`NIDP`: Solving Feature Distribution Shifts in Network Intrusion Detection via Neural Pruning

Jiangtao Ding[1], Junli Zheng[1], Chengyang Mo[2], Zhicheng Xu[1], Hongbing Cheng[1]*
[1]*Department of Computer Science and Technology*
Zhejiang University of Technology
Hangzhou, China
[2]*China Jiliang University*
Hangzhou, China
{221123120216, 211122120039, 1111912015, chenghb}@zjut.edu.cn
874892887@qq.com

***Abstract*—Machine learning-based network intrusion detection(NID) systems often face concept drift in two forms: *unknown class emergence* due to the arising of previously unseen classes and *feature distribution shifts* due to the shifts in existing classes. While current approaches, such as drift detection and model retraining, can effectively address the challenge of unknown categories, they struggle to cope with shifts in feature distributions. Our preliminary analysis reveals that features like packet length vary considerably across different network configurations, resulting in unstable representations and degraded model performance.**

In this paper, we propose `NIDP`, a novel approach based on neural pruning designed to improve the robustness of NID models against changes in feature distribution. The `NIDP` method comprises three key components: 1) *Dynamic Drift Anchoring (DDA)*, which detects drift samples with variant features; 2) *Dual-Guided Feature Sculpting (DGFS)*, which leverages patterns identified by DDA to prune fluctuating features selectively; and 3) *Contrastive Representation Forging (CRF)*, which retrains the model to embed samples from the same category more cohesively. Together, these components improve the model's adaptability to dynamic network environments.

`NIDP` is evaluated on two widely used NID benchmarks, Kyoto2016 and CIC-IDS2017&2018, as well as two synthetic variants with amplified feature distribution shifts. Compared to state-of-the-art methods, it achieves F1 score improvements of 7.04% and 40.00% on the real-world datasets, and demonstrates the fastest performance recovery on the synthetic ones, highlighting its adaptability and robustness.

***Index Terms*—Network Intrusion Detection (NID), Neural Pruning, Feature Distribution Shifts, Variant Features**

I. Introduction

Traditional network intrusion detection (NID) solutions perform well when the training data distribution closely matches that of testing data. However, this assumes a closed-world environment, which is often violated when the network environment changes, a more realistic, dynamic network settings [1]–[7]. This violation, known as *concept drift*, typically occurs in two forms: (1) *Unknown class emergence*, where new classes, such as novel attack types in binary network traffic classification, appear in test data but are absent from the training set, challenging model generalization; and (2) *feature distribution shifts*, where sample features in existing classes (e.g., attack types in the training set) are subtly altered due to the evolution of benign applications or mutations in malware [1]–[4].

* Hongbing Cheng is the corresponding author.

For unknown class emergence, since new classes differ significantly from existing ones, advanced NID methods [2]–[4], [6], [8], [9] can accurately identify drift samples and retrain models to adapt to these new classes, effectively addressing this challenge. However, these methods often struggle with feature distribution shifts, where drift samples closely resemble known samples from other categories. This similarity complicates sample selection and ultimately reduces the effectiveness of these methods.

Our Approach. In this paper, we propose `NIDP`, a novel neural pruning-based approach designed to enhance the robustness of machine learning (ML)-based NID models against feature distribution shifts. Our approach is based on the key insight that features in network intrusion detection can be categorized as *variant* or *invariant*, similar to concepts in image detection tasks [10]–[13]. Variant features are sensitive to environmental changes and prone to distribution shifts, whereas invariant features remain stable despite changes in the environment. *By leveraging neural pruning to dynamically adjust model weights, we can minimize the impact of variant features, thereby improving the model's robustness to feature distribution shifts.*

To validate this insight, we conducted preliminary experiments on two widely adopted NID datasets: Kyoto2016 [14], [15] and CIC-IDS2017&2018 [16] datasets. We calculated the symmetric Kullback-Leibler (SKL) divergence for each feature across different categories and environments to identify variant features that may need to be pruned. The results showed that features, such as *packet length*, *packet number*, or *protocol types*, exhibited significantly higher SKL values across environments. These features are considered *variant*, indicating substantial feature distribution shifts when the network environment changes.

Specifically, `NIDP` performs model pruning in three stages:

- **Dynamic Drift Anchoring (DDA)**: We propose a *DDA* module that detects drift samples with variant features deviating from learned representations, treating them as

counterexamples to reveal unstable feature-class correlations. It adaptively weights these samples by their distributional shift magnitude to guide feature pruning.
- **Dual-Guided Feature Sculpting (DGFS)**: We develop a *DGFS* module that creates binary masks guided by dual tasks (NID and environment recognition), selectively pruning variant features using DDA's identified patterns while amplifying invariant cross-environment connections. This induces sparsity to enforce invariant feature consolidation.
- **Contrastive Representation Forging (CRF)**: After pruning, we retrain the model using Supervised Contrastive (SupCon) loss [17] as a regularization technique. This approach encourages the model to embed samples from the same class more closely, further enhancing performance and supporting robust adaptation to feature distribution shifts.

Challenge. Although the aforementioned methods seem feasible, directly applying neural pruning techniques to NID tasks is challenging. Existing neural pruning methods are primarily used in image detection, where they focus on discrete environmental changes (spatial variations). In contrast, NID tasks involve temporal variations, where network traffic cannot be easily classified into discrete environments or time segments [14]–[16], [18]. To address this issue, we redesign the environment recognition task in our Dual-Guided Feature Sculpting (DGFS) module as a regression problem. This adjustment avoids treating temporal variations as discrete environments and accounts for the similarity of traffic in adjacent periods, making our method more suitable for NID.

Contributions. In summary, The contributions of this paper are as follows:

- We introduce `NIDP`, a novel neural pruning-based approach that enhances the robustness of network intrusion detection (NID) systems against feature distribution shifts by reducing the influence of variant features on model predictions.
- We propose the *Dynamic Drift Anchoring*, *Dual-Guided Feature Sculpting*, and *Contrastive Representation Forging* modules. This approach enhances detection performance under feature distribution shifts by restructuring the model to depend more on invariant features, enabling more reliable predictions.
- We conduct experiments on two widely used NID benchmarks, Kyoto2016 and CIC-IDS2017&2018, as well as two synthetic datasets derived from them to introduce more pronounced feature distribution shifts. Compared to state-of-the-art NID methods, `NIDP` achieves average F1 score improvements of 7.04% and 40.00% over the strongest baseline on the real-world datasets. On the synthetic datasets, it demonstrates the fastest performance recovery under distribution shifts, further highlighting its adaptability and robustness.

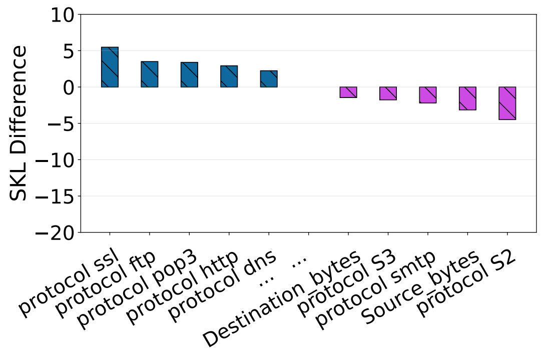
II. Background & Related Work

Managing concept drift in network intrusion detection. In network intrusion detection (NID), concept drift frequently disrupts the performance of existing methods, requiring continuous updates to adapt to evolving network environments [3], [4]. To address this, advanced NID methods often employ algorithms to detect drift samples and adapt to concept drift. TRANS [8], [9] calculates both *credibility*, which compares a sample's similarity to its most relevant class, and *confidence*, which measures its dissimilarity to the next most relevant class. Samples with both low credibility and confidence are flagged as drift samples. CADE [3] and HICL [4] use contrastive encoders to transform raw traffic features into low-dimensional latent space representations, ensuring that samples from the same class are closely clustered. Unlike CADE, which directly trains classifiers on raw traffic features, HICL focuses on the transformed features in the latent space and further explores the relationships between different malicious classes.

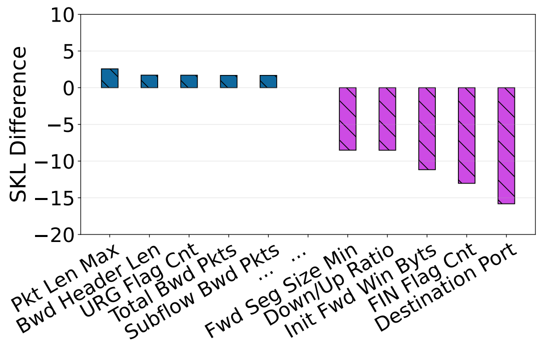
However, these methods primarily handle *unknown class emergence* but struggle with detecting *feature distribution shifts*, which often resemble other categories in training dataset due to significant changes in variant features. For example, shifts in these variant features in CADE [3] and HICL [4] result in unpredictable changes in the latent space, rendering sample selection strategies ineffective. In contrast, our method introduces neural pruning to better manage variant features, significantly improving detection performance under feature distribution shifts.

Neural pruning for enhancing model robustness. To improve the consistency of machine learning (ML) model predictions and enhance their robustness across different environments, one mainstream approach is neural pruning [12], [19]–[21], which typically consists of three main stages: Pre-training, Pruning, and Retraining. Each stage uses the same training dataset, with the possible addition of a few counterexamples. In the *Pre-training* stage, an L-layer neural network model $f(\theta; \cdot)$ is initialized with weights $\theta = w_1, \dots, w_L$. The model is then trained using the standard cross-entropy loss function, forming a model influenced by variant features for pruning. In the *Pruning* stage, each weight tensor w_i in the model is randomly assigned a binary mask $m_i \in 0, 1^{n_i}$ to indicate whether it is activated or not, where n_i corresponds to the length of w_i. This mask is either obtained based on criteria like connection sensitivity [22] or optimized through methods such as those in [19]. Our approach adopts the advanced optimization-based method, keeping the weight tensors w_i fixed while optimizing the binary masks m_i. To enable end-to-end training, the Gumbel-softmax trick [23] is applied as a sparsity constraint during training. Finally, in the *Retraining* stage, the pruned sub-network is retrained, with the flexibility to incorporate a regularization loss for improved robustness and overall performance.

Originally proposed by [24], neural pruning techniques have demonstrated that sparse neural networks can achieve strong generalization with proper initialization. Later studies



(a) Kyoto2016



(b) CIC-IDS2017&2018

Fig. 1. Feature distribution observations of traffic flows. The blue bar indicates that a feature's across-environment SKL divergence is greater than its within-environment SKL divergence, suggesting it is more likely a variant feature, while the purple bar represents the opposite case.

have further refined this approach: MRM [10] shows that sparse training and re-initialization improve out-of-distribution (OOD) generalization, while DCWP [11] identifies the optimal sub-network by oversampling samples with feature distributions differing from those of the training dataset. EVIL [12] strengthens its robustness and overall performance by dynamically raising invariant parameters and lowering variant parameters according to model gradients.

Our approach differs from the neural pruning methods mentioned above, which are primarily applied to image detection scenarios involving discrete spatial changes. In NID scenarios, temporal variations present a unique challenge, as network traffic cannot be easily segmented into distinct time intervals. To address this, we modify the environment recognition task into a regression problem, enabling the model to handle continuous traffic shifts and thus suitable for NID scenarios.

III. Preliminary Experiment

To validate the insight that features in network intrusion detection (NID) can be categorized as either *variant* or *invariant*, we conducted preliminary experiments on two widely adopted NID datasets, Kyoto2016 [14], [15] and CIC-IDS2017&2018 [16]. These experiments aim to identify features prone to distribution shifts in dynamic environments, referred to as *variant* features.

Experiment Setup. Specifically, for each feature in these two datasets, we calculate the difference between its across-environment symmetric Kullback-Leibler (SKL) divergence, which measures distribution shifts between different environments within the same category, and its within-environment SKL divergence, capturing distribution differences between categories within the same environment. The SKL divergence is defined as the average of the Kullback-Leibler (KL) divergence in both directions:

$$\mathsf{SKL}(P,Q) = \frac{D_{KL}(P||Q) + D_{KL}(Q||P)}{2} \tag{1}$$

Here, the KL divergence $D_{KL}(P||Q)$ quantifies the dissimilarity between a probability distribution P and a reference distribution Q as follows:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)} \tag{2}$$

Features with high across-environment SKL divergence but low within-environment SKL divergence demonstrate that the feature's distribution *varies significantly across environments yet remain consistent within the same environment across categories*. This pattern suggests that the feature is more sensitive to environmental changes, and should be treated as a variant feature. In contrast, invariant features, with low across-environment divergence and high within-environment SKL divergence, reflecting sensitivity to category changes but relative stability across different environments.

Experiment Results. We sort the features based on the difference between the across-environment SKL divergence and the across-category SKL divergence. As shown in Figure 1, features with higher difference values (on the left) indicate a higher probability of being *variant features*, while those with lower values (on the right) suggest to be *invariant features*.

For the Kyoto2016 dataset, following [15], we divide the data into three different environments: IID (2006-2010), $NEAR$ (2011-2015), and FAR (2014-2015) to calculate the above SKL divergences. As shown in Figure 1(a), features such as "*protocol ssl*", "*protocol ftp*" and "*protocol pop3*" have higher difference values, reflecting their sensitivity to environmental changes. Conversely, features like "*protocol smtp*", "*protocol S2*" and "*protocol S3*", as well as numbers like "*Source Bytes*" and "*Destination Bytes*" showing greater stability, and are more likely to be invariant features.

For the CIC-IDS2017&2018 dataset, we divide it into two different environments based on the year. As shown in Figure 1(b), features such as "*Pkts Len*", "*URG Flag*", "*Bwd Pkts*" indicate their sensitivity to environmental changes, and tend to be variant features. On the other hand, "*Fwd Bytes*", "*Down/Up ratio*", "*FIN Flag Count*", and "*Dst Port*" possess

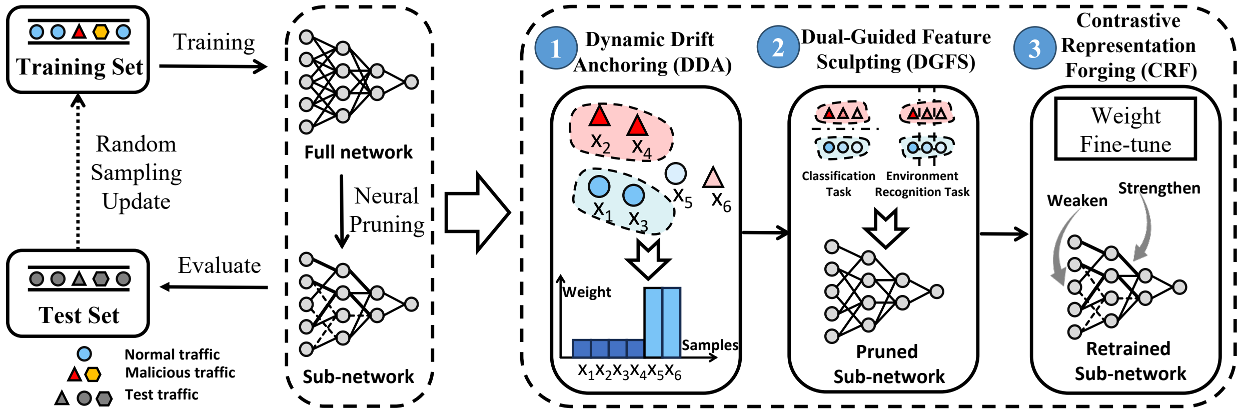


Fig. 2. The high-level workflow of NIDP. The Dynamic Drift Anchoring (DDA) module (①) identifies counterexamples that conflict with training samples and consequently increases their weight. The Dual-Guided Feature Sculpting (DGFS) module (②) focuses on both the target task (classification) and the environment recognition task to identify sub-network structures that rely solely on *invariant features*. The Contrastive Representation Forging (CRF) module (③) fine-tunes the model weights, using Supervised Contrastive (SupCon) loss to adjust the latent representations of samples, making those within the same class closer together and those in different classes further apart. In each iteration, NIDP tests a window of the dataset and randomly selects a subset of samples for model updates, enhancing adaptability to feature distribution shifts in network intrusion detection.

low SKL divergence difference, thus are more likely to be invariant features.

Overall, this analysis confirms the presence of distinct variant and invariant features in widely used NID datasets, forming the foundation for our neural pruning-based approach to enhance robustness in NID models.

IV. Design Detail

A. Design Overview

Our approach NIDP addresses feature distribution shifts in Network Intrusion Detection (NID) systems through a tripartite strategy that synergistically enhances model robustness. As illustrated in Figure 2, the framework comprises three coordinated components:

Dynamic Drift Anchoring (DDA). Detects feature distribution shifts through deviation analysis, then dynamically anchors critical drift samples by amplifying their training weights. This creates to guide pruning towards shift-affected model regions (§IV-B).

Dual-Guided Feature Sculpting (DGFS). Jointly train the main NID task and the environment recognition task. Prunes parameters that respond to variant features (flagged by environment recognition) while preserving invariant discriminative features for intrusion detection (§IV-C).

Contrastive Representation Forging (CRF). Retrains pruned model using Supervised Contrastive Loss, forcing samples of the same class (regardless of environments) into tight clusters. Strengthens decision boundaries against residual distribution shifts by enhancing feature compactness (§IV-D).

B. Dynamic Drift Anchoring (DDA)

This stage identifies *distribution drift samples* instances where feature patterns deviate significantly from training data norms and strategically amplifies their influence through adaptive reweighting. Unlike conventional NID methods that treat such anomalies as noise, we exploit their capability to reveal the instability of variant features, whose distributional shifts make them unreliable for classification.

We propose a two-phase approach to detect samples with variant features in DDA and apply dynamic weighting. In the first stage, a loss function with a bias amplifier identifies samples exhibiting feature distribution shifts caused by variant features. In the second stage, adaptive weight adjustments are applied to these samples, facilitating the subsequent DGFS process to remove features that lead to unstable training.

Phase 1: Drift Detection. A biased model trained with generalized cross-entropy (GCE) loss identifies hard-to-predict samples:

$$L_{GCE}(X_i, y_i) = \underbrace{\frac{1 - p_{y_i}(X_i)^q}{q}}_{\text{Bias Amplifier}} \cdot L_{CE}(X_i, y_i), \quad q \in (0, 1] \tag{3}$$

where $p_{y_i}(X_i)$ denotes the biased model's confidence for sample X_i's true label y_i. Lower q values intensify the model's focus on high-confidence predictions, thereby surfacing samples where variant features dominate its decisions.

We first compute the prediction confidence $p_{y_i}(X_i)$ for each sample's true label. Using the GCE loss, we then identify two distinct types of samples: *low-confidence correct samples* (where p_{y_i} is low but $\hat{y}_i = y_i$) and *high-confidence incorrect samples* (where p_{y_i} is high but $\hat{y}_i \neq y_i$). Both are labeled as drifted samples X_{drift}, as they indicate feature distribution shifts caused by variant features.

The bias amplifier $\frac{1-p_{y_i}(X_i)^q}{q}$ is inversely proportional to the prediction confidence p_{y_i}. For *low-confidence correct samples*, where the true label y_i is correct but the model's confidence p_{y_i} is low, the bias amplifier decreases their loss values. Conversely, for samples where the model's confidence p_{y_i} is high, the bias amplifier increases their loss values, increasing the model's focus on them. By adjusting $q \to 0$, we emphasize samples with high confidence, which often signal feature distribution shifts caused by variant features.

Phase 2: Adaptive Weight Allocation. We compute sample-specific pruning weights for mispredicted instances ($\hat{y}_i \neq y_i$):

$$W_i = \frac{M}{N_d} \cdot \exp\left(-\alpha \cdot p_{y_i}(X_i)\right) \tag{4}$$

where $\alpha = \ln \frac{N}{M}$, $N_d = |\mathcal{X}_{\text{drift}}|$ denotes drift sample count and M is total samples. This formulation: Normalizes by drift prevalence $\frac{N_d}{M}$ to prevent overemphasis on rare shifts and Modulates weights via prediction uncertainty $1 - p_{y_i}(X_i)$

These weights are applied to mispredicted samples, creating "gradient pressure points". During pruning, these pressure points influence the gradients, guiding the DGFS process to target regions of the model affected by variant features, known as variant feature pathways.

C. Dual-Guided Feature Sculpting (DGFS)

This module prunes the network using two complementary tasks: the target classification task for identifying *invariant features*, and an environment regression task for capturing *variant features*.

The target classification task aims to identify and retain *invariant features*, which remain stable and effective for classification across various conditions, such as different environments or data distributions. The environment regression task, conversely, is designed to capture *variant features*, which are subject to instability. One task preserves stable, invariant features to ensure robust classification performance, while the other detects variant features and flags them for pruning. These complementary tasks enhance the neural network's focus on reliable features while minimizing reliance on unstable ones.

The objectives are:

$$\min_{f_{\theta_{inv}}, h} L_{inv}\left(h\left(f_{\theta_{inv}}(X_i)\right), y_i\right) \tag{5}$$

$$\min_{f_{\theta_{var}}, g} L_{var}\left(g\left(f_{\theta_{var}}(X_i)\right), d_i\right) \tag{6}$$

Here, $\theta_{inv} = m \odot \theta$ and $\theta_{var} = (1 - m) \odot \theta$ represent the parameters of the invariant and variant subnetworks, respectively, with m as a binary mask applied via element-wise multiplication ($\odot$), and θ denoting the full network parameters. The functions $h(\cdot)$ and $g(\cdot)$ act as the final layers, mapping subnetwork outputs to classification labels y_i and environmental labels d_i, respectively.

We define the composite pruning loss, balancing both tasks, as:

$$L_{prune} = L_{inv} + \lambda_{prune} L_{var} \tag{7}$$

For NID's continuous traffic evolution, we redesign environment recognition as regression:

$$L_{var} = (\widehat{d_i} - d_i)^2, \quad \widehat{d_i} = g(f_{\theta_{var}}(X_i)) \tag{8}$$

where d_i is the timestamp of sample X_i. This approach captures local temporal correlations without manual segmentation, which is essential for non-stationary traffic.

Finally, we apply *unstructured pruning* by reassigning the neural network weight masks (independent Bernoulli variables) to 1 or 0 based on a pruning threshold (e.g. 0.5). The pruning mechanism follows a dual-path strategy: High-activation weights (≈ 1) are retained to extract invariant features, stable across environments; Low-activation weights (≈ 0) are pruned to capture variant features, sensitive to environmental changes.

D. Contrastive Representation Forging (CRF)

This stage forges robust representations by contrasting and refining the pruned sub-network. After establishing the sparse architecture through pruning, CRF retrains the model to simultaneously: 1) *Correct pruning-induced distortions* in feature embeddings, 2) *Forge invariant decision boundaries* through contrastive alignment.

The forging process is governed by:

$$L_{\text{retrain}} = \underbrace{L_{\text{inv}}}_{\text{Accuracy Preservation}} + \lambda_{\text{retrain}} \underbrace{L_{\text{reg}}}_{\text{Representation Forging}} \tag{9}$$

Contrastive Feature Alignment: The core *forging* mechanism uses Supervised Contrastive Loss [17] to restructure the feature space:

$$L_{\text{reg}}^{\text{SupCon}} = -\sum_{i \in \mathcal{B}} \frac{1}{|\mathcal{P}_i|} \sum_{p \in \mathcal{P}_i} \log \frac{\exp(z_i \cdot z_p / \tau)}{\sum_{a \neq i} \exp(z_i \cdot z_a / \tau)} \tag{10}$$

where $\mathcal{P}_i$ denotes positives (same-class samples) in batch $\mathcal{B}$, and $z_i = f_{\theta_{\text{inv}}}(X_i)$ are pruned-subnetwork embeddings. This *pulls* intra-class samples into tight clusters while *pushing* inter-class clusters apart, directly forging discriminative representations resistant to distribution shifts.

For enhanced forging stability, we optionally apply Sharpness-Aware Minimization (SAM) [25]:

$$L_{\text{reg}}^{\text{SAM}} = \max_{||\epsilon|| \leq \rho} L(f_{\theta_{\text{inv}} + \epsilon}(X)) \tag{11}$$

By optimizing for flat loss landscapes around θ_{inv}, SAM prevents the sparse architecture from overfitting to residual variant features.

Reinitialization-Enhanced Forging: Inspired by advanced pruning strategies [10], [26], we reset weights to initial values θ_0 before forging:

$$\theta_{\text{inv}} = m \odot \theta_0 \tag{12}$$

This allows the sparse architecture to escape pruning-induced local minima, enabling contrastive losses to more effectively reshape feature geometry from first principles (validated in §V-C).

V. Experiment

In this section, we first introduce our experimental setup in §V-A. Then, in §V-B, we compare the detection performance and the time and space overhead of our approach under *feature distribution shifts* against state-of-the-art methods, including three advanced NID methods and four neural pruning techniques. Finally, in §V-C and §V-D, we conduct ablation experiments to evaluate the effectiveness of key components in our approach and assess the impact of various hyperparameters on overall performance.

TABLE I
Traffic of Kyoto2016 dataset after randomly sampling.

Id	Family	# of Samples (IID)	# of Samples $(NEAR)$	# of Samples (FAR)
0	Benign	1,875,000(75.0%)	1,350,000(75.0%)	900,000(75.0%)
1	Attack	625,000(25.0%)	450,000(25.0%)	300,000(25.0%)

TABLE II
Traffic of CIC-IDS2017&2018 dataset after randomly sampling.

Id	Family	# of Samples (CIC-IDS2017)	# of Samples (CIC-IDS2018)
0	Benign	79,678(72.6%)	129,541(64.8%)
1	Bot	1,956(1.8%)	13,675(6.8%)
2	DoS GoldenEye	744(0.7%)	7,922(4.0%)
3	DoS Hulk	16,551(15.1%)	22,166(11.1%)
4	DoS SlowHTTPTest	377(0.3%)	6,592(3.3%)
5	DoS Slowloris	474(0.4%)	2,078(1.0%)
6	FTP-BruteForce	5,775(5.3%)	9,148(4.6%)
7	SSH-BruteForce	4,225(3.8%)	8,878(4.4%)

TABLE III
Traffic of Construct dataset Kyoto&BtoM.

Id	Family	# of Samples (IID)	# of Samples $(OOD\text{-}Test)$	# of Samples $(IID\text{-}Test)$
0	Benign	37,500(75.0%)	75,000(75.0%)	75,000(75.0%)
1	Attack	12,500(25.0%)	25,000(25.0%)	25,000(25.0%)

A. Experimental Setup

1) Implementation: We implement NIDP with over 2,000 lines of code using Python (version 3.7.16) and PyTorch 1.13.1+cu117. We implement NIDP with over 2,000 lines of code using Python (version 3.7.16) and PyTorch 1.13.1+cu117. We apply grid search to find the optimal hyperparameter configurations for our framework. All of the experiments of NIDP and other methods are performed on the same devices: Intel® Core™ i5-12600KF CPU @ 3.70GHz×10, 32GB RAM, Nvidia RTX 4060Ti GPU with 16GB of VRAM. For all methods, the classifier is updated after detection, and the labeling cost is set to 500 per window (50,000), approximately matching the HICL setting of 200 out of nearly 30,000 samples [4].

We optimized the hyperparameters of the MLP model using grid search. The search space included the learning rate ($\eta \in \{0.0001, 0.001, 0.01\}$) and hidden layer dimensions

TABLE IV
Traffic of Construct dataset CIC-IDS&MtoB.

Id	Family	# of Samples (IID)	# of Samples $(OOD\text{-}Test)$	# of Samples $(IID\text{-}Test)$
0	Benign	37,500(75.0%)	75,000(75.0%)	75,000(75.0%)
1	Attack	12,500(25.0%)	25,000(25.0%)	25,000(25.0%)

TABLE V
Layer configurations and parameter counts of MLP.

Layer	Type	Input Dim	Output Dim	Parameters
Hidden 1	GateMLP*	d_{in}	100	$d_{\text{in}} \times 100 + 100$
Hidden 2	GateMLP*	100	100	10,100
Hidden 3	GateMLP*	100	32	3,232
Output	Linear	32	2	66

* GateMLP is a hidden layer designed based on the theoretical foundation of DGFS.

($[100, 100, 32]$, $[200, 100, 64]$), and pruning threshold (0.3, 0.5, 0.7). Classification accuracy on the validation set was used as the evaluation metric to determine the optimal configuration. The core model, a multilayer perceptron (MLP) used for feature classification, is detailed in Table V. We configured it with a batch size of 256, 2000 epochs, the Adam optimizer with a learning rate of 0.001, and a pruning threshold of 0.5 based on a binarization criterion to determine node retention. Implementation details are available in our code repository [1].

2) Baseline: We compare the detection performance and computational overhead of NIDP with state-of-the-art machine learning (ML)-based network intrusion detection (NID) methods [27]–[29] and baseline methods, including Vanilla MLP, Updated MLP, and Random MLP, to evaluate its effectiveness in real-world scenarios. All state-of-the-art ML-based methods and NIDP are configured with their optimal hyperparameters to ensure a fair comparison. Vanilla MLP is trained only once and remains unchanged during the detection process. Updated MLP adopts an uncertainty-based (UC) active learning strategy [2] to identify samples with high prediction uncertainty as concept drift. Random MLP updates the model by randomly selecting samples, balancing adaptability to data distribution changes with computational simplicity.

3) Datasets: To validate detection performance in real-world scenarios, we use two widely used network intrusion detection (NID) datasets, Kyoto2016 and CIC-IDS2017&2018. Additionally, we construct two datasets that specifically extract real-world feature distribution shifts from these datasets, aiming to evaluate performance under more severe shifts. The detailed analyses are shown in Table I, II, III and IV. Note that additional NID datasets are also considered but are excluded from this study, a detailed explanation is provided in §VI.

- **Kyoto2016** [14], [15] is a long-term network intrusion detection dataset that captures continuous temporal changes. The dataset contains 10 years of traffic data and contains more than 1 million samples per month,

[1] https://github.com/Blank-stack/Sub-network-pruning-for-Network-Intrusion-Detection

TABLE VI
COMPARISON OF ACC AND F1 BETWEEN BASELINE ON **KYOTO2016** AGAINST REALISTIC CONCEPT DRIFT.

		Train	2011	2012	2013	2014	2015	Average
Vanilla MLP	Accuracy	0.96±0.00	0.94±0.01	0.90±0.02	0.71±0.06	0.39±0.04	0.27±0.01	0.64±0.03
	F1	0.93±0.01	0.89±0.02	0.81±0.05	0.65±0.05	0.43±0.02	0.40±0.00	0.63±0.03
Updated MLP	Accuracy	0.96±0.00	0.94±0.01	0.92±0.01	0.84±0.02	0.60±0.04	0.41±0.02	0.75±0.02
	F1	0.93±0.01	**0.91**±0.01	0.85±0.03	0.76±0.02	0.57±0.03	0.43±0.00	0.71±0.02
Random MLP	Accuracy	0.96±0.00	**0.95**±0.00	0.92±0.01	0.77±0.02	0.62±0.03	0.51±0.01	0.76±0.01
	F1	0.93±0.01	0.90±0.01	0.84±0.02	0.71±0.02	0.58±0.02	0.49±0.00	0.70±0.01
TRANS	Accuracy	0.96±0.00	**0.95**±0.01	0.93±0.01	0.84±0.02	0.58±0.01	0.41±0.02	0.75±0.01
	F1	0.93±0.01	**0.91**±0.02	0.87±0.02	0.77±0.02	0.55±0.00	0.43±0.01	0.71±0.01
CADE	Accuracy	0.96±0.00	0.94±0.01	0.92±0.01	0.76±0.02	0.51±0.01	0.49±0.05	0.72±0.02
	F1	0.93±0.01	0.89±0.02	0.84±0.00	0.70±0.01	0.50±0.01	0.47±0.03	0.68±0.01
HICL	Accuracy	0.96±0.00	0.94±0.01	0.91±0.01	0.73±0.00	0.42±0.01	0.29±0.00	0.66±0.00
	F1	0.93±0.00	0.88±0.02	0.84±0.01	0.68±0.01	0.45±0.01	0.40±0.00	0.65±0.00
NIDP	Accuracy	0.97±0.00	**0.95**±0.00	**0.95**±0.02	**0.86**±0.01	**0.70**±0.02	**0.82**±0.01	**0.85**±0.01
	F1	0.93±0.00	**0.91**±0.01	**0.89**±0.03	**0.78**±0.00	**0.61**±0.02	**0.69**±0.00	**0.76**±0.01

which are classified as benign or malicious. We randomly sampled the data while maintaining a 3:1 benign-to-malicious ratio [15], resulting in 50,000 samples per month. Consequently, this precess yields a total of 5.5 million samples over 10 years (the first year consists of only two months of data). *Traffic sampling* is a common approach in network intrusion detection, as including more traffic only increases computational overhead and has a negligible performance impact [3]. Following AnoShift [15], we divide the data into three environments: *IID*, *NEAR*, and *FAR*. We use *IID* as the training set and then test sequentially to simulate real-world time progression. The features *Service* and *Flag* are processed by one-hot encoding, while others are log-transformed and standardized, resulting in 52 features in total.

- **CIC-IDS2017&2018** [16] consists of traffic generated by executing known attacks daily. We treat CIC-IDS2017 and CIC-IDS2018 as two different environments and select attack types appearing in both datasets for evaluation. We sample and maintain a 3:1 benign-to-malicious ratio where it exceeds 10:1. CIC-IDS2017 is used as the training set and CIC-IDS2018 as the test set. For feature processing, we remove the features that appeared in only one dataset due to version differences in the data generator[2], then we log-transform and standardize the data, resulting in 76 features.
- **Kyoto&BtoM** and **CIC-IDS&MtoB** are datasets derived from Kyoto2016 and CIC-IDS2017&2018, specifically designed to highlight *feature distribution shifts*. Using t-SNE dimensionality reduction [30], we project the features into two dimensions and combine all malicious classes into a single class for visualization. We observe a substantial number of incoming samples (in *FAR* or CIC-IDS2018) resembling the known opposite class (in *IID* or CIC-IDS2017), with overlapping distributions. As a result, we construct the two datasets: Kyoto&BtoM and CIC-IDS&MtoB, and also maintain a 3:1 benign-to-malicious ratio. As shown in Table III and IV, the datasets are divided into two parts: *IID* and *OOD*. *IID* includes benign and malicious samples from known classes, while *OOD* consists of incoming benign (or malicious) samples that overlap with the opposite known class, supplemented by additional malicious (or benign) samples from the same environment as the known samples. Starting from the 50,000th sample, the data transitions from *IID* to *OOD*, and at the 150,000th sample, it switches back to *IID*. This design allows determining whether NID methods can adapt to feature distribution shifts and continue accurately detecting samples with stable distributions once the adaptation is complete.

[2]The data generator CICFlowMeter has updated 2017 and 2018, according to its documentation.

4) Metrics: We repeat the experiment at least 3 times to eliminate experimental contingency. By comparing the labels of unseen traffic with the corresponding predicted labels, we can evaluate the detection performance of the model. We use the widely adopted evaluation metrics F1-score (F1) and Accuracy as indicators of model performance [3], [4], [9], [31]. F1-score (F1) is defined as: $F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$, where $\text{precision} = \frac{TP}{TP+FP}$ and $\text{recall} = \frac{TP}{TP+FN}$. Accuracy represents the proportion of correctly classified samples to all testing samples: $\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$.

B. Performance and Overhead Comparison

1) Detection Performance Comparison: We compare the detection performance of `NIDP` with ML-based NID methods and neural pruning methods on two realistic datasets and two constructed datasets to validate the effectiveness. Kyoto2016 and CIC-IDS2017&2018 datasets contain long-term data, diverse network environments, and various malware attacks, making them ideal for evaluating performance under real-world NID environments. Kyoto&BtoM and CIC-IDS&MtoB are constructed from Kyoto2016 and CIC-IDS2017&2018 datasets, respectively, with test samples resembling the known opposite class, which can evaluate NID performance under more significant feature distribution shifts.

TABLE VII
COMPARISON OF ACC AND F1 BETWEEN BASELINE ON **CIC-IDS2017&2018** AGAINST REALISTIC CONCEPT DRIFT.

		Train	2018-Test1	2018-Test2	2018-Test3	2018-Test4	Average
Vanilla MLP	Accuracy	0.98±0.01	0.71±0.04	0.71±0.04	0.71±0.04	0.71±0.04	0.71±0.04
	F1	0.96±0.01	0.40±0.19	0.40±0.19	0.40±0.19	0.40±0.20	0.40±0.19
Updated MLP	Accuracy	0.98±0.01	0.71±0.04	0.73±0.07	0.72±0.08	0.76±0.10	0.73±0.07
	F1	0.96±0.01	0.40±0.19	0.39±0.25	0.39±0.28	0.50±0.32	0.42±0.25
Random MLP	Accuracy	0.98±0.01	0.71±0.04	0.75±0.08	0.73±0.04	0.78±0.05	0.75±0.02
	F1	0.96±0.01	0.40±0.19	0.64±0.11	0.65±0.08	0.69±0.09	0.60±0.06
TRANS	Accuracy	0.98±0.01	0.71±0.04	0.76±0.05	0.76±0.02	0.83±0.02	0.78±0.01
	F1	0.96±0.01	0.40±0.19	0.53±0.16	0.54±0.06	0.72±0.04	0.59±0.04
CADE	Accuracy	0.98±0.01	0.71±0.04	0.80±0.04	0.80±0.04	0.79±0.04	0.77±0.01
	F1	0.96±0.01	0.40±0.19	0.63±0.12	0.66±0.09	0.61±0.10	0.57±0.05
HICL	Accuracy	0.98±0.01	0.71±0.05	0.78±0.06	0.78±0.06	0.82±0.02	0.77±0.02
	F1	0.96±0.01	0.43±0.16	0.58±0.21	0.58±0.18	0.72±0.06	0.58±0.09
NIDP	Accuracy	0.98±0.00	**0.81**±0.05	**0.89**±0.03	**0.93**±0.05	**0.98**±0.01	**0.90**±0.01
	F1	0.96±0.01	**0.68**±0.10	**0.82**±0.06	**0.89**±0.09	**0.97**±0.01	**0.84**±0.02

In **Kyoto2016 dataset**, Table VI illustrates the performance declines of all methods from 2014 onwards due to significant feature shifts, particularly in *service type* and *number of bytes sent by the source IP*. AnoShift [15] attributes this decline to an increase in DNS traffic from 4% in 2013 to 37% in 2014. Our method effectively mitigates these feature shifts by adapting to evolving data patterns, achieving a 5.17% F1 score improvement over the best baseline, Random MLP, in 2014. Compared to other methods, our approach yields average F1 score gains of 20.6% over Vanilla MLP, and 7.04%, 11.76%, and 16.92% over Trans, CADE, and HICL, respectively, demonstrating superior robustness to feature changes.

In **CIC-IDS2017&2018 dataset**, we use CIC-IDS2017 for training and CIC-IDS2018 for testing. To evaluate each method's ability to handle network environment drift, we select four evaluation windows from the test dataset. As shown in Table VII, our method achieves rapid performance recovery by identifying invariant features through counterexamples unexplained by variant features, effectively adapting to shifts in traffic patterns. In contrast, vanilla MLP, CADE, and HICL exhibit poor performance from the second evaluation window onward, indicating ineffective update strategies against feature distribution shifts. Our method outperforms vanilla MLP by an average F1 score increase of 110% and surpasses Trans, CADE, and HICL by 42.37%, 47.36%, and 44.83%, respectively, demonstrating superior robustness to drift.

Table VIII and IX present the performance comparison on the **Kyoto&BtoM dataset** and **CIC-IDS&MtoB dataset**. As observed, all methods experience a sharp performance decline in the OOD-Test1 window across both datasets. Notably, in the Kyoto&BtoM dataset, the F1 score drops below 0.50, indicating detection performance worse than random labeling. However, in the OOD-Test2 window, Trans, CADE, and HICL show marginal improvement or further degradation, whereas NIDP , leveraging neural pruning, achieves significant performance recovery, with F1 score improvements of 47.61%, 44.18%, and 63.15% over Trans, CADE, and HICL, respectively. To verify the ability to detect non-shifted samples post-adaptation, we evaluate the IID windows (i.i.d. with training samples) and find that NIDP sustains high F1 scores, comparable to its pre-drift performance. These results demonstrate that NIDP rapidly adapts to feature distribution shifts, such as changes in traffic patterns, while maintaining robust detection accuracy for non-shifted samples.

In conclusion, NIDP demonstrates superior effectiveness in addressing concept drift for network intrusion detection. On real-world datasets such as Kyoto2016 and CIC-IDS2017&2018, NIDP achieves average F1 score improvements of 7.04% and 40.00% over the strongest baseline, respectively. On synthetic datasets, it exhibits the fastest performance recovery under distribution shifts, further underscoring its adaptability and robustness.

2) Overhead Comparison: We compare the time and space overhead of NIDP with those of the other methods on the Kyoto2016 dataset, a dataset that contains long-term NID traffic and exhibits realistic concept drift.

Time Overhead. We compare the training and update times of NIDP with those of other approaches in the end-to-end detection process. Training time refers to the time required to build the initial model, while update time denotes the duration needed for model updates in future iterations.

As shown in Table XI, NIDP significantly reduces update overhead for concept drift adaptation, with only a modest increase in one-time training. Specifically, HICL requires significantly more training time due to its complex feature embedding. NIDP also incurs slightly higher training time because of its pruning step. For updates, CADE and HICL are slower due to contrastive autoencoder training and complex sample selection. In contrast, NIDP uses random selection and integrates drift removal during training, resulting in minimal update time, making it well-suited for large-scale deployment.

Furthermore, in Table XI, we evaluated replacing the pruning mechanism in NIDP, our network intrusion detection framework, with MRM and EVIL. Compared to NIDP, MRM and EVIL reduced training time by 34.2s and 24.7s, respectively, while NIDP improved F1 scores by 16.7% and 23.8%

TABLE VIII
Comparison of ACC and F1 between baseline on **Kyoto&BtoM** against feature distribution shifts.

		Train	OOD-Test1	OOD-Test2	IID-Test1	IID-Test2	Average
Vanilla MLP	Accuracy	0.97±0.00	0.29±0.03	0.24±0.06	0.90±0.00	0.91±0.00	0.60±0.02
	F1	0.93±0.00	0.39±0.01	0.37±0.02	0.79±0.00	0.83±0.00	0.61±0.01
Updated MLP	Accuracy	0.97±0.00	0.29±0.03	0.37±0.03	0.91±0.00	0.91±0.00	0.64±0.01
	F1	0.93±0.00	0.39±0.01	0.42±0.01	0.81±0.01	0.84±0.00	0.63±0.01
Random MLP	Accuracy	0.97±0.00	0.29±0.03	0.65±0.02	0.91±0.00	0.90±0.00	0.70±0.01
	F1	0.93±0.00	0.39±0.01	0.56±0.01	0.80±0.01	0.84±0.01	0.65±0.01
TRANS	Accuracy	0.97±0.00	0.29±0.03	0.38±0.04	0.91±0.00	0.91±0.00	0.64±0.02
	F1	0.93±0.00	0.39±0.01	0.42±0.01	0.81±0.00	0.84±0.00	0.63±0.01
CADE	Accuracy	0.97±0.00	0.29±0.03	0.42±0.05	0.91±0.00	0.91±0.01	0.65±0.01
	F1	0.93±0.00	0.39±0.01	0.43±0.02	0.82 ±0.01	0.83±0.01	0.63±0.00
HICL	Accuracy	0.96±0.00	0.31±0.07	0.26±0.02	0.91±0.01	0.91±0.00	0.62±0.02
	F1	0.93±0.00	0.39±0.03	0.38±0.01	0.82±0.02	0.83±0.00	0.62±0.01
`NIDP`	Accuracy	0.97±0.00	**0.35**±0.01	**0.72**±0.05	**0.92**±0.00	**0.92**±0.00	**0.72**±0.01
	F1	0.93±0.00	**0.41**±0.00	**0.62**±0.03	**0.83**±0.00	**0.85**±0.01	**0.67**±0.01

TABLE IX
Comparison of ACC and F1 between baseline on **CIC-IDS&MtoB** against feature distribution shifts.

		Train	OOD-Test1	OOD-Test2	IID-Test1	IID-Test2	Average
Vanilla MLP	Accuracy	1.00±0.00	0.89±0.05	0.89±0.05	**1.00**±0.00	**1.00**±0.00	0.95±0.03
	F1	1.00±0.00	0.71±0.18	0.71±0.18	**1.00**±0.00	**1.00**±0.00	0.86±0.09
Updated MLP	Accuracy	1.00±0.00	0.89±0.05	0.97±0.01	**1.00**±0.00	**1.00**±0.00	0.97±0.01
	F1	1.00±0.00	0.71±0.18	0.95±0.01	**1.00**±0.00	**1.00**±0.00	0.92±0.05
Random MLP	Accuracy	1.00±0.00	0.89±0.05	0.97±0.02	**1.00**±0.00	**1.00**±0.00	0.97±0.02
	F1	1.00±0.00	0.71±0.18	0.94±0.05	**1.00**±0.00	**1.00**±0.00	0.92±0.06
TRANS	Accuracy	1.00±0.00	0.89±0.05	0.97±0.01	**1.00**±0.00	**1.00**±0.00	0.97±0.01
	F1	1.00±0.00	0.71±0.18	0.94±0.02	**1.00**±0.00	**1.00**±0.00	0.92±0.05
CADE	Accuracy	1.00±0.00	0.89±0.05	0.95±0.02	**1.00**±0.00	**1.00**±0.00	0.96±0.01
	F1	1.00±0.00	0.71±0.18	0.90±0.05	0.99±0.00	0.99±0.00	0.90±0.03
HICL	Accuracy	1.00±0.00	0.90±0.02	0.89±0.05	0.99±0.00	**1.00**±0.00	0.95±0.01
	F1	1.00±0.00	0.76±0.07	0.70±0.16	0.99±0.00	**1.00**±0.00	0.87±0.04
`NIDP`	Accuracy	1.00±0.00	**0.99**±0.01	**1.00**±0.00	**1.00**±0.00	**1.00**±0.00	**1.00**±0.00
	F1	1.00±0.00	**0.98**±0.01	**1.00**±0.00	**1.00**±0.00	**1.00**±0.00	**0.99**±0.00

over MRM and EVIL on the CIC-IDS2017 dataset.

Space Overhead. We also compare memory usage on both CPU and GPU across all approaches. As shown in Table XI, our method requires no additional memory, as the *Dynamic Drift Anchoring (DDA)* module eliminates drift without storing explicit drift samples. In contrast, CADE and HICL consume significantly more memory due to their use of contrastive autoencoders, which require storing both encoder parameters and gradients for backpropagation. Moreover, HICL further increases memory consumption by using a KDTree for sample correlation analysis, adding to its space complexity. As a result, our memory usage is comparable to TRANS, only 25% of the memory required by CADE and HICL.

C. Ablation Experiment

As shown in Table X, we compare `NIDP` with several variants in which individual components are excluded or modified to assess their impact on model performance.

Dynamic Drift Anchoring (DDA). As described in §IV, we evaluated variants applying network reinitialization, which resets model weights, and omitting the *DDA* reweighting process, which enhances feature discrimination via sample weighting. Network reinitialization reduced the average F1 score by 0.63% on public datasets and 1.21% on constructed datasets, while omitting *DDA* reweighting caused drops of 6.25% and 2.41%. These declines stem from the model's struggle to learn from outliers with unpredictable feature distributions, impairing effective pruning.

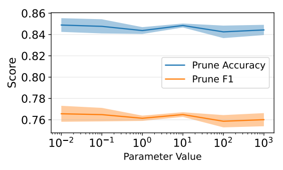
Dual-Guided Feature Sculpting (DGFS). We assessed the impact of omitting L_{var}, the Environment Recognition Task in the DGFS phase, which models environmental variations. Excluding L_{var} decreased the average F1 score by 1.25% on two public datasets and 1.81% on two constructed datasets, likely due to overfitting to diverse feature distributions. To address temporal environmental shifts in network intrusion detection, we modified the pruning phase's environment recognition task into a classification task, resulting in a 2.6% F1 score decline. This underscores the need to account for continuous network traffic variations.

Contrastive Representation Forging (CRF). In the CRF phase, which refines feature representations, we omitted the supervised contrastive (SupCon) loss [17], designed to align same-class latent representations and separate different-class ones, resulting in F1 score reductions of 1.25% and 1.81%

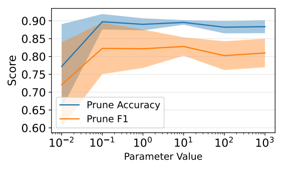
TABLE X
PERFORMANCE COMPARISON OF NIDP AND OTHER VARIANTS ON DIFFERENT DATASETS.

		Kyoto2016	CIC-IDS2017&2018	Kyoto&BtoM	CIC-IDS&MtoB
NIDP	Accuracy	**0.85**±0.01	**0.90**±0.01	**0.72**±0.01	**1.00**±0.00
	F1	**0.76**±0.01	**0.84**±0.02	**0.67**±0.01	**0.99**±0.00
NIDP with reinitial in DDA	Accuracy	0.84±0.00	0.90±0.02	0.71±0.01	0.99±0.01
	F1	0.75±0.00	0.84±0.05	0.66±0.00	0.98±0.01
NIDP without re-weight in DDA	Accuracy	0.82±0.00	0.87±0.03	0.71±0.01	0.99±0.01
	F1	0.74±0.00	0.76±0.07	0.65±0.00	0.97±0.01
NIDP without L_{var} in DGFS	Accuracy	0.84±0.01	0.89±0.02	0.70±0.01	0.99±0.00
	F1	0.75±0.01	0.83±0.06	0.65±0.00	0.98±0.01
NIDP without temporal labels in DGFS †	Accuracy	0.83±0.00	-	-	-
	F1	0.74±0.00	-	-	-
NIDP with $L_{\text{reg}}^{\text{SupCon}}$ in CRF	Accuracy	0.84±0.00	0.89±0.01	0.71±0.01	0.99±0.01
	F1	0.75±0.00	0.83±0.03	0.66±0.01	0.97±0.02
NIDP with $L_{\text{reg}}^{\text{SAM}}$ in CRF	Accuracy	0.82±0.01	0.80±0.01	0.71±0.02	0.87±0.02
	F1	0.71±0.01	0.64±0.05	0.66±0.03	0.59±0.08
NIDP with MRM	Accuracy	0.83±0.01	0.82±0.01	0.70±0.01	0.98±0.01
	F1	0.74±0.01	0.70±0.04	0.62±0.01	0.95±0.04
NIDP with EVIL‡	Accuracy	0.76±0.01	0.80±0.02	0.70±0.01	0.92±0.00
	F1	0.70±0.01	0.64±0.05	0.64±0.01	0.74±0.01

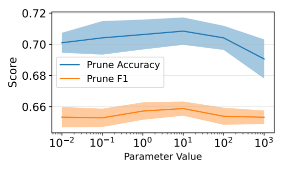
†Temporal labels were excluded from all datasets except Kyoto2016 due to constraints in dataset construction and format conversion.
‡In the prediction of CIC-IDS&MtoB's OOD-Test1, the correct prediction rate of positive samples is very low, only 7.1% (882 in 12,500) at most, resulting in a low F1-score.



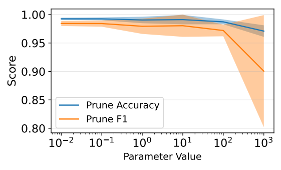
(a) Comparison on Kyoto2016



(b) Comparison on CIC-IDS2017&2018

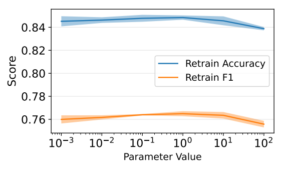


(c) Comparison on Kyoto&BtoM

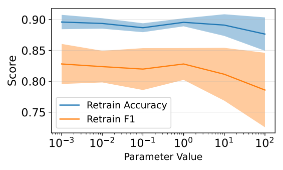


(d) Comparison on CIC-IDS&MtoB

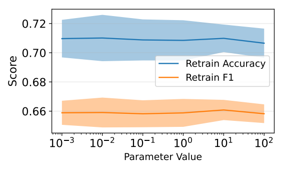
Fig. 3. The detection performance varies according to different λ_{prune} on the corresponding dataset.



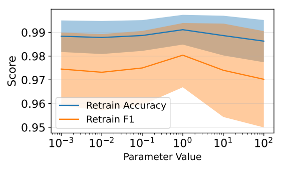
(a) Comparison on Kyoto2016



(b) Comparison on CIC-IDS2017&2018



(c) Comparison on Kyoto&BtoM



(d) Comparison on CIC-IDS&MtoB

Fig. 4. The detection performance varies according to different $\lambda_{retrain}$ on the corresponding dataset.

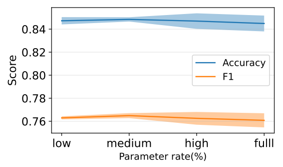
on public and constructed datasets. In contrast, applying the SAM regularization loss from EVIL severely degraded performance, with F1 scores dropping by 15.63% and 24.70% on these datasets. This substantial decline arises from SAM's gradient sharpness smoothing, which disrupts the temporal and contextual semantics of network traffic features.

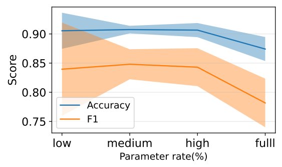
Other Weight Selection Methods We evaluated our unstructured pruning scheme against MRM and EVIL, which adjust weight selection differently, on the Kyoto2016 and CIC-IDS2017/2018 datasets. Replacing our scheme with MRM and EVIL led to F1 score declines of 2.64% and 7.89% on Kyoto2016, and 16.67% and 23.80% on CIC-IDS2017/2018, respectively. These results highlight the superior performance of our unstructured pruning scheme.

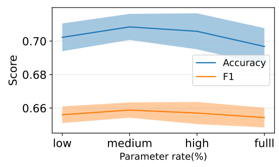
D. Hyperparameter Sensitivity

This section assesses the impact of different hyperparameters on model performance using the four datasets.

Hyperparameters in Regularization Loss. Figure 3 and Figure 4 show that the optimal performance is mostly achieved with λ_{prune} set to 10 and $\lambda_{retrain}$ set to 1. This configuration leads to stable predictions (low variance) and improved performance (higher scores). Notably, the performance drop remains minimal (less than 0.01) even when hyperparameters deviate by over 100-fold from their optimal values, demonstrating the model's strong robustness to hyperparameter selection.


(a) Comparison on Kyoto2016


(b) Comparison on CIC-IDS2017&2018


(c) Comparison on Kyoto&BtoM

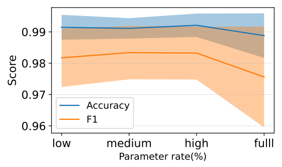
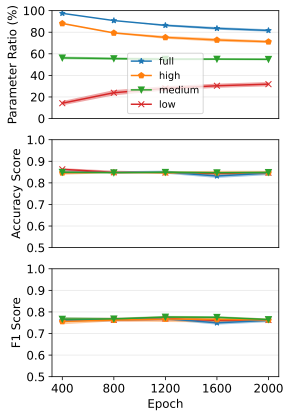
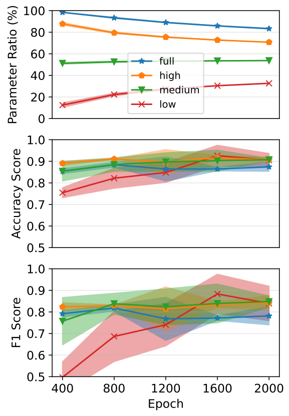
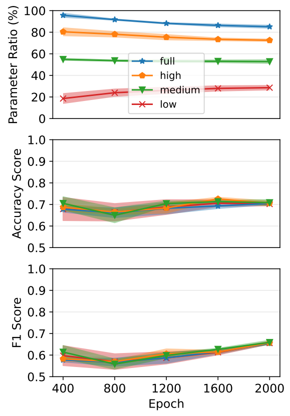

(d) Comparison on CIC-IDS&MtoB

Fig. 5. The detection performance varies according to different initial parameter ratios on the corresponding dataset.


(a) Evolution on Kyoto2016


(b) Evolution on CIC-IDS2017&2018


(c) Evolution on Kyoto&BtoM

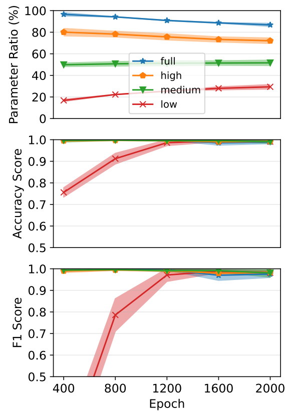

(d) Evolution on CIC-IDS&MtoB

Fig. 6. The changing trend of network parameters with the evolution of the pruning process on the corresponding datasets.

TABLE XI
TIME AND SPACE OVERHEAD BETWEEN NIDP AND OTHER METHODS ON THE KYOTO2016 DATASET.

Methods	Training Time(/s)	Updating Time(/s)	Space Use(/GB)†
Vanilla MLP	24.97	0.57	1.3 + 0.5
TRANS	24.97	668.34	1.3 + 0.5
CADE	24.97	842.03	4.4 + 0.5
HICL	251.55	417.01	6.8 + 2.7
NIDP-MRM	41.56	0.57	1.3 + 0.5
NIDP-EVIL	51.09	0.57	1.3 + 0.5
NIDP	75.76	0.57	1.3 + 0.5

†We measured peak memory usage, including both CPU memory and dedicated GPU memory, excluding dataset reading overhead.

Network Parameter Ratio. Figure 5 and Figure 6 illustrate the impact of varying initial parameter ratios ("full", "high", "medium", "low" corresponding to nearly 100%, 85%, 55%, and 15%, respectively) through network masking. For the network intrusion detection datasets we examined, model performance stabilizes at an initial parameter ratio of around 55% ("medium"), yielding higher and more consistent scores. Interestingly, this optimal ratio is notably higher than the typical 10-20% found in image recognition tasks [26]. When set to a "low" configuration (similar to ratios used in image recognition), performance scores fluctuate significantly and often reach a minimum. This difference likely arises because image pixels contain inherent redundancy, allowing for greater pruning, whereas network traffic features are carefully selected by experts, necessitating a higher retention of parameters.

VI. DISCUSSION

Preliminary Experiment's Setup. The preliminary experiment's setup differs from that of image identification tasks. Neural pruning methods [10], [12] assume that feature X is generated as $X = G(Z_{inv}, Z_{var})$, where Z_{inv} and Z_{var} represent invariant and variant features, respectively. We directly analyze traffic features because: 1. Traffic's statistical features carry practical significance, unlike the unstructured pixels in image tasks. 2. Existing feature selection methods directly remove features through static selection [32]–[34], dynamic selection [35]–[38], or their combinations [39].

Class Imbalance Issue. Following the AnoShift methodology [15], we set the benign-to-malicious ratio to 3:1 each month in the Kyoto2016 dataset. For the CIC-IDS2017&2018 dataset, we resample and maintain a 3:1 benign-to-malicious ratio where it exceeds 10:1. The 3:1 ratio is reasonable because it is a close approximation of the benign-to-malicious ratio in most real-world network environments.

Frequent Retraining. To mitigate concept drift in network traffic, where data distributions shift over time, NIDP employs periodic retraining. Although it incurs higher training overhead than competing methods, it achieves faster model updates. In deployment, lightweight data updates (0.57s) handle minor drift in most scenarios, while full parameter retraining (75.76s) addresses significant traffic shifts. This elastic mechanism ensures robustness during abrupt changes while minimizing routine overhead. Nonetheless, approaches such as CADE [3] also require retraining, and as shown in Table XI, the time required is relatively comparable to our method. As a result, the need for frequent retraining is within an acceptable range for achieving robust performance. To further reduce training overhead in addressing concept drift, we propose future optimizations. For example, incremental learning [40], leveraging online or mini-batch updates, incrementally adapts the model to minimize dependency on large-scale datasets, thus reducing training time.

Random Selection for Updating. We opt for random sample selection rather than active learning to update the model in our approach. Relying on the classifier's output for sample selection will amplify existing biases when facing benign evolution or malware mutations, hindering effective model updates. While exploring alternative sample selection methods to enhance pruning efficacy is promising, it is beyond the scope of this work. Future research could investigate these alternatives to improve neural pruning.

Other NID Datasets. To our knowledge, only seven other widely used datasets are available for evaluating NID performance [41], though each with notable limitations. The *CrossNet2021* dataset, introduced by ProGraph [5], includes traffic from two network environments to assess performance under varying conditions. However, we exclude it because it classifies samples into 20 distinct categories rather than simply benign or malicious, and its sample size of 7180 (4429+2751) is insufficient for robust analysis. Similarly, we do not use the *RWDIDS2022* dataset [1], the *Mirai* dataset [42], the *CIC-DDoS2019* dataset [43], the *UNSW-NB15-IDS* dataset [44], the *UGR'16* dataset [45] and the *NSL-KDD* dataset [46], as they lack concept drift, particularly feature distribution shifts, and typically span only a few days or are missing time labels.

VII. Conclusion

In this paper, we present NIDP, a neural pruning-based approach to mitigate the impact of *feature distribution shift* in network intrusion detection (NID). By integrating Dynamic Drift Anchoring, Dual-Guided Feature Sculpting, Contrastive Representation Forging modules, along with a time-regression-based environment recognition task, our method significantly enhances detection performance under dynamic network conditions. We evaluate NIDP on two widely adopted NID benchmarks—Kyoto2016 and CIC-IDS2017&2018—as well as two custom synthetic datasets derived from them to amplify feature distribution shifts. Compared to state-of-the-art approaches, NIDP achieves F1 score improvements of 7.04% and 40.00% on the real-world datasets, and exhibits the fastest recovery under distribution shifts in the synthetic settings, underscoring its adaptability and robustness.

VIII. Acknowledgement

This work was supported in part by the Key Program of Nature Science Foundation of Zhejiang Province under Grant LZ24F020007, in part by the National Nature Science Foundation of China under Grant 62072407, in part by the "Leading Goose Project Plan" of Zhejiang Province under Grant 2022C01086, Grant 2022C03139, and in part by the National Key R&D Program of China under Grant 2022YFB2701400, Supported by the "Tianchi Talent "Distinguished Expert Program of Xinjiang Province.

References

[1] Xian Wang. Enidrift: A fast and adaptive ensemble system for network intrusion detection under real-world drift. *Proceedings of the 38th Annual Computer Security Applications Conference*, pages 785–798, 2022.

[2] Giuseppina Andresini, Feargus Pendlebury, Fabio Pierazzi, Corrado Loglisci, Annalisa Appice, and Lorenzo Cavallaro. Insomnia: Towards concept-drift robustness in network intrusion detection. *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, pages 111–122, 2021.

[3] Limin Yang, Wenbo Guo, Qingying Hao, Arridhana Ciptadi, Aliakbar Ahmadzadeh, Xinyu Xing, and Gang Wang. Cade: Detecting and explaining concept drift samples for security applications. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2327–2344, 2021.

[4] Yizheng Chen, Zhoujie Ding, and David A. Wagner. Continuous learning for android malware detection. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 1127–1144, 2023.

[5] Wenhao Li, Xiao-Yu Zhang, Huaifeng Bao, Haichao Shi, and Qiang Wang. Prograph: Robust network traffic identification with graph propagation. *IEEE/ACM Transactions on Networking*, 31(3):1385–1399, 2023.

[6] Ying Zhong, Wenqi Chen, Zhiliang Wang, Yifan Chen, Kai Wang, Yahui Li, Xia Yin, Xingang Shi, Jiahai Yang, and Keqin Li. Helad: A novel network anomaly detection model based on heterogeneous ensemble learning. *Computer Networks*, 169:107049, 2020.

[7] Dongqi Han, Zhiliang Wang, Wenqi Chen, Kai Wang, Rui Yu, Su Wang, Han Zhang, Zhihua Wang, Minghui Jin, Jiahai Yang, Xingang Shi, and Xia Yin. Anomaly detection in the open world: Normality shift detection, explanation, and adaptation. *Proceedings 2023 Network and Distributed System Security Symposium*, 2023.

[8] Roberto Jordaney, Kumar Sharad, Santanu Kumar Dash, Zhi Wang, Davide Papini, Ilia Nouretdinov, and Lorenzo Cavallaro. Transcend: Detecting concept drift in malware classification models. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 625–642, 2017.

[9] Federico Barbero, Feargus Pendlebury, Fabio Pierazzi, and Lorenzo Cavallaro. Transcending transcend: Revisiting malware classification in the presence of concept drift. *2022 IEEE Symposium on Security and Privacy (SP)*, pages 805–823, 2020.

[10] Dinghuai Zhang, Kartik Ahuja, Yilun Xu, Yisen Wang, and Aaron C. Courville. Can subnetwork structure be the key to out-of-distribution generalization? In *International Conference on Machine Learning*, pages 12356–12367. PMLR, 2021.

[11] Geon Yeong Park, Sangmin Lee, Sang Wan Lee, and Jong-Chul Ye. Training debiased subnetworks with contrastive weight pruning. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7929–7938, 2023.

[12] Zhuo Huang, Muyang Li, Li Shen, Jun Yu, Chen Gong, Bo Han, and Tongliang Liu. Winning prize comes from losing tickets: Improve invariant learning by exploring variant parameters for out-of-distribution generalization. *International Journal of Computer Vision*, pages 1–19, 2024.

[13] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2018.

[14] Jungsuk Song, Hiroki Takakura, Yasuo Okabe, Masashi Eto, Daisuke Inoue, and Koji Nakao. Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation. *Proceedings of the first workshop on building analysis datasets and gathering experience returns for security*, pages 29–36, 2011.

[15] Marius Dragoi, Elena Burceanu, Emanuela Haller, Andrei Manolache, and Florin Brad. Anoshift: A distribution shift benchmark for unsupervised anomaly detection. *Advances in Neural Information Processing Systems*, 35:32854–32867, 2022.

[16] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 2018.

[17] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

[18] Chuanpu Fu, Qi Li, and Ke Xu. Detecting unknown encrypted malicious traffic in real time via flow interaction graph analysis. In *Proceedings 2023 Network and Distributed System Security Symposium (NDSS)*, 2023.

[19] Xiao Zhou, Yong Lin, Weizhong Zhang, and Tong Zhang. Sparse invariant risk minimization. In *International Conference on Machine Learning*, pages 27222–27244. PMLR, 2022.

[20] Elliot Creager, Jörn-Henrik Jacobsen, and Richard S. Zemel. Environment inference for invariant learning. In *International Conference on Machine Learning*, pages 2189–2200. PMLR, 2021.

[21] David Krueger, Ethan Caballero, Jörn-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Rémi Le Priol, and Aaron C. Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pages 5815–5826. PMLR, 2021.

[22] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations (ICLR 2019)*, 2019.

[23] Eric Jang, Shixiang Shane Gu, and Ben Poole. Categorical reparametrization with gumble-softmax. In *International Conference on Learning Representations (ICLR 2017)*. OpenReview. net, 2017.

[24] Ari S. Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 4932–4942, 2019.

[25] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2020.

[26] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018.

[27] Federico Barbero, Feargus Pendlebury, Fabio Pierazzi, and Lorenzo Cavallaro. Transcending transcend: Revisiting malware classification in the presence of concept drift. https://github.com/s2labres/transcendent-release.

[28] Limin Yang, Wenbo Guo, Qingying Hao, Arridhana Ciptadi, Aliakbar Ahmadzadeh, Xinyu Xing, and Gang Wang. Cade: Detecting and explaining concept drift samples for security applications. https://github.com/whyisyoung/CADE.

[29] Yizheng Chen, Zhoujie Ding, and David A. Wagner. Continuous learning for android malware detection. https://github.com/wagner-group/active-learning.

[30] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[31] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. Tesseract: Eliminating experimental bias in malware classification across space and time. In *28th USENIX security symposium (USENIX Security 19)*, pages 729–746, 2019.

[32] Makiya Nakashima, Alex Sim, Youngsoo Kim, Jong-Hoi Kim, and Jinoh Kim. Automated feature selection for anomaly detection in network traffic data. *ACM Transactions on Management Information Systems (TMIS)*, 12(3):1–28, 2021.

[33] Hojjat Aghakhani, Fabio Gritti, Francesco Mecca, Martina Lindorfer, Stefano Ortolani, Davide Balzarotti, Giovanni Vigna, and Christopher Kruegel. When malware is packin' heat; limits of machine learning classifiers based on static analysis features. In *Proceedings 2020 Network and Distributed System Security Symposium (NDSS)*, 2020.

[34] Mansour Ahmadi, Dmitry Ulyanov, Stanislav Semenov, Mikhail Trofimov, and Giorgio Giacinto. Novel feature extraction, selection and fusion for effective malware family classification. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pages 183–194, 2016.

[35] Farzana Alam, Rasha F. Kashef, and Muhammad Jaseemuddin. Enhancing the performance of network traffic classification methods using efficient feature selection models. *2021 IEEE International Systems Conference (SysCon)*, pages 1–6, 2021.

[36] B. Mounica and K. Lavanya. Feature selection with a deep learning based high-performance computing model for traffic flow analysis of twitter data. *The Journal of Supercomputing*, 78:15107–15122, 2022.

[37] Yubo Hou, Tram Truong-Huu, Zhenghua Chen, Chee-Keong Kwoh, and Sin G. Teo. Proteus: Domain adaptation for dynamic features in ai-assisted windows malware detection. In *2023 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 1322–1331, 2023.

[38] Haikuo Yin, Brandon Lou, and Peter Reiher. A method for summarizing and classifying evasive malware. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 455–470, 2023.

[39] Savino Dambra, Yufei Han, Simone Aonzo, Platon Kotzias, Antonino Vitale, Juan Caballero, Davide Balzarotti, and Leyla Bilge. Decoding the secrets of machine learning in malware classification: A deep dive into datasets, feature extraction, and model performance. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 60–74, 2023.

[40] Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Class-incremental learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[41] Dylan Chou and Meng Jiang. A survey on data-driven network intrusion detection. *ACM Computing Surveys (CSUR)*, 54(9):1–36, 2021.

[42] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: An ensemble of autoencoders for online network intrusion detection. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2018.

[43] Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, and Ali A. Ghorbani. Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In *IEEE 53rd International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2019.

[44] The University of New South Wales. The UNSW-NB15 Dataset Description. https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/, 2015. Accessed: 2025-03-12.

[45] Gabriel Macia Fernandez, Jose Camacho, Roberto Magan-Carri, Pedro Garcia-Teodoro, and Roberto Theron. UGR'16: A new dataset for the evaluation of cyclostationarity-based network IDSs. *Computers and Security*, 73:411–424, 2018.

[46] Ghulam Mohi-ud din. Nsl-kdd. https://dx.doi.org/10.21227/425a-3e55, 2018.