

$$(B^{\top} M_{wssc} B)\phi = 0 \tag{29}$$

After finding the solution $\phi = \{\phi^1, \phi^2, \ldots, \phi^{Q-1}\}$ from Equation 29, we can compute the NPDs ν and arrange them as a project matrix P to calculate the discriminative features via the following equation:

$$\widetilde{x} = P^{\top} x_i \tag{30}$$

4.2.3. Kernel trick

Based on the Collary 1, a fundamental assumption is that dataset X has the number of samples (N) less than the sum of features (D) and classes (Q). However, this requirement is impractical in the intrusion detection domain, where N is typically larger than D and Q many times. Hence, to address this problem, we apply the kernel trick to map X into kernel space to perform the calculation on an high-dimensional space.

Let denote the kernel function as $k(x_i, x_j) = \left\langle \Phi(x_i), \Phi(x_j) \right\rangle$. Some kernel can be applied such as Gaussian, Polyomial, or Histogram intersection kernel. However, to guarantee existing NPDs, we apply the Radial basis function kernel, which maps data into an infinite-dimensional space. The formula of the RBF kernel is described as:

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\epsilon^2}\right) \tag{31}$$

where:
$\|x_i - x_j\|^2$ denotes the squared Euclidean distance between x_i and x_j.
ϵ is the kernel width parameter, controlling the smoothness of the kernel.

The orthonormal basis of the subspace $Z_t^{\perp}$ can be obtained by applying kernel PCA (22). Let define the matrix K size $N \times N$ is form by the inner products within mapped training data. The centered kernel matrix $\bar{K}$ is defined as:

$$\bar{K} = (I - 1_N)K(I - 1_N) \tag{32}$$

where:
I denotes the size $N \times N$ identify matrix.
1_N denotes the size $N \times N$ matrix which each elements equal $\frac{1}{N}$.

The matrix $\bar{K}$ is considered as a collection of pairwise inner products $\Phi(\bar{x}_i) = \Phi(x_i) - \sum_{k=1}^{N} \frac{\Phi(x_k)}{N}$. Subsequencely, We calcualte the eigendecomposition of $\bar{K}$ by:

$$\bar{K} = UAU^{\top} \tag{33}$$

where:
A denotes the size $N \times N$ diagonal matrix containing non-zero eigenvalues of $\bar{K}$.
U consists of the corresponding eigenvectors.

Then, with scaled eigenvectors $\bar{U} = UA^{\frac{-1}{2}}$, the eigenbasis B can be rewritten as:

$$b_i = \sum_{i=1}^{N} \bar{u}_i \Phi(\bar{x}_i) \tag{34}$$

To simplify the calculation, we do not directly calculate b_i. Instead, solving the Equation 29, with $E = B^{\top} M_{wscc}$, we solve the equivalent equation:

$$EE^{\top}\phi = 0 \tag{35}$$

With the $x_i \in X$ sorted by its class, the matrix E can be computed by:

$$E = ((I - 1_N)\bar{U})^{\top} K (I - L) \tag{36}$$

with the matrix L is a block diagonal matrix which consists of square blocks with size equal to corresponding $N_j, \forall j = 1, \ldots, Q$ and has a value equal to $\frac{1}{N_j}$ at each element. After computing the matrix E, we now can solve the Equation 35 to find ϕ and then get the NPDs $\widetilde{\nu}$ by the following equation:

$$\widetilde{\nu} = ((I - 1_N)\bar{U})\phi \tag{37}$$

Finally, from the Equation 30, we can project a data point x^* into the null space by $\widetilde{x^*} = (k^{*\top}\widetilde{\nu})^{\top}$, with k^* calculated by applying the kernel trick between x^* and all training data: $k^* = kernel(x^*, X)$.

4.2.4. Sparse Matrix Multiple

It is easy to see that the matrix L is a sparse matrix with only the sample in square blocks on the main diagonal having values. Therefore, we consider the form L in the form of a sparse matrix when implemented. The matrix can now be described as:

$$L = \text{csr\_matrix}(data, indices, indptr, (N, N)) \tag{38}$$

where:
data array stores the non-zero elements of the matrix L in row-major order.
indices array stores the column indices corresponding to the non-zero elements in the *data* array.
indptr array stores the indices that point to the start of each row in the *data* and *indices* arrays.
(N, N) denotes the size of the matrix.

This representation reduces both the memory usage and the computational cost of matrix operations, particularly when the total training sample increases.

4.3. The algorithm implementation

Based on the theory analysis provided in Subsection 4.2, Algorithm 2 and Algorithm 3 respectively outline the step-by-step procedures for training and testing with NFS. The training process includes two phases: localizing the data structures and then training the discriminative algorithm to get the projection vectors and set of base points *Base_point*, which represent each class in the null space. In the testing phase, novelty scores for each sample are determined as follows:

$$Nolvety\_scores(x_i) = Min(d(x_i^*, bases_j)), \forall j = 1, \ldots, Q \quad (39)$$

where:
x_i^* is the projection of x_i in the null space, calculated by Equation 37

After calculating the novelty scores, a hard threshold is specified based on the zero value and the min distance of x_i^* to the base points. If the novelty scores is larger than the threshold, x_i is determined as novelty samples. Otherwise, they are assigned to the class to which SCC^r they belong:

$$y_{pred}(x_i) = y_{scc}(\arg\min(d(x_i^*, bases_j))), \forall j = 1, \ldots, Q \quad (40)$$

Algorithm 2: The training process of our proposed algorithm

Input: S_{train}: the training dataset, r: the r value, C: the number of class

Output: *Base_point*: the base point represent for each class in the null spaces, $\widetilde{\nu}$: the set of project vectors, S_{scc}: the training set corresponding by SCC

`/* Constructing local cluster for each class */`

1: Get the local structure of data: S_{scc} = Localize_data($Strain$,r, C)
`/* Finding the NPDs vectors */`
2: Calculating the centered kernel matrix $\bar{K}$ as Equation 32
3: Obtain the scaled eigenvectors $\bar{U}$
4: Calculating the eigenbasis B as Equation 34
5: Sorting the samples in S_{scc} by its classes
6: Calculating the matrix E as Equation 36
7: Obtain ϕ from Equation 35
8: Obtain the NPDs $\widetilde{\nu}$ from Equation 37
9: Project S_{scc} to nullspace and get the base points *Base_point* represent for each class.
10: **return** $\widetilde{\nu}$, *Base_point*, S_{scc}

Algorithm 3: The test process of our proposed algorithm

Input: S_{scc}, S_{test}: the test dataset, *Base_point*, $\widetilde{\nu}$

Output: *y_pred*: the model predictions, *Novelty_scores*

1: Calculate the kernel inner project between S_{test} and S_{scc}
2: Obtain S_{test}^* by mapping S_{test} into the null space by Equation 37
3: Calculate the distance between S_{test}^* and *Base_point*
4: Obtain the *Novelty_scores* as Equation 39
5: Assign the class prediction *y_pred* based on the novelty score as Equation 40
6: **return** y_{pred}, *Novelty_scores*

5. Experiment Set Up

5.1. Datasets

In this study, we comprehensively evaluate our proposal by conducting experiments on four widely used datasets in the domain of attack detection: BoT-IoT, CIC-IoT2023, N-BaIoT, and ToN-IoT. These datasets are specifically designed to simulate IoT network environments and include a range of up-to-date attacks. A summary of each dataset is provided in Table 3, and further details are described below:

Table 3: The summary of datasets used for evaluating the proposed model.

Dataset	No. Classes	No. features	No. sample
ToN_IoT	10	21	9,507
N_BaIoT	9	115	9,000
BoT_IoT	6	21	10,118
CIC_IoT2023	6	46	34,000

***BoT-IoT*(23)**: This dataset was provided by the Cyber Range Lab of UNSW Canberra to support research aimed at enhancing IoT network security. It contains network traffic data collected from IoT devices infected with different types of botnet malware, which executes a variety of attack scenarios, including distributed denial of service (DDoS), brute-force attacks, and scan attacks. The network traffic was captured within a realistic network environment and analyzed to extract over 20 features related to network information, flow rate, and volume.

***ToN-IoT*(24)**: This dataset was also provided by the Cyber Range Lab of UNSW Canberra to support the development of artificial intelligence-based security solutions for industrial IoT networks. It consists of benign network traffic generated by legitimate IoT devices under normal operating conditions. Moreover, the dataset includes diverse IoT applications and communication protocols commonly used in IoT deployments.

***N-BaIoT*(25)**: This dataset captures the typical behavior of various IoT devices, including sensors, actuators, and smart home appliances. It covers a wide range of IoT applications and communication protocols. Additionally, the dataset focuses on various botnet attacks originating from two well-known botnet families: Mirai and Bashlite.

***CIC-IoT2023*(26)**: This is a comprehensive dataset provided by the Canadian Institute for Cybersecurity (CIC) to evaluate intrusion detection systems in IoT environments. It contains a wide range of IoT devices and different cyber-attacks. In this research, these attacks are grouped into five attack categories: DDoS/DoS, scan, web attacks, spoofing, and botnet attacks.

5.2. Experiment training/testing strategy

To evaluate our detection performance of the proposed model, we conduct experiments in three contexts: classification, multiclass novelty detection, and novelty detection. In more detail, the setup for each context is described below:

- ***Classification***: The model is trained with full-label training set and is evaluted on full-label testing set.
- ***Multiclass novelty detection***: For each dataset, we partition it into test cases corresponding to the total number of attacks. Subsequently, each type of attack is chosen as a novelty attack and excluded from the training set of the corresponding test cases. The testing set is fixed in all test cases, which contain all classes. The model is trained and tested for in each test case to evaluate its performance in multiclass novelty detection.
- ***Novelty detection***: The test cases and training/testing set are set as multiclass novelty detection context. However, all attacks are grouped as a super-label.

5.3. Evaluation Metric

The evaluation of a network intrusion detection system typically relies on metrics derived from the confusion matrix, such as True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). However, these conventional metrics may not accurately assess the system's ability to detect unknown attacks. Therefore, we used two specific metrics - Novelty Detection Rate (NDR) and Novel Area Under Curves (N-AUC) - to evaluate the system's performance in detecting unknown attacks. These metrics are calculated from True Unknown (TU) and False Unknown (FU), which parallel TP and FN, respectively. Besides that, we also use conventional metrics such as Accuracy, True Positive Rate (TPR), F1-score, Fall Positive Rate, and Matthew correlation coefficient for benchmarking our proposed models. These metrics offer a clearer depiction of the system's capability to detect network attacks. We describe the formula for each metric below:

- Accuracy (ACC):

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

- Recall - True Positive Rate (TPR):

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

- False Alarm Rate - False Positive Rate (FPR):

$$FPR = \frac{FP}{FP + TN}$$

- F1-score:

$$\text{F1-score} = \frac{2 * TPR * PPV}{TPR + PPV}$$

- Novelty Detection Rate (NDR):

$$NDR = \frac{TU}{TU + FU}$$

- Matthews Correlation Coefficient (MCC):

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

5.4. Software and hardware environment configuration

We run our experiments on Python 3.10.13 environments and its well-known libraries. The training server was using Ubuntu 22.04 with a 13th Gen Intel(R) Core(TM) i5-13400F processor and 32GB of RAM. Since there are no public Python sources for the model NFST, to ensure the model work within the intrusion detection system, we implement the version uising kernel trick followed by its original publications (3). In the following experiments, we denote the NFST model by KNFST. The implementation of other competitive models is provided by the open-framework PyOD (27; 28) version 1.0.8 which has detailed documentation on GitHub [1]. Besides, we used Sklearn library [2] version 1.3.2 for pre-processing data and calculating evaluation metrics.

6. Experiment Results and Discussion

This section focuses on evaluating the efficacy of our proposed model within the multiclass novelty detection context across well-known datasets. In addition, we examine the impact of key parameters on the model's detection performance. We also performed comparative experiments to highlight the superior detection performance of our proposed model compared to baseline models. The detailed setup of the experimental and datasets are provided in Section 5.

6.1. Main Models results

Firstly, we present the performance of our proposed model in classifying attacks within the context of training full classes. Subsequently, we illustrate the model's accuracy in the multiclass novelty detection task. Finally, we demonstrate the reducing of runtime through the implementation of sparse matrix operations.

6.1.1. Classification results

Table 4 showcases the high multi-class classification performance of the proposed model across four benchmarking datasets. The findings indicated that the proposed reached remarkable accuracy on all datasets with, accuracy rates ranging from 97.12% to 99.56% and the MCC scores from 0.8761 to 0.9838. Additionally, the false positive rate is kept below 1% for all datasets except the CIC-IoT2023, where it reaches 1.73%. Notably, the proposed model achieves the highest score

[1] https://github.com/yzhao062/pyod
[2] https://scikit-learn.org/

Table 4: The performance of proposed model on classifying attacks.

Datasets	K	Detection performance.					
		MCC	ACC	TPR	FPR	PPV	F1
BoT-IoT	**5**	**0.9838**	**99.56**	98.31	**0.27**	**98.49**	**98.39**
	10	0.9826	99.52	98.22	0.29	98.33	98.27
	15	0.9826	99.52	98.31	0.29	98.23	98.26
	20	0.9826	99.52	98.22	0.29	98.33	98.27
	30	**0.9838**	**99.56**	**98.32**	**0.27**	**98.49**	**98.39**
N-BaIoT	5	0.9444	98.86	94.89	0.64	95.91	94.68
	10	0.9451	98.88	94.94	0.63	95.96	94.74
	15	**0.9514**	**99.00**	**95.50**	**0.56**	**96.72**	**95.33**
	20	0.9483	98.94	95.22	0.60	96.41	95.01
	30	0.9483	98.94	95.22	0.60	96.41	95.01
ToN-IoT	**5**	**0.9306**	**98.75**	**93.53**	**0.69**	**94.04**	**93.70**
	10	0.8988	98.17	90.77	1.02	91.25	90.72
	15	0.8954	98.10	90.77	1.05	90.75	90.39
	20	0.8808	97.84	89.45	1.20	89.51	89.15
	30	0.8733	97.70	88.68	1.28	89.01	88.27
CIC-IoT2023	5	0.8689	96.95	85.02	1.83	85.83	85.35
	10	**0.8761**	**97.12**	**85.78**	**1.73**	**86.65**	**86.16**
	15	0.8676	96.90	85.10	1.86	85.12	85.02
	20	0.8649	96.83	84.77	1.90	84.91	84.71
	30	0.8677	96.90	85.23	1.86	85.24	85.14

on the dataset Bot-IoT and the lowest score on the dataset CIC-IoT2023, with a difference of about 2% in accuracy rate. This difference could result from the increased complexity of the CIC-IoT2023 dataset, which encompasses a greater variety of subclass attacks compared to the Bot-IoT dataset.

In addition to its high accuracy, Table 4 illustrates the consistent performance of the proposed model across different K values for each dataset, with only minor variations in scores. For instance, in the Bot-IoT dataset, the MCC remains stable at approximately 0.983, or the MCC remains approximately at 0.8761 on dataset CIC-IoT2023. However, there is a slight fluctuation in dataset ToN-IoT, where the MCC ranges from 0.8733 at K=30 to 0.9306 at K=5, which results from the highest number of classes and lowest number of features. These results indicate that the selection of the K value may be affected on specific datasets. Besides, the results also show that lower K values tend to return better results,

6.1.2. Multiclass novelty detection results

Figures 8 depict the aggregate classification performance of the proposed model in addressing the multiclass novelty detection task across each benchmark dataset at various K values. The line chart illustrates that models achieve a high classification performance in all datasets with the best average MCC values ranging from above 0.825 to approximately 0.975. Specifically, the graph shows that the model reaches its peak MCC on dataset BoT-IoT while maintaining consistently high performance across all K values. Whereas, there a slight fluctuations in other datasets, with variances of approximately 0.05 points between the best and worst MCC values within each dataset. These results underscore the stabilization of the model with low variance at different K values. However, the results also show no linear relationship between the K-value and accuracy for all data sets.

Figures 9 and 10 depict the aggregate novelty detection performance of the proposed model in detecting novelty attacks across each benchmark dataset at various K values. In particular, line chart 9 displays the NDR score while chart 10 presents the N-AUC scores. Both charts demonstrate that models achieve a high detection performance in all datasets with the highest average NDR values and the highest average NDR values ranging from above 0.92 to approximately 0.99. Notably, the model reaches its peak NDR and N-AUC on dataset BoT-IoT and hits its lowest point on dataset CIC-IoT2023. Besides, the model maintains consistently high performance across all K values on datasets BoT-IoT and N-BaIoT but shows fluctuations between K values in other datasets, with the highest variances of approximately 10% in NDR and 5% in N-AUC in the ToN-IoT dataset. These results underscore that the selection of K values has a notable impact on the model performance in specific datasets. Additionally, the results also show no linear relationship between the K-value and accuracy across all datasets.

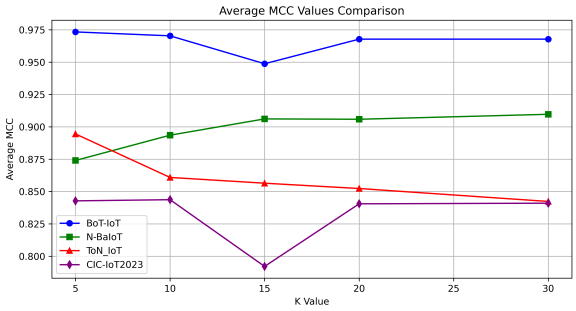

Figure 8: The comparison of MCC values between different K values of our proposed models in the MND context.

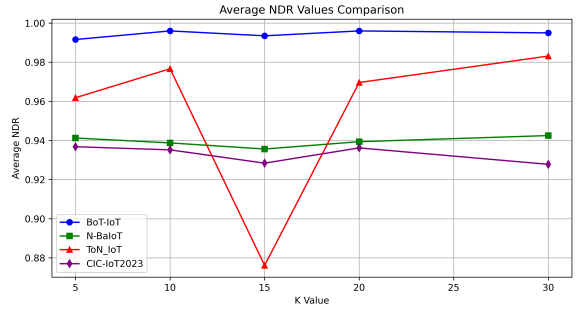

Figure 9: The comparison of NDR values between different K values of our proposed models in the MND context.

6.1.3. Sparse matrix results

Figure 11 depicts the robustness of using sparse matrix multiplication (SMM) in speeding up the training process by comparing the average training time across four benchmark datasets. In general, the application of SMM speeds up and stabilizes the training process of the proposed model. Particularly, employing SMM results in the lowest training time across different K values, approximately 340 seconds, while maintaining a stable training process with minimal variance across K values. In

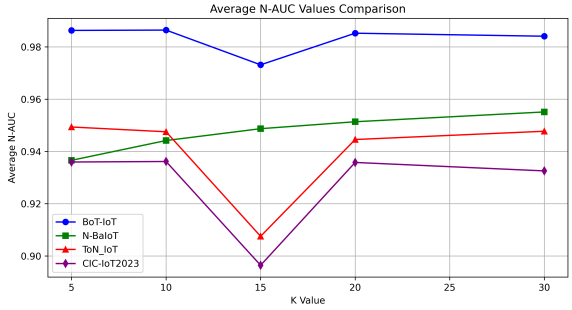

Figure 10: The comparison of N-AUC values between different K values of our proposed models in the MND context.

contrast, the model without SMM gives a significantly higher training time (600s compared to 340s when using SMM) and depicts a clear linear relationship between K values. Notably, it shows a significant reduction of training time when increasing the K value, from approximately 1600 seconds at K=5 to about 600 seconds at K=30. . Besides, using SMM helps the proposed model reduce its training time by 20% (from 420 seconds to 340 seconds) compared to the original NFST model, which does not use SMM.

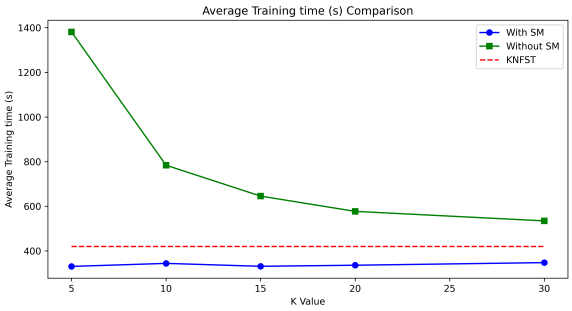

Figure 11: The comparison of training time between different K values of using SMM and without using SMM.

6.2. *Comparing with baseline models*

To better highlight our superiority in addressing the MND task, we compare our model with the state-of-the-art model, KNFST. The implementation of KNFST is introduced in research (3). Besides, we also conduct experiments on context of semi-supervised to compare with other state-of-the-art one-class models, including DeepSVDD, INNE, LOF, and OCSVM.

6.2.1. *Classification results*

Table 5 compares our model's classification performance with the KNFST model across benchmarking datasets. Overall, the results demonstrate the superior performance of our proposed solution compared to the KNFST model. Our approach improves the accuracy in all datasets with higher MCC and ACC values. Particularly on the BoT-IoT dataset, our model achieves an MCC of 0.9838 and an ACC of 99.56%, whereas

Table 5: The comparison of model detection performance when detecting without novelty attacks.

Dataset	Model	MCC	ACC	TPR	FPR	PPV	F1
BoT-IoT	Ours	0.9838	99.56	98.32	0.27	98.49	98.39
	KNFST	0.8347	95.44	85.70	2.74	84.47	84.86
N-BaIoT	Ours	0.9514	99.00	95.50	0.56	96.72	95.33
	KNFST	0.9081	98.05	91.22	1.10	94.37	89.84
ToN-IoT	Ours	0.9306	98.75	93.53	0.69	94.04	93.70
	KNFST	0.6444	93.42	67.87	3.66	63.29	63.33
CIC-IoT2023	Ours	0.8761	97.12	85.78	1.73	86.65	86.16
	KNFST	0.8673	96.90	84.70	1.86	84.80	84.71

the KNFST model demonstrates lower performance with an MCC of 0.8347 and an ACC of 95.44%. The biggest difference stays on the dataset ToN-IoT when KNFST only reaches an MCC of 0.6444 and the FPR of 3.66%, while our proposed model reaches 0.9306 of MCC and maintains an FPR of only 0.69%. Conversely, the smallest discrepancy stays on the dataset CIC-IOT2023 when KNFST reaches an MCC of 0.8673, slightly lower than our model by only 0.0088 points.

6.2.2. *Multiclass novelty detection results*

This section conducts a comparative analysis between our proposed model and KNFST in addressing the MND task within the contexts outlined in Section 3. The best results of each test cases in each dataset are aggregated and visualized using boxplots. Figure 12 compares the matthews correlation coefficient values for two models, KNFST and ours, across benchmarking datasets. In general, The graph demonstrates the robustness of our proposed model in the multiclass novelty detection context, which consistently achieves higher MCC values across all datasets. Particularly, on the BoT-IoT dataset, our model reaches higher mean MCC values, exceeding 0.95, compared to approximately 0.85 for KNFST. Similarly, the ToN-IoT dataset also depicts a significant difference as our model is above 0.2 mean MCC higher than KNFST. In the other datasets, our model maintains a slight advantage, with differences of above 0.02 points mean MCC value. Moreover, the graph also highlights the stability of our model compared to KNFST. The variance within test cases of our model is lower than KNFST in three out of four datasets.

Figures 13 and 14 depict a comparison of the NDR and N-AUC for two models, KNFST and ours, across benchmarking datasets. In general, the graph demonstrates the high accuracy of our proposed model in the detecting novel, which consistently achieves higher mean NDR and N-AUC values across all datasets. Specifically, our model reaches the highest mean NDR and N-AUC values on the BoT-IoT dataset, exceeding 0.975 for both indicators, compared to NDR of above 0.85 for and N-AUC of 0.9 for KNFST. Similarly, our proposed model reaches a higher mean NDR and mean N-AUC than KNFST in the other datasets, with both indicators reaching above 0.925. Meanwhile, the KNFST has the lowest mean NDR at above 0.825 and the lowest mean N-AUC at 0.875. Additionally, the results reveal that the median N-AUC of our model surpasses the third quartile of KNFST in three out of four datasets. These results indicate the superiority of our model over KNFST in de-

tecting novel attacks.

In summary, the results indicate the outstanding performance of our proposed approach to the KNFST in solving the MND task. The proposed model provides higher and more stable accuracy in various novelty attack contexts.

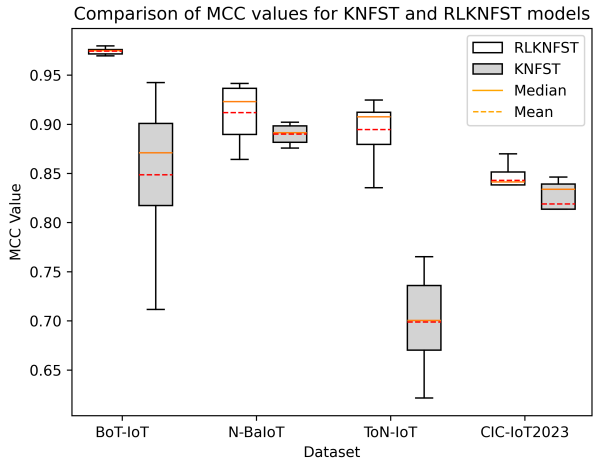

Figure 12: The Comparison of MCC values for KNFST and RLKNFST models.

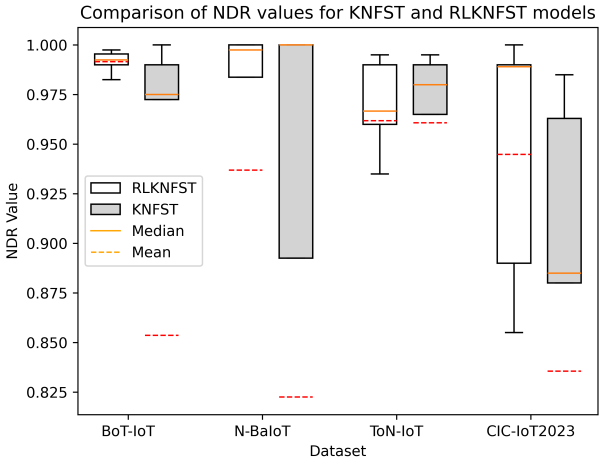

Figure 13: The Comparison of NDR values for KNFST and RLKNFST models.

6.2.3. Novelty results

To better understand the superiority of our models in detecting novelty attacks, this section provided the comparison results between our model with other one-class detection models in the context. Since the experiments focus on the ability of models to detect novelty attacks, hence, we aggregated the results of all test cases within each dataset, which drops an attack from training data for each test case. Then, we compared it with other one-class models, which were trained on normal traffic. All models were tested on a full-label testing set to detect attacks from normal traffic.

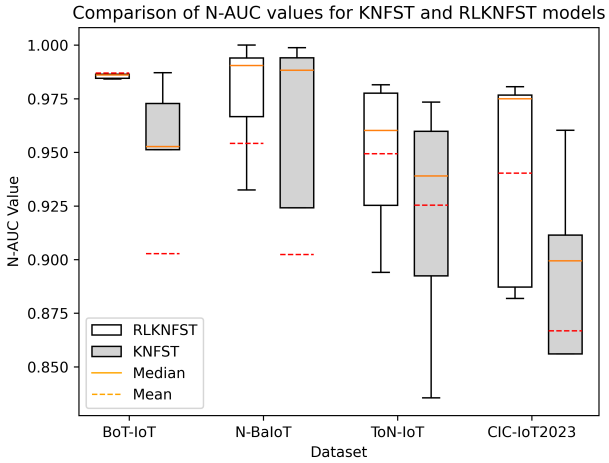

Figure 14: The Comparison of N-AUC values for KNFST and RLKNFST models.

Table 6 provided a comparison of our model with other competitors, highlighting the highest values in red and the second-highest values in blue. In general, the results indicated our proposed model reaches high accuracy in detecting novelty attacks and surpasses other competitors across all datasets and metrics. Specifically, in dataset BoT-IoT and ToN-IoT, our model achieves an MCC of 0.95 and 0.93 respectively, while the second-highest model only reaches an MCC of 0.46 and 0.44. In the datasets N-BaIoT, although the best competitor model achieves an MCC close to ours (0.95% compared to 0.97%), our model outperforms in terms of TPR and F1, with values of 99.53% and 99.62%, respectively, compared to 96% and 97.67% for the competitor. In the CIC-IoT2023, both our model and the best competitors have a relatively low MCC compared to other datasets with an MCC of 0.68 for our model and an MCC of 0.24 for the best competitor. However, our model still maintains high ACC, TPR, and F1, which are all above 94%.

In addition to the high accuracy, the result also indicates the stability of our models in different environments. While other models present fluctuating performance, our model consistently achieves the highest MCC values across all datasets. Particularly, the best competitor is model LOF, which reaches the second position in two out of four datasets and has a large fluctuation of MCC - range from 0.24 to 0.95. Whereas, our model archives the highest MCC in all datasets and keeps a shirk fluctuation of MCC - range from 0.68 to 0.97. Additionally, our model keeps a high and stable score in other metrics with all above 94% across all datasets.

7. Conclusion

In conclusion, this study introduces a novel algorithm called neighbor Null Foley-Sammon Transformation, designed to address the multiclass novelty detection challenge within the realm of network intrusion detection. The nNFST algorithm enhances the traditional NFST model by applying the inverse nearest neighbour algorithm for selecting representative points

Table 6: The comparison between our proposed model and another semi-supervised model on detecting attacks.

Model	BoT-IoT				N-BaIoT				ToN-IoT				CIC-IoT2023			
	MCC	ACC	TPR	F1	MCC	ACC	TPR	F1	MCC	ACC	TPR	F1	MCC	ACC	TPR	F1
DeepSVDD	0.38	75.30	79.62	62.45	0.93	98.56	93.50	96.12	**0.44**	**77.50**	**83.02**	**65.64**	0.20	67.13	78.22	46.78
INNE	0.43	77.87	82.60	65.17	**0.95**	99.00	95.50	97.36	0.19	49.06	65.38	43.59	0.20	67.50	78.89	47.05
LOF	0.45	**81.92**	82.84	**68.46**	**0.95**	**99.11**	**96.00**	**97.67**	0.32	66.56	75.59	56.31	**0.24**	**73.65**	**82.06**	**50.61**
OCSVM	**0.46**	78.90	**85.40**	66.77	0.34	65.67	76.53	56.68	0.10	30.83	56.76	29.79	0.16	62.25	72.80	43.79
Ours	**0.95**	**99.05**	**99.66**	**99.47**	**0.97**	**99.33**	**99.53**	**99.62**	**0.93**	**98.74**	**99.55**	**99.30**	**0.68**	**94.53**	**97.83**	**96.99**

and regulating the optimization function. This approach preserves both local and global data structures, mitigating the impact of singular points and improving accuracy when dealing with complex data. Moreover, the nNFST algorithm leverages the kernel trick to ensure accuracy and employs sparse matrix multiplication to reduce computational costs. Extensive experiments on various public datasets demonstrate the superiority of the nNFST algorithm compared to other baseline models across diverse attack scenarios. This research represents a significant advancement in securing computer network by addressing the MND challenge in network intrusion detection and open a potential approach for enhancing network security.

References

[1] Z. Yang, X. Liu, T. Li, D. Wu, J. Wang, Y. Zhao, and H. Han, "A systematic literature review of methods and datasets for anomaly-based network intrusion detection," *Computers & Security*, vol. 116, p. 102675, 2022.

[2] A. Aldweesh, A. Derhab, and A. Z. Emam, "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues," *Knowledge-Based Systems*, vol. 189, p. 105124, 2020.

[3] P. Bodesheim, A. Freytag, E. Rodner, M. Kemmler, and J. Denzler, "Kernel null space methods for novelty detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3374–3381, 2013.

[4] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," *arXiv preprint arXiv:2110.11334*, 2021.

[5] Y.-F. Guo, L. Wu, H. Lu, Z. Feng, and X. Xue, "Null foley–sammon transform," *Pattern recognition*, vol. 39, no. 11, pp. 2248–2251, 2006.

[6] T. F. Ali and S. Chaudhuri, "Theoretical analysis of null foley-sammon transform and its implications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 6445–6459, 2022.

[7] F. Zhu, J. Gao, J. Yang, and N. Ye, "Neighborhood linear discriminant analysis," *Pattern Recognition*, vol. 123, p. 108422, 2022.

[8] A. Khraisat and A. Alazab, "A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges," *Cybersecurity*, vol. 4, pp. 1–27, 2021.

[9] A. Thakkar and R. Lohiya, "A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions," *Artificial Intelligence Review*, vol. 55, no. 1, pp. 453–563, 2022.

[10] A. Aleesa, B. Zaidan, A. Zaidan, and N. M. Sahar, "Review of intrusion detection systems based on deep learning techniques: coherent taxonomy, challenges, motivations, recommendations, substantial analysis and future directions," *Neural Computing and Applications*, vol. 32, pp. 9827–9858, 2020.

[11] N. Liao and X. Li, "Traffic anomaly detection model using k-means and active learning method," *International Journal of Fuzzy Systems*, vol. 24, no. 5, pp. 2264–2282, 2022.

[12] A. Singh and J. Jang-Jaccard, "Autoencoder-based unsupervised intrusion detection using multi-scale convolutional recurrent networks," *arXiv preprint arXiv:2204.03779*, 2022.

[13] W. Wang, S. Jian, Y. Tan, Q. Wu, and C. Huang, "Robust unsupervised network intrusion detection with self-supervised masked context reconstruction," *Computers & Security*, vol. 128, p. 103131, 2023.

[14] M. J. Idrissi, H. Alami, A. El Mahdaouy, A. El Mekki, S. Oualil, Z. Yartaoui, and I. Berrada, "Fed-anids: Federated learning for anomaly-based network intrusion detection systems," *Expert Systems with Applications*, vol. 234, p. 121000, 2023.

[15] T. Zoppi, A. Ceccarelli, T. Capecchi, and A. Bondavalli, "Unsupervised anomaly detectors to detect intrusions in the current threat landscape," *ACM/IMS Transactions on Data Science*, vol. 2, no. 2, pp. 1–26, 2021.

[16] M. Soltani, B. Ousat, M. J. Siavoshani, and A. H. Jahangir, "An adaptable deep learning-based intrusion detection system to zero-day attacks," *Journal of Information Security and Applications*, vol. 76, p. 103516, 2023.

[17] X.-H. Nguyen, X.-D. Nguyen, H.-H. Huynh, and K.-H. Le, "Realguard: A lightweight network intrusion detection system for iot gateways," *Sensors*, vol. 22, no. 2, p. 432, 2022.

[18] S. M. Kasongo, "A deep learning technique for intrusion detection system using a recurrent neural networks based framework," *Computer Communications*, vol. 199, pp. 113–125, 2023.

[19] V. Hnamte and J. Hussain, "Dcnnbilstm: An efficient hybrid deep learning-based intrusion detection system," *Telematics and Informatics Reports*, vol. 10, p. 100053, 2023.

[20] G. S. C. Kumar, R. K. Kumar, K. P. V. Kumar, N. R. Sai, and M. Brahmaiah, "Deep residual convolutional neural network: An efficient technique for intrusion detection system," *Expert Systems with Applications*, vol. 238, p. 121912, 2024.

[21] X.-H. Nguyen and K.-H. Le, "Robust detection of unknown dos/ddos attacks in iot networks using a hybrid learning model," *Internet of Things*, vol. 23, p. 100851, 2023.

[22] W. Zheng, L. Zhao, and C. Zou, "Foley-sammon optimal discriminant vectors using kernel approach," *IEEE Transactions on Neural Networks*, vol. 16, no. 1, pp. 1–9, 2005.

[23] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.

[24] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "Ton_iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems," *Ieee Access*, vol. 8, pp. 165130–165150, 2020.

[25] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[26] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment," *Sensors*, vol. 23, no. 13, p. 5941, 2023.

[27] M. Jiang, C. Hou, A. Zheng, S. Han, H. Huang, Q. Wen, X. Hu, and Y. Zhao, "Adgym: Design choices for deep anomaly detection," *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[28] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao, "Adbench: Anomaly detection benchmark," *Advances in Neural Information Processing Systems*, vol. 35, pp. 32142–32159, 2022.