TATA: Benchmark NIDS Test Sets Assessment and Targeted Augmentation

Omar Anser, Jérôme François, Isabelle Chrisment, Daishi Kondo

TATA: Benchmark NIDS Test Sets Assessment and Targeted Augmentation

Omar Anser[1], Jérôme François[1,2], Isabelle Chrisment[1], and Daishi Kondo[3]

[1] Inria, Université de Lorraine, CNRS, LORIA, Nancy, France
`{firstname.lastname}@inria.fr`
[2] SnT - University of Luxembourg, Luxembourg
`{firstname.lastname}@uni.lu`
[3] Information Technology Center, The University of Tokyo, Japan
`daishi.kondo@nc.u-tokyo.ac.jp`

Abstract. Research works on Network Intrusion Detection Systems (NIDSs) using Machine Learning (ML) usually reports very high detection rate, often well above 90%. However, these results typically originate from overly simplistic NIDS datasets, where the test set, often just a subset of the overall dataset, mirrors the training set distribution, failing to rigorously assess the NIDS's robustness under more varied conditions. To address this shortcoming, we propose a method for **T**est sets **A**ssessment and **T**argeted **A**ugmentation (TATA). TATA is a model-agnostic approach that assesses and augments the quality of benchmark ML–based NIDS test sets. First, TATA encodes both training and test sets in a structured latent space via a contrastive autoencoder, defining three quality metrics (diversity, proximity, and scarcity) to identify test set gaps where the ML-based classification is harder. Next, TATA employs a reinforcement learning (RL) approach guided by these metrics, configuring a testbed that produces realistic data specifically targeting these gaps, creating a more robust test set. Using CIC-IDS2017 and CSE-CIC-IDS2018, we observe a positive correlation between higher metric values and increased detection difficulty, confirming their utility as meaningful indicators of test set robustness. With the same datasets, TATA's RL-based augmentation significantly raises detection difficulty for multiple NIDS models, revealing previously overlooked weaknesses.

Keywords: NIDS · ML · Data quality · Data augmentation.

1 Introduction

In Network Intrusion Detection System (NIDS) research, Machine Learning (ML) and Deep Learning (DL) have become foundational tools (throughout this paper, NIDS refers exclusively to systems that use ML or DL). Benchmark NIDS datasets play a crucial role in the design and comparative evaluation of these systems. For instance, LUCID [3] (a state-of-the-art DDoS detector) used ISCXIDS2012 [34], CIC-IDS2017 [33] (IDS2017), and CSE-CIC-IDS2018 [33] (IDS2018); both ADA [35] (an adaptive NIDS with minimal data requirements) and FlowTransformer [25] (a transformer-based model capturing intricate traffic patterns) utilized UNSW-NB15 [26]. FlowTransformer additionally relied on NSL-KDD and IDS2018. Although these datasets are widely adopted, they can contain errors and biases in data collection, labeling, post-processing, and even after publication [4, 21, 20].

Besides, one of the key issues with NIDS datasets is their simplicity [6, 1, 7], often illustrated by minimal heterogeneity among traffic that shares the same label identifying the different types of attacks or benign flows. Flood *et al.* [7] use Principal Component Analysis (PCA) to analyze the DoS Hulk labeled flows in IDS2017, showing that its training and test subsets exhibit near-complete overlap in feature-value distributions. They attribute this phenomenon to both IDS2017 and IDS2018 being generated using automated tools with limited attack exploration. Consequently, concerns arise about the relevance of these largely used datasets—particularly their test sets, often just subsets of the same data—for evaluating an NIDS, since their limited heterogeneity may lead to an overestimation of detection performance and fail to meaningfully reflect the system's robustness under operational conditions [36].

Existing metrics can analyze intrinsic dataset characteristics to quantify its overall quality [7, 6], while ignoring the downstream NIDS-classification task. For instance, they highlight issues such as mislabeled flows, dubious labeling assumptions, and near-duplicate attacks. Others evaluate a dataset from a classification perspective, assessing how challenging it is for a NIDS to separate the training labeled traffic to build its decision boundaries [22, 21, 4] without examining the later testing stage which relies on a test set and its quality to challenge the system. Prior studies in non-NIDS domains propose methods for assessing test set quality [17, 14, 30], then extend these efforts with augmentation strategies [31, 32] to better challenge the models. Yet most of these approaches rely on model-dependent indicators (e.g., neuron activation), which can be impractical when the NIDS model is unknown or closed-source. Moreover, this model-centric focus does not provide a comprehensible view of how much the distribution of a test set is aligned relative to the training set. To the best of our knowledge, only two studies address the problem of test set augmentation in the NIDS domain [7, 6], but their solutions are manual and thus not generalizable across NIDS datasets.

Given these challenges, we aim at answering two research questions:

RQ1. How can we **assess** the quality of test sets in challenging NIDS models, ensuring a robust evaluation of their detection capabilities?

RQ2. How to **augment** these test sets to better evaluate the robustness and real-world applicability of NIDS models?

To address these questions, we introduce a method for **T**est sets **A**ssessment and **T**argeted **A**ugmentation, **TATA**. TATA tackles RQ1 by defining comprehensive quality metrics to evaluate a test set relative to a training set, without relying on model internals. TATA trains a contrastive autoencoder [11, 16] to increase both inner and intra-label separability of the training set in the latent space, thus approximating the decision boundaries. Once trained, the autoencoder additionally projects the test data points, enabling the measuring of **di-**

versity (captures the range of test data points, reflecting whether the test set broadly spans the feature space or remains redundant), **proximity** (how test data points lie close to differently labeled training data points, indicating how borderline they are for classification), and **scarcity** (how uniformly test data points are spread across the decision boundaries, ensuring multiple boundary regions are tested). Higher values imply a more challenging test for the NIDS.

Addressing RQ2, TATA uses Reinforcement Learning (RL) [37]. The RL agent iteratively generates configurations, such as network conditions (e.g., bandwidth constraints or latency) or traffic patterns (e.g., bursty or steady flows). Using these on a testbed, real traffic is generated, unlike model-based data augmentation techniques (e.g., Generative Adversarial Networks, or GANs), which remains synthetic. To improve the quality of the test set, this generation is guided by the predefined quality metrics. Once trained, the agent can be applied to multiple NIDS test sets without retraining, a practical transferability that could pave the path to an easier generalization.

We evaluate TATA on the IDS2017 and IDS2018 datasets, showing that diversity, proximity, and scarcity effectively quantify a test set's challenging aspects. For example, we examine how changes in proximity correlate with detection performance for the three NIDS models used in this paper (Random Forest (RF), Support Vector Machine (SVM), and a Deep Neural Network (DNN)). We use these metrics to guide TATA's RL-based test set augmentation on IDS2018, applying the learned strategy to IDS2017. TATA increases the original metrics (diversity, proximity, scarcity) by approximately 437%, 190%, and 136% respectively by generating benign traffic only, which in turn reduces each model's macro-averaged F1-score (macro-F1) by nearly 30 points, exposing previously overlooked NIDS weaknesses hidden by the original test split. Beyond IDS2017 and IDS2018, we conducted a broader temporal and comparative analysis across multiple network-intrusion datasets showing that the test set quality is far from being satisfactory for research on NIDS.

2 Related Work

Research on test set quality assessment (RQ1) and augmentation (RQ2) has been prominent in DL, software engineering, and software testing fields. Table 1 categorizes the main studies, as identified in the literature, into **Neuron Coverage**, **Surprise Coverage**, and **Mutation Testing**, linking them to RQ1 and/or RQ2 while highlighting limitations. Notably, only one study, under mutation testing, addresses both RQ1 and RQ2, while others focus solely on RQ1. Additionally, two uncategorized methods [7, 6] address only RQ2. All categorized methods share a common limitation: requiring model access, a critical issue in the NIDS domain where access is often restricted, for example assuming commercial products.

Neuron coverage methods interprets higher neuron activations as broader model exploration, with [30] introducing neuron coverage (NC) as a metric that measures the proportion of neurons activated by a test set. However, these ap-

Table 1: Methods & limitations for test set assessment (RQ1) & augmentation (RQ2)

Method category	**Studies**	**RQ1**	**RQ2**	**Limitations**	
Neuron coverage	[30, 24, 23]	✓	✗	– Need for model access – Ignores training–test alignment	– High sensitivity to hyperparameters – Not adapted to non NN models
Surprise coverage	[17, 40, 18]	✓	✗	– Need for model access – Not adapted to non NN models	– Only proximity-based analysis
Mutation testing	[13, 38, 14]	✓	✗	– Need for model access – Lack of adaptability to model changes	– Ignores training–test alignment
Mutation testing	[31]	✓	✓	– Same limitations as [13, 38, 14] – Needs a new training run per dataset	– Use of a data generator that may produce unrealistic data points
N/A	[7, 6]	✗	✓	– Focus solely on increasing proximity	– Manual and unguided test set augmentation

proaches offer limited insight into how training and test sets align, require delicate hyperparameter tuning [12], and are limited to neural networks, excluding widely used NIDS models (e.g., RF). Other methods [10, 41, 5] employ similar coverage criteria for RQ2 but remain primarily fuzzing-based, focusing on stress-testing with diverse or adversarial inputs that lack semantic or real-world coherence, contrasting with our emphasis on realistic test set augmentation.

Surprise coverage methods assess how unexpected a test data point is by comparing its activation pattern (i.e., hidden-layer outputs of a neural network) with those of the training set. Various implementations (e.g., distance-based or likelihood-based) share the core idea that test data points whose the activations deviate substantially from the training set are considered surprising [17]. This category mainly focus on proximity in activation space and thus overlook other test set problems unlike our method that also incorporates scarcity and diversity considerations, leading to a more comprehensive evaluation.

Mutation testing introduces modifications to the model and its training set, producing **mutants** whose detection rate (the mutation score) indicates test set quality. A higher mutation score means a more robust test set. These solutions relies on mutation operators specific to each model, requiring new operators whenever the model changes. They also evaluate test sets in isolation, without relating them to the training set. Riccio *et al.* [31] extend prior mutation testing work [14] with test set augmentation but, in addition to mutation-testing limits, these approaches depend on human-interpretable, image-based generators [32], which are unsuited to the complex, non-visual nature of network traffic. Their approach requires re-training on each new dataset, further limiting its reusability.

Two NIDS-oriented works [7, 6] are not categorized in Table 1, yet each includes a subsection discussing test set augmentation (RQ2). Flood *et al.* [6] propose a complexity metric capturing spatial and temporal diversity, while Flood *et al.* [7] use heuristics to quantify dataset quality, both focusing on overall dataset aspects (rather than explicit test–train alignment as excepted in RQ1). They augment IDS2017 by replaying DoS-Hulk traffic on a testbed [2], randomly sampling page sizes and attacker bandwidths. Capturing traffic on a live testbed preserves protocol semantics, a strategy we also adopt, unlike fully synthetic

GAN outputs (e.g., NetShare [42]), diffusion models (e.g., NetDiffusion [15]), or simple feature-jittering [39], which can break flow coherence. However, their procedure remains manual, unguided, and purely random, as the configuration values are chosen without reference to any quality metric; consequently, the added data points may leave key coverage gaps unaddressed in the latent space.

TATA addresses the gaps identified: it evaluates test data points in relation to the training set through diversity, proximity, and scarcity, then augments the test set via RL with realistic network traffic data points to better assess the NIDS robustness. It is model-independent, requires no model access, and, in principle, the trained RL agent can be applied to new NIDS datasets without retraining.

3 Test sets Assessment and Targeted Augmentation

3.1 TATA Overview

TATA operates on a labeled NIDS training set ($\mathcal{D}_{\text{train}}$) and test set ($\mathcal{D}_{\text{test}}$), each containing network traffic data points $(\mathbf{x}, l_k)$, where $\mathbf{x}$ is a high-dimensional feature vector including information about traffic flow (e.g., the number of bytes exchanged), and $l_k \in \mathcal{L} = \{l_1, \ldots, l_n\}$ is the label (e.g., benign or a type of attack). Fig. 1 outlines TATA using an illustrative example with $\mathcal{L} = \{\text{Benign, Attack-1, Attack-2}\}$. TATA proceeds in three steps: *(i)* a preliminary

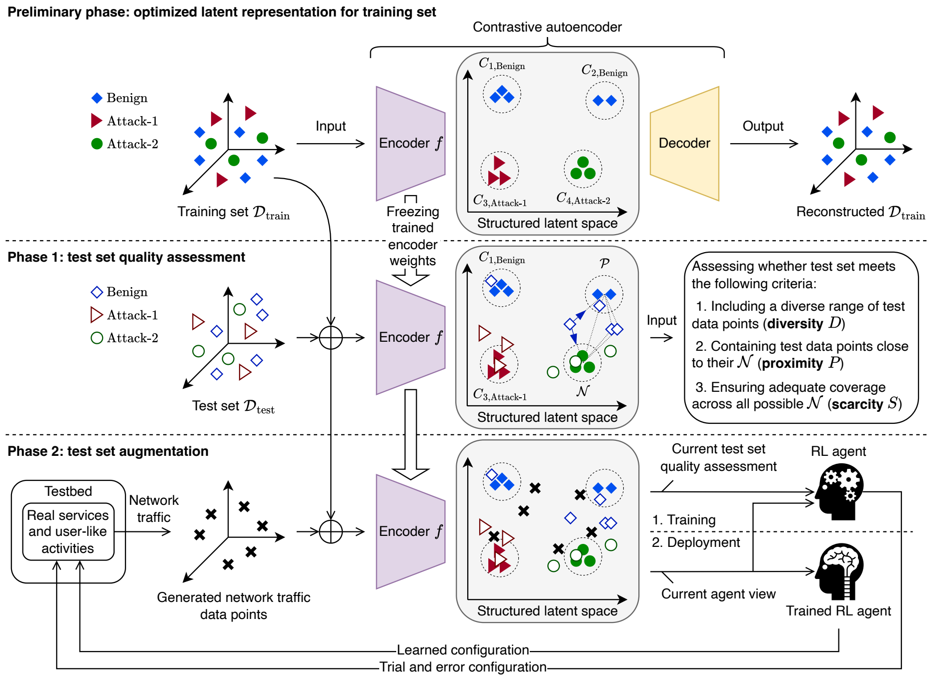
Preliminary phase: optimized latent representation for training set
Contrastive autoencoder
Benign
Attack-1
Attack-2
Training set $\mathcal{D}_{\text{train}}$
Input
Encoder f
$C_{1,\text{Benign}}$
$C_{2,\text{Benign}}$
$C_{3,\text{Attack-1}}$
$C_{4,\text{Attack-2}}$
Structured latent space
Decoder
Output
Reconstructed $\mathcal{D}_{\text{train}}$
Freezing trained encoder weights
Phase 1: test set quality assessment
Test set $\mathcal{D}_{\text{test}}$
$\mathcal{P}$
$\mathcal{N}$
Input
Assessing whether test set meets the following criteria:
1. Including a diverse range of test data points (diversity D)
2. Containing test data points close to their $\mathcal{N}$ (proximity P)
3. Ensuring adequate coverage across all possible $\mathcal{N}$ (scarcity S)
Phase 2: test set augmentation
Testbed
Real services and user-like activities
Network traffic
Generated network traffic data points
Current test set quality assessment
RL agent
1. Training
2. Deployment
Current agent view
Trained RL agent
Learned configuration
Trial and error configuration

Fig. 1: TATA, a method to assess and augment the quality of a test set

phase that constructs a structured latent space from $\mathcal{D}_{\text{train}}$, *(ii)* phase 1 that assesses $\mathcal{D}_{\text{test}}$'s quality, and *(iii)* phase 2 that augments $\mathcal{D}_{\text{test}}$. $\mathcal{L}$ is the set of labels that the NIDS must classify.

The preliminary phase provides the latent space foundation by training a **contrastive autoencoder** with encoder f, parameterized by $\boldsymbol{\theta}$, on $\mathcal{D}_{\text{train}}$. During training, the contrastive objective [11, 16] shapes the latent space into well-separated clusters, each containing encoded training data points with similar labels and closely aligned input feature values, while preserving reconstruction accuracy. This process yields a trained f that encodes each $\mathbf{x} \in \mathcal{D}_{\text{train}}$ into a latent representation $\mathbf{z} = f(\mathbf{x}; \boldsymbol{\theta})$, forming $\mathcal{Z}_{\text{train}} = \{\mathbf{z} \mid \mathbf{z} = f(\mathbf{x}; \boldsymbol{\theta}),\ \mathbf{x} \in \mathcal{D}_{\text{train}}\}$. We determine the number of clusters in $\mathcal{Z}_{\text{train}}$ using the silhouette score, then apply k-means to assign each $\mathbf{z} \in \mathcal{Z}_{\text{train}}$ to a cluster C_{i,l_k} (where i is the cluster index and l_k is the majority label among its members). All clusters form the set $\mathcal{C}$ knowing that a single label may correspond to multiple clusters (e.g., $C_{1,\text{Benign}}$ and $C_{2,\text{Benign}}$ in Fig. 1), which reflects intra-class variability in $\mathcal{D}_{\text{train}}$. By reducing the data's dimensionality, the contrastive autoencoder not only speeds up subsequent computations but also approximates the decision boundaries inherent in $\mathcal{D}_{\text{train}}$, all without being tied to a specific NIDS model.

In phase 1, we use the trained f to encode $\mathcal{D}_{\text{test}}$ into the same structured latent space as $\mathcal{D}_{\text{train}}$. We obtain $\mathcal{Z}_{\text{test}} = \{\mathbf{z} \mid \mathbf{z} = f(\mathbf{x}; \boldsymbol{\theta}),\ \mathbf{x} \in \mathcal{D}_{\text{test}}\}$, which positions each test data point relative to $\mathcal{D}_{\text{train}}$'s distribution. For each $\mathbf{z} \in \mathcal{Z}_{\text{test}}$ with input label l_k, we compute its Euclidean distance $\|\mathbf{z} - \boldsymbol{\mu}_{C_{i,l_j}}\|$ to every cluster centroid $\boldsymbol{\mu}_{C_{i,l_j}}$ in $\mathcal{C}$. Among these distances, the cluster C_{i,l_j} whose label l_j matches l_k and minimizes $\|\mathbf{z} - \boldsymbol{\mu}_{C_{i,l_j}}\|$ is termed as its positive cluster $\mathcal{P}(\mathbf{z})$. Conversely, the cluster C_{i,l_j} whose label differs ($l_j \neq l_k$) and minimizes that distance is its negative cluster $\mathcal{N}(\mathbf{z})$. Fig. 1 illustrates this pairing by arrows linking a Benign encoded test data point to its positive and negative clusters, representing the intrinsic maximal difficulty to make the right decision. Accordingly, we define three complementary metrics (diversity, proximity, and scarcity) to assess how effectively $\mathcal{D}_{\text{test}}$ challenges the NIDS in its detection task. They are computed in the structured latent space on a **per-cluster basis** (details in Section 3.2). We then aggregate each metric's results across all clusters $C_{i,l_k} \in \mathcal{C}$ to measure how effectively $\mathcal{D}_{\text{test}}$ challenges each subgroup of $\mathcal{D}_{\text{train}}$.

In phase 2 (detailed in Section 3.3), the RL agent is trained by iteratively interacting, through trial and error, with a configurable testbed that it directs to generate network traffic (see Fig. 1). At each iteration, the agent first consults its current view of the structured latent space and then selects a traffic-generation configuration. The resulting network traffic is merged into the evolving $\mathcal{D}_{\text{test}}$ and, after encoding, is re-assessed with TATA's predefined metrics, yielding a quality measure that serves as the reward. This reward guides the agent's configuration choices, which are refined over multiple iterations. Once training, the agent's learned policy can be used on the same or other NIDS datasets without requiring further retraining (i.e., without additional reward signals).

3.2 Test Set Quality Assessment

Because TATA's metrics are computed on a per-cluster basis, we begin by defining the subset of encoded test data points tied to each cluster C_{i,l_k}. Specifically, for each C_{i,l_k}, we focus on $\mathcal{Z}_{\text{test}}^{(C_{i,l_k})} = \{\mathbf{z} \in \mathcal{Z}_{\text{test}} \mid \mathcal{P}(\mathbf{z}) = C_{i,l_k}\}$, the $\mathcal{Z}_{\text{test}}$ subset of **encoded test data points whose positive cluster is** C_{i,l_k}. To support metrics computation, we also introduce the function $\text{PairNeg}(\mathcal{Z}_{\text{test}}, C_{i,l_k}) = \{(\mathbf{z}, \mathcal{N}(\mathbf{z})) \mid \mathbf{z} \in \mathcal{Z}_{\text{test}}^{(C_{i,l_k})}\}$ that pairs every $\mathbf{z} \in \mathcal{Z}_{\text{test}}^{(C_{i,l_k})}$ with its negative cluster $\mathcal{N}(\mathbf{z})$. Each pair $(\mathbf{z}, \mathcal{N}(\mathbf{z}))$ indicates the nearest differently labeled cluster that could challenge the classification of $\mathbf{z}$ away from its positive cluster C_{i,l_k} (*i.e.* leading to a classification error so).

Diversity A common testing requirement is to cover a wide range of test cases. Diversity (D) captures the variability among the data points in $\mathcal{Z}_{\text{test}}^{(C_{i,l_k})}$. A higher D reflects a $\mathcal{D}_{\text{test}}$ in which individual data points exhibit minimal redundancy and more fully cover the available feature space.

D is calculated using the Vendi Score [8] (denoted as V_{i,l_k}), which quantifies how evenly the data points span the feature space by computing the von Neumann entropy of their normalized similarity matrix. Since V_{i,l_k} ranges from 1 (minimal diversity) to the size of $\mathcal{Z}_{\text{test}}^{(C_{i,l_k})}$ (maximal diversity), we min–max normalize it to $[0, 1]$. Finally, we average these normalized scores across all clusters to obtain :

$$D = \frac{1}{|\mathcal{C}|} \sum_{C_{i,l_k} \in \mathcal{C}} V_{i,l_k} \tag{1}$$

Proximity Because diverse data points lying far from decision boundaries are typically simpler to classify, proximity (P) measures how close data points in $\mathcal{Z}_{\text{test}}^{(C_{i,l_k})}$ are to their negative clusters. A higher P suggests that $\mathcal{D}_{\text{test}}$ includes data points near differently labeled training data point, which are more likely to challenge the NIDS.

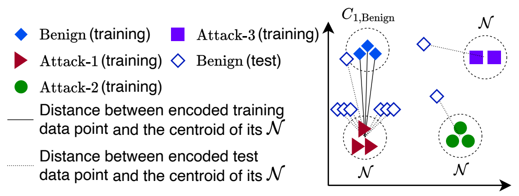
Benign (training)
Attack-3 (training)
Attack-1 (training)
Benign (test)
Attack-2 (training)
Distance between encoded training data point and the centroid of its $\mathcal{N}$
Distance between encoded test data point and the centroid of its $\mathcal{N}$
$C_{1,\text{Benign}}$
$\mathcal{N}$
$\mathcal{N}$
$\mathcal{N}$

Fig. 2: Training vs. Test Distances for $C_{1,\text{Benign}}$

Formally, for each cluster C_{i,l_k}, we define:

$$d_{\text{test}_i} = \left\{ \|\mathbf{z} - \boldsymbol{\mu}_{\mathcal{N}(\mathbf{z})}\| \mid (\mathbf{z}, \mathcal{N}(\mathbf{z})) \in \text{PairNeg}\big(\mathcal{Z}_{\text{test}}, C_{i,l_k}\big) \right\} \tag{2}$$

as the set of Euclidean distances from each $\mathbf{z} \in \mathcal{Z}_{\text{test}}^{(C_{i,l_k})}$ to the centroid of its negative cluster. To determine whether these encoded test data points lie farther from C_{i,l_k} than their training counterparts $\mathbf{z} \in \mathcal{Z}_{\text{train}}$ assigned to C_{i,l_k} (and thus pose a greater challenge to the NIDS), we analogously define d_{train_i}. Fig. 2 illustrates d_{test} (dashed) and d_{train} (solid) derived for the set $\mathcal{Z}_{\text{test}}^{(C_{1,\text{Benign}})}$. Using the empirical cumulative distribution functions F_{test_i} and F_{train_i} of d_{test_i} and d_{train_i}, we derive the maximum positive difference using the one-sided Kolmogorov–Smirnov statistic $\text{KS}_{C_{i,l_k}} \in [0, 1]$. A high value means that the distances in d_{test_i}

are significantly smaller than those in d_{train_i}, so the encoded test data points are very close to their negative clusters. To capture the worst-case, we define P as the maximum KS value computed across all $\mathcal{Z}_{\text{test}}^{(C_{i,l_k})}$ (for $C_{i,l_k} \in \mathcal{C}$):

$$P \;=\; \max_{C_{i,l_k} \in \mathcal{C}} \Big(\mathrm{KS}_{\,C_{i,l_k}} \Big). \tag{3}$$

Scarcity Beyond diversity and proximity, a robust test approach must ensure that test data points appear across multiple decision boundaries where misclassifications may arise. Scarcity (S) evaluates how uniformly data points in $\mathcal{Z}_{\text{test}}^{(C_{i,l_k})}$ are spread across **all possible negative clusters**, Fig. 3 depicts an optimal scenario where each of the three possible negative clusters of $C_{1,\text{Benign}}$ is assigned an equal number of encoded test data points (three points each). In contrast, Fig. 4 shows a non-optimal scenario and reflects thus a poor scarcity.

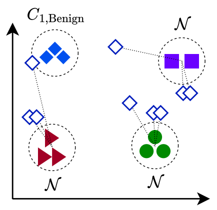

Fig. 3: Optimal scarcity scenario

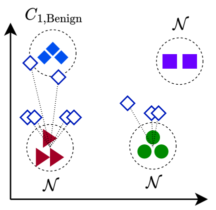

Fig. 4: Non-optimal scarcity scenario

Formally, we define the set of all possible negative clusters of C_{i,l_k} as:

$$\mathcal{N}_{\text{all}}(C_{i,l_k}) = \{\, C_{j,l_p} \in \mathcal{C} \mid j \neq i,\; l_p \neq l_k \,\} \tag{4}$$

For each $C_{j,l_p} \in \mathcal{N}_{\text{all}}\big(C_{i,l_k}\big)$, we define the number of associated data points as:

$$\mathrm{PointCount}(C_{j,l_p}) = \big|\, \{\, \mathbf{z} \mid (\mathbf{z}, \mathcal{N}(\mathbf{z})) \in \mathrm{PairNeg}(\mathcal{Z}_{\text{test}}, C_{i,l_k}) \;\wedge\; \mathcal{N}(\mathbf{z}) = C_{j,l_p} \} \big| \tag{5}$$

Normalizing $\mathrm{PointCount}(C_{j,l_p})$ by $|\mathrm{PairNeg}(\mathcal{Z}_{\text{test}}, C_{i,l_k})|$ yields the distribution R_{i,l_k}. The complement of the Gini Coefficient measures its uniformity:

$$R_{i,l_k} = \left\{ \frac{\mathrm{PointCount}(C_{j,l_p})}{\big|\mathrm{PairNeg}\big(\mathcal{Z}_{\text{test}},\, C_{i,l_k}\big)\big|}, \forall\, C_{j,l_p} \in \mathcal{N}_{\text{all}}\big(C_{i,l_k}\big) \right\}, G_{i,l_k} = 1 - \mathrm{Gini}\big(R_{i,l_k}\big) \tag{6}$$

Values of G_{i,l_k} range from 0 to 1, where higher values indicate a more even distribution among negative clusters. S is the average of G_{i,l_k} for all $C_{i,l_k} \in \mathcal{C}$:

$$S = \frac{1}{|\mathcal{C}|} \sum_{C_{i,l_k} \in \mathcal{C}} G_{i,l_k}. \tag{7}$$

3.3 Test Set Targeted Augmentation

We formulate the augmentation of $\mathcal{D}_{\text{test}}$ as a Partially Observable Markov Decision Process (POMDP) and train a RL agent whose policy π—a learnable rule that maps the current view of the latent space (the observation), to the next testbed configuration (the action)—interacts with this POMDP to maximise the three test set quality objectives: diversity, proximity, and scarcity. The main components of this POMDP are:

- **Observations Ω.** Each observation $o \in \Omega$ is a compressed summary of the structured latent space, comprising the cluster centroids $\boldsymbol{\mu}_{C_{i,l_k}}$ (for all $C_{i,l_k} \in \mathcal{C}$) and basic statistics (mean, min, max, variance, std) computed over $\mathcal{Z}_{\text{test}}$.
- **Actions $\mathcal{A}$.** Each action $a \in \mathcal{A}$ specifies a traffic-generation configuration (e.g., bandwidth throttling, latency injection, or packet corruption) that the testbed uses to generate one new network traffic data point.
- **Reward R.** After each action, the agent receives a scalar reward r quantifying the improvement in test set quality in terms of diversity D, proximity P, and scarcity S. Formally, $R = \frac{1}{\frac{w_D}{D} + \frac{w_P}{P} + \frac{w_S}{S}}$, where w_D, w_P, and w_S are tunable hyperparameters.

During training, the agent first observes the cluster centroids and the summary of the initial $\mathcal{Z}_{\text{test}}$, denoted by o_0, while its policy π is randomly initialized. We implement this policy using Deep Reinforcement Learning (DRL), where a neural network approximates π. At each step t, the agent selects a traffic-generation configuration a_t (i.e., how the testbed should create one network traffic data point). In response, the configured testbed instantiates real network traffic under those conditions, yielding a new data point. It is then encoded, $f(\mathbf{x})$, and appended to $\mathcal{Z}_{\text{test}}$. The updated $\mathcal{Z}_{\text{test}}$ is summarized to yield, with the cluster centroids, the new partial observation o_{t+1}, and the test set is evaluated using the three quality metrics to compute the reward r_t. Upon receiving the transition tuple $\big(o_t, a_t, r_t, o_{t+1}\big)$, the agent updates π with the aim of increasing future rewards. This process repeats for multiple steps, with each step adding one new encoded data point to $\mathcal{Z}_{\text{test}}$. If a target test set quality is reached or a maximum step budget is met, the current episode terminates and the environment resets to the original test set (or a fresh copy), after which a new episode begins. Over multiple episodes, π converges to a better policy π^*, learning a strategy for configuring the testbed to generate network traffic flows that progressively enhance the diversity, proximity, and scarcity of the test set.

Once the training phase is complete, π^* can be reused during testing to augment either the same $\mathcal{D}_{\text{test}}$ or another NIDS dataset. In the latter case, the preliminary phase and phase 1 must first be carried out to establish a compatible structured latent space for that new NIDS dataset; the pre-trained agent can then directly apply π^* to guide the generation of additional network traffic.

4 Experiments and Results

4.1 Experimental Setup

Experiments were conducted on a server running Ubuntu 22.04.3 LTS with an Intel(R) Xeon(R) Gold 6258R CPU @ 2.70GHz processor, 500 GB of RAM, and an NVIDIA RTX A6000. Our implementation is provided.[4]

[4] https://gitlab.inria.fr/oanser/tata

Datasets We used refined versions of the IDS2017 and IDS2018 datasets [21], recognized benchmarks for NIDS [7] yet noted for their relatively low input complexity [6] and classification complexity [22]. Both are flow-based using CICFlowMeter [21] to create bidirectional flows enriched with statistical features. While both cover the same number of attacks, they differ in scale: about 926k benign and 253k attack flows for IDS2017 and about 59.8M benign and 4.1M attack flows for IDS2018. Benign traffic is mostly DNS, HTTP, and HTTPS flows with other minority protocols (e.g., SSH, FTP, SMTP). The attack types range from infiltration and port scanning to DoS, DDoS, and web exploits. Due to the very low occurrence (10 to 20 flows) of certain attack types in IDS2017, a pattern not observed in IDS2018, we removed these attack types.

NIDS Models We show TATA's model-agnostic nature using three multi-class ML models: RF, SVM, and a DNN which features three fully connected ReLU layers (128 neurons each) and a softmax output. These models are widely used in NIDS research, showing strong results on IDS2017 and IDS2018 [28]. Each dataset is stratified 60/20/20 into training ($\mathcal{D}_{\text{train}}$), validation ($\mathcal{D}_{\text{val}}$), and test ($\mathcal{D}_{\text{test}}$) splits; $\mathcal{D}_{\text{val}}$ is used to select model hyper-parameters via random search.

Contrastive Autoencoder The contrastive autoencoder is a multilayer perceptron selected after a grid search over candidate architectures and training hyper-parameters. The best configuration, ranked according to the Silhouette-guided k-means accuracy, uses three fully connected layers with 64, 32, and 16 neurons, and ReLU activations. It is trained for 250 epochs with a batch size of 128 and a learning rate of 0.001. The contrastive loss employs a margin of 10 and a regularisation factor of $\lambda = 0.1$, while the latent space is three-dimensional.

4.2 Preliminary Phase: Evaluating the Contrastive Autoencoder

We evaluate the output from the contrastive autoencoder (the cluster set $\mathcal{C}$, derived from $\mathcal{Z}_{\text{train}}$) comparing it to a vanilla autoencoder (which omits the contrastive loss). Both autoencoders are trained on ten distinct 60%-$\mathcal{D}_{\text{train}}$ splits of IDS2017 and IDS2018, each generated with a different random seed, and assessed via silhouette score (a measure of cluster separability) and K-Means (KM) accuracy (alignment of clusters with true labels).

The contrastive autoencoder achieves significantly higher silhouette scores ($0.96_{\pm 0.00}$ vs. $0.73_{\pm 0.05}$ on IDS2017 and $0.98_{\pm 0.00}$ vs. $0.76_{\pm 0.02}$ on IDS2018), indicating that the clusters in $\mathcal{Z}_{\text{train}}$ are both well-grouped and clearly divided. These strong internal structures yield near-perfect KM Accuracy (over 99% on both datasets), indicating that each cluster is almost entirely composed of encoded data points from the same input label. In contrast, the vanilla autoencoder's lower KM Accuracy (around 60%) reveals considerable label mixing.

Takeaway: The contrastive autoencoder yields a highly discriminative and well-structured latent space, outperforming a vanilla autoencoder.

4.3 Phase 1: Test Set Assessment Metrics in Practice

Having introduced each metric's rationale, we now address RQ1 by examining whether their values vary in alignment with the performance of the implemented NIDSs. We use a single 60/20 $\mathcal{D}_{\text{train}}$–$\mathcal{D}_{\text{test}}$ stratified split on the IDS2017 dataset (IDS2018 yields similar observations). Because this split alone gives each metric a single value, we selectively manipulate $\mathcal{D}_{\text{test}}$ to produce varying test set qualities, as detailed in the subsections below. We measure correlation using Pearson's r for linear relationships and Spearman's ρ for rank-based (monotonic) trends, so both absolute differences and relative orderings are captured.

Diversity. We investigate whether including additional traffic types (labels) in the NIDS evaluation (an inherently more challenging scenario) correlates with higher diversity. To do this, we generate all sub-test sets from IDS2017's $\mathcal{D}_{\text{test}}$ corresponding to each label combination of size k (k ranges from 1 to 9 since IDS2017 includes 9 types of traffic). We then compute and plot the resulting diversity values in Fig. 5, observing a perfect correlation (Pearson's $r = 1$, Spearman's $\rho = 1$) between the number of labels and mean diversity. As k grows from 1 to 9, diversity rises monotonically from near-minimal to a maximum of 0.1209.

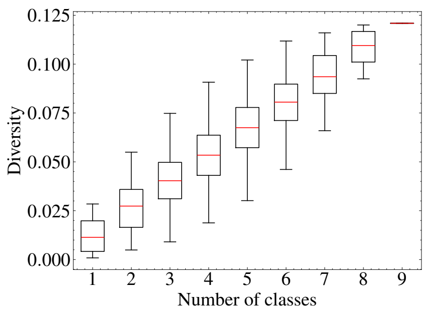

Fig. 5: Box plots of the diversity metric for sub-test sets of size $k = 1, \ldots, 9$

Takeaway: The diversity metric captures the increased evaluation challenge that results from adding more label types, which increases the range of traffic types the NIDS must handle.

Proximity. To assess proximity, we compute each encoded test data point's distance to its negative cluster. We rank these distances from largest to smallest and split the ordered list into 100 cumulative sub-test sets: the first contains the top 1 %, the second the top 2 % (including the first), and so on. As we move from sets with the largest distances to those with smaller ones, the average distance decreases, thereby increasing the proximity metric. Using each sub-test set, we evaluate whether higher proximity levels challenge the NIDS, as indicated by increased error.

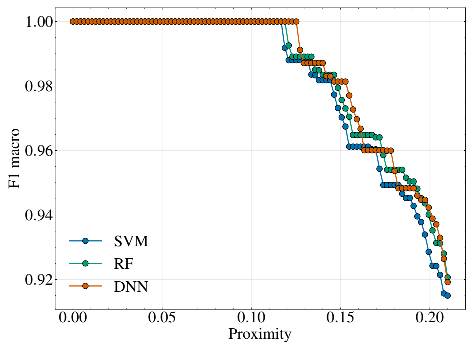

Fig. 6: Effect of Proximity on NIDS Performance

In Fig. 6, each circular marker represents a sub-test set, with its average proximity (x-axis) plotted against the macro-F1 of the three NIDS (RF, DNN, SVM). All three NIDS maintain macro-F1=1.0 across more than half the sub-test sets (proximity ≤ 0.11–0.14), but macro-F1 gradually declines (to about 0.92 for RF/DNN and 0.91 for SVM) as more borderline data points appear. We observe a strong negative Spearman correlation ($\rho = -0.90 \pm 0.01$) and a similarly negative Pearson correlation ($r = -0.86 \pm 0.01$), both averaged across the three NIDS, confirming that higher proximity corresponds to lower macro-F1. Their nearly identical trajectories (pairwise correlations Pearson > 0.98, Spearman > 0.97) indicate that proximity is agnostic to the ML model.

Takeaway: Proximity captures the escalating difficulty posed by borderline test data points, as higher proximity values coincide with lower NIDS performance.

Scarcity. As with proximity, we investigate whether scarcity correlates with $\mathcal{D}_{\text{test}}$'s difficulty and thus affects NIDS performance. Defined in Sec. 3.2, scarcity measures how uniformly encoded test data points distribute across all possible negative clusters. To produce sub-test sets with different scarcity levels, we run each NIDS on $\mathcal{D}_{\text{test}}$, record misclassified data points, and map them to their latent representations. We then systematically redistribute these misclassified instances among the available negative clusters in 10 steps, transitioning from a less-uniform (clustered) to a more-uniform (dispersed) arrangement. As with proximity, we therefore create sub-test sets with an increasing scarcity and evaluate how this impacts the NIDS performance.

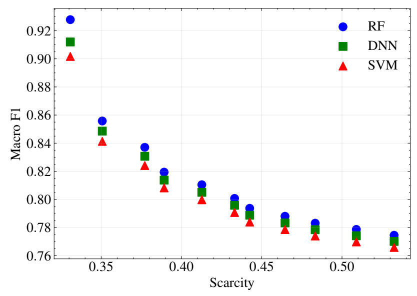

Fig. 7: Effect of Scarcity on NIDS Performance

Because our objective is to create a wide range of types of misclassification errors (*i.e.* mixing different labels), macro-F1 is also considered, correcting imbalance effect of a particular label. In overall, a strong negative Spearman correlation ($\rho = -1 \pm 0$, averaged across models) and a strong negative Pearson correlation ($r = -0.8885 \pm 0.0055$) clearly indicate that higher scarcity corresponds to a lower macro-F1. In details, Figure 7 shows that, after the first redistribution step, macro-F1 shows a sharp decline. Performance then continues to decrease more gradually and monotonically; for instance, the macro-F1 of the RF-based NIDS falls below 0.78 when the scarcity level reaches 0.53.

Takeaway: A more uniform spread of misdetection data points across decision boundaries increases scarcity and decreases the macro-averaged F1-score, confirming scarcity's relevance as a difficulty indicator.

4.4 Phase 2: Test set Augmentation Evaluation

In this second phase, TATA focuses on augmenting $\mathcal{D}_{\text{test}}$ with Benign flows only, as a proof of concept. The RL agent relies on a two-host testbed (client and server) exchanging Benign SSH traffic (a protocol present in IDS2017 and IDS2018). While RL ideally operates at millisecond timescales, allowing the collection of numerous transition tuples $\big(o_t, a_t, r_t, o_{t+1}\big)$ for effective learning, our full pipeline (traffic generation, flow conversion, encoding with f and metrics computation) requires seconds, making a conventional online loop infeasible. Thus, we employ offline RL, training on a static dataset of transitions (o_t, a_t, r_t, o_{t+1}) derived from IDS2018 while reserving IDS2017 for final testing. We omit the reverse scenario because IDS2018's larger cluster set $\mathcal{C}$ exceeds what an agent trained on IDS2017 can handle as observation input.

To construct these transition tuples, we first split IDS2018 into $\mathcal{D}_{\text{train}}$ and an initial $\mathcal{D}_{\text{test}}$ and augment the latter iteratively with a random traffic-generation configuration (action) on the SSH traffic, drawn from uniform parameter ranges (forming the action set a_t): loss [5%, 10%], jitter [4 ms, 10 ms], delay [10 ms, 40 ms], duplication [0.1%, 5%], corruption [0.1%, 10%], reordering [0.1%, 50%], and correlation [50%, 100%]. The action is applied with the Linux `tc` command on the client side while running a predefined SSH scenario creating large random files, performing frequent file operations, executing complex commands, modifying file permissions, and cleaning up. Next, we convert the resulting network traffic into flow(s) using CICFlowMeter, after which we apply our pipeline (metric computation and reward calculation with weights $w_P = w_S = w_D = 1$). If 5,000 steps have elapsed or the reward exceeds 0.9, we reset the initial $\mathcal{D}_{\text{test}}$ to begin a new episode. Across these episodes, we collect roughly 500,000 transitions without any filtering, deliberately retaining failed episodes to ensure the offline RL algorithms (CQL [19] and TD3+BC [9], chosen for continuous-action support and for penalizing out-of-dataset actions to remain within known transitions, and tuned via grid search) encounter a diverse range of outcomes.

After training, we evaluate each RL agent for 20 episodes on IDS2017, using different random seeds to define an 60–20 train–test split each time. During these evaluations, the agent manipulates traffic configurations on the same SSH testbed, but no further policy updates or reward signals occur. Newly generated flows are labeled benign and added to the IDS2017 test set. Finally, we measure how this augmentation affects (i) our test-set quality metrics and (ii) NIDS detection performance (macro-averaged precision and recall). We report mean and standard deviation over the 20 episodes. Three baselines are considered:

– **Random Agent:** The agent selects traffic-shaping parameters (loss, jitter, delay, etc.) at random, using no learned strategy.
– **GAN-Based Method (Netshare[42]):** Netshare employs a GAN to synthesize network traffic. Although it attempts to mimic real-world distributions, its generated flows may not fully capture the complexity of realistic network behavior.
– **Augmentation guided with NC [30]:** This variant uses the TATA's offline RL pipeline but replaces the default reward with a neuron-coverage metric from the DNN-based NIDS. During training, after each newly generated traffic flow, we measure the fraction of activated neurons (e.g., 70% activation yields a reward of 0.7).

We compare only methods with publicly available, working code that can be adapted to NIDS; others, such as [31], focus on image-based augmentation and are therefore omitted.

Fig. 8 summarizes our results, plotting each augmentation method alongside the test set (no augmentation) on the x-axis.

Our TATA approach focuses on CQL-based findings, as TD3+BC produces similar outcomes that do not change our conclusions. The y-axis shows both quality metrics (diversity, proximity, and scarcity) and NIDS performance metrics (macro-averaged precision and recall), all normalized to [0,1].

Both the Random agent and Netshare dramatically reduce diversity, driving it close to zero. However, the impact on proximity is negligible (the Random agent lowers it slightly, while Netshare raises it slightly) whereas scarcity declines for both. Together, these changes have nearly no effect on NIDS performance. In contrast, TATA increases proximity by about 190%, scarcity by roughly 136%, and diversity by around 437%, whereas the augmentation guided with NC focuses on proximity, increasing it by approximately 129%. This limitation arises because neuron coverage targets data points near negative clusters, those most stressful to the NIDS, while overlooking diversity and scarcity. These differences also influence precision and recall. For example, for the RF-based NIDS, TATA lowers precision by roughly 46 %, compared with about 23 %

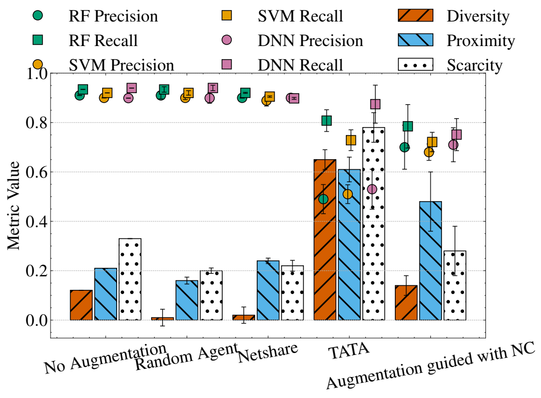

Fig. 8: Impact of TATA and Baselines on Test-Set Quality Metrics and Detection Performance

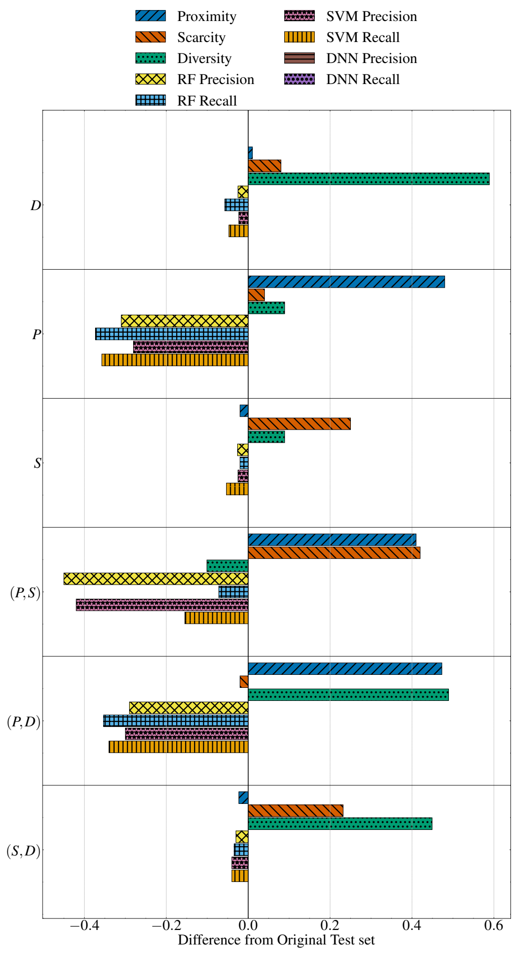

Fig. 9: Impact of Partial Metric Combinations on Test-Set Quality and NIDS Performance

for NC, while recall drops by around 13 % with TATA versus 16 % with NC. We see that scarcity, by distributing misclassified points more widely, strongly affects precision (tied to false positives).
Takeaway: TATA comprehensively stresses the NIDS through diversity, proximity, and scarcity, creating a more challenging test set (thus lower precision/recall) than augmentation guided with NC, which focuses solely on proximity.

To have a higher understanding of the importance of the quality metrics, the reward is now calculated from a single metrics or any combination of two of them resulting in 6 different cases reported in Fig. 9 where D, P, and S stands for diversity, proximity, and scarcity respectively. Horizontal bars, centered at zero, illustrate how much each configuration improves or degrades performance relative to the original set. Combinations that include proximity $(P), (P, S), (P, D)$ offer the greatest challenge to the NIDSs, while the others have minimal impact. This is because proximity targets the decision boundary of negative clusters, where the NIDS struggles most with detection. For example, with the RF-based NIDS, in the (P, S) combination, recall declines by about 8% (from 0.934 to 0.862) and precision by about 49% (0.910 to 0.460), whereas in (S, D), they fall by roughly 4% (0.934 to 0.900) and 3% (0.910 to 0.880), respectively.
Takeaway: Proximity emerges as the critical factor in stressing the NIDSs, whereas scarcity and diversity offer complementary effects by covering all possible decision boundaries and avoiding redundant test cases, together expanding test coverage.

4.5 Datasets Analysis

We applied TATA's metrics to various networking-related datasets, including intrusion detection and other traffic-classification benchmarks, to assess their test sets over time.

Table 2 highlights studied benchmarks, from early, small-scale examples like NSL-KDD (2009) to more recent, encrypted-oriented sets (ISCX Tor, VPN-NonVPN) and IoT-focused ones (Bot-IoT, ToN-IoT). They have evolved toward more realistic testbeds, extensive logging, and multi-vector attacks.

Using ten 60/20 training–test splits, Figure 10 plots our metrics. As shown, proximity remains fairly stable (mostly 0.2–0.3), though early

Table 2: Datasets Summary

Dataset	**Key Highlights**	**#Attacks**	**#Feat**
NSL-KDD (2009)	DARPA'98 refinement; ∼148k records	22	41
CTU-13 (2011)	13 botnet scenarios; >15M flows	13	15
ISCX IDS 2012	24-host lab; ∼2.5M flows	4	14
UNSW-NB15 (2015)	Cyber-Range lab; 2.54 M flows	9	49
CIC-UNSW (2015)	Augmented UNSW-NB15; ∼450 k flows	9	∼80
ISCX Tor (2016)	5-user Tor traffic; 2 scenarios	2	28
VPN-NonVPN (2016)	Multi-app data; VPN vs non-VPN	14	84
CIC-IDS 2017	18-host testbed; 3.12M flows	7	83
CSE-CIC-IDS 2018	Enterprise-scale net; 16.23 M flows	7	∼80
Bot-IoT (2018)	Simulated IoT; >72 M records	4	46
CIC-DDoS 2019	Lab network; multi-vector DDoS	13	80
ToN-IoT (2020)	IoT/IIoT testbed; ∼0.48 M flows	9	42

versions of IDS2017 and IDS2018 (with labeling errors) reach around 0.5. Diversity shows a broader range, from near zero to about 0.5, without a clear monotonic trend. Scarcity largely tracks proximity until CIC-IDS2017 (errors present), then aligns more with diversity and fluctuates more. Across all ten splits, each metric exhibits minimal, often negligible, standard deviation.

These findings suggest that, despite increasingly complex testbed designs over time (including more attacks, applications, and intricate topologies), the core quality of these datasets' test sets remains consistent, and their inherent difficulty has not substantially increased. In contrast, widely used image classification benchmarks, MNIST and CIFAR-10, exhibit higher complexity. Notably, we adapted our contrastive autoencoder with convolutional layers for image data.

Takeaway: Even with recent advances in dataset design, reference datasets differ widely in difficulty across application domains, and our metrics reveal that most NIDS test sets still pose only a moderate challenge, highlighting the network-security community's ongoing shortage of truly demanding, well-suited datasets.

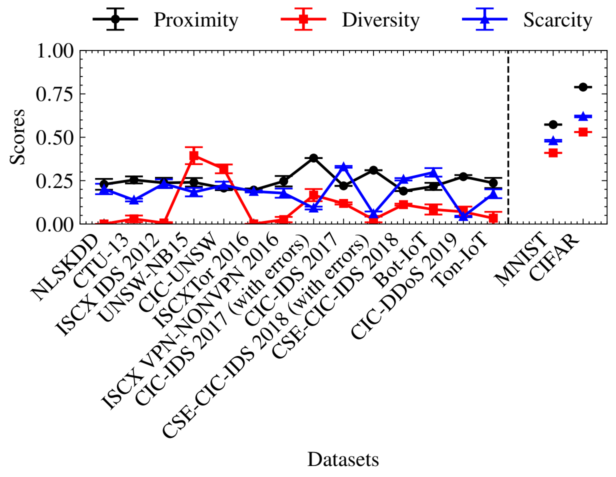

Fig. 10: Test Sets Assessment of NIDS Datasets

5 Conclusion

In this paper, we presented TATA, a model-agnostic method to assess and augment the quality of benchmark NIDS test sets. TATA uses a contrastive autoencoder to build a structured latent space from a training set, derives three comprehensive metrics (diversity, proximity, scarcity) from the test set, and combines them with RL to guide realistic traffic augmentation. Our evaluation shows that the contrastive autoencoder effectively organizes the latent space, these dataset-centric metrics capture test set difficulty, and the RL-based agent increases the challenge for various ML-based NIDS models.

TATA's pipeline still exhibits some limitations. *(i) Computational overhead.* Assessing a new test set currently entails retraining the contrastive autoencoder and running an exhaustive hyperparameter search, which dominates the total runtime. We aim to reduce this cost by starting the preliminary phase from a pre-trained autoencoder and fine-tuning only lightweight adapters. *(ii) Limited traffic diversity.* So far, the augmentation phase produces only benign SSH flows. We plan to enrich the testbed with additional benign protocols and with attack traffic by adopting a multi-agent design in which each agent generates a specific traffic type. *(iii) Fixed-cluster assumption.* The present RL agent is built for a fixed number of clusters, preventing its use on datasets that contain more clusters than expected. *(iv) Unverified RL generalization.* Evaluation has been restricted to the IDS2017 benchmark. The agent's ability to generalize to datasets with markedly different feature distributions (such as UNSW-NB15 and Bot-IoT) remains untested. Assessing performance across a broader suite of NIDS benchmarks is therefore a central goal of future work.

Acknowledgement. The authors acknowledge partial support from the French National Research Agency through the France 2030 initiative (project *Superviz*, ANR-22-PECY-0008) and additional funding in part from the Luxembourg National Research Fund (FNR, grant C23/IS/18088425/COCTEL). The opinions expressed are solely those of the authors and do not necessarily reflect the views of the French government.

A Computational Complexity of Test Set Assessment

To assess a given test set $\mathcal{D}_{\text{test}}$, TATA follows a workflow that begins with contrastive autoencoder training and concludes with computing the cluster–wise metrics D, P, and S.
(i) Preliminary phase (contrastive autoencoder training). The contrastive autoencoder is trained on the corresponding training set $\mathcal{D}_{\text{train}}$. A hyper-parameter tuning examine H candidate configurations; for each one, the autoencoder is trained for E complete epochs over $\mathcal{D}_{\text{train}}$. A single forward–backward pass scales with the number of parameters in the model, and the worst case is obtained with the largest candidate network, whose size is $|\theta_{\max}|$. Hence, the overall cost of the preliminary phase is: $\boxed{O\big(H\,E\,|\mathcal{D}_{\text{train}}|\,|\theta_{\max}|\big)}$.
(ii) Phase 1-a (cluster identification). On the embeddings from the best run, silhouette-guided k-means performs I refinement rounds, yielding a cost of: $\boxed{O\big(I\,|\mathcal{D}_{\text{train}}|\,|\mathcal{C}|\big)}$.

(iii) Phase 1-b (centroid assignment). Every embedding $\mathbf{z} \in \mathcal{Z}_{\text{train}} \cup \mathcal{Z}_{\text{test}}$ is then compared with the $|\mathcal{C}|$ centroids to locate its $\mathcal{P}(\mathbf{z})$ and $\mathcal{N}(\mathbf{z})$ clusters, for a total of $\boxed{O\big((|\mathcal{D}_{\text{train}}| + |\mathcal{D}_{\text{test}}|)\,|\mathcal{C}|\big)}$ distance evaluations.

(iv) Phase 1-c (metric computation). A single linear pass over $\mathcal{Z}_{\text{test}}$ updates the per-cluster counters needed for D, P, and S, costing: $\boxed{O\big(|\mathcal{D}_{\text{test}}| + |\mathcal{C}|\big)}$.

Combining the four steps, the overall time complexity becomes:

$$\boxed{O\big(H\,E\,|\mathcal{D}_{\text{train}}|\,|\theta_{\max}| + I\,|\mathcal{D}_{\text{train}}|\,|\mathcal{C}| + (|\mathcal{D}_{\text{train}}| + |\mathcal{D}_{\text{test}}|)|\mathcal{C}|\big)}$$

Pre-training the encoder once on an heterogeneous set of network traffic and then fine-tuning a small adapter on $m \ll |\mathcal{D}_{\text{train}}|$ data points for $\tilde{E} \ll E$ epochs cuts the training component from $H\,E\,|\mathcal{D}_{\text{train}}|\,|\theta_{\max}|$ to $\tilde{E}\,m\,|\theta_\star|$, yielding an overall per-dataset complexity of:

$$\boxed{O\big(\tilde{E}\,m\,|\theta_\star| + I\,|\mathcal{D}_{\text{train}}|\,|\mathcal{C}| + (|\mathcal{D}_{\text{train}}| + |\mathcal{D}_{\text{test}}|)\,|\mathcal{C}|\big), \quad \text{where } |\theta_\star| \ll |\theta_{\max}|}$$

B Offline RL Hyperparameter Tuning

We adopt the Split-Select-Retrain pipeline proposed by Nie *et al.* [27]. Concretely, we create $K = 20$ independent 50/50 train/validation splits of the offline dataset. For each algorithm–hyperparameter configuration (CQL or TD3+BC variants), we train a candidate policy on every training split. We then estimate each policy's expected return on the corresponding validation split using Fitted Q Evaluation (FQE), a simple method that learns a reward-predicting value function from the logged data and reuses it to score a policy without any new environment interaction, originally introduced by Paine *et al.* [29]. Averaging the K FQE estimates yields a robust performance metric for each configuration; we choose the configuration with the highest mean score and retrain the agent on the full offline dataset before deployment.

References

1. Catillo, M., Pecchia, A., Villano, U.: Machine learning on public intrusion datasets: Academic hype or concrete advances in nids? In: 2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S) (2023)
2. Clausen, H., Flood, R., Aspinall, D.: Traffic generation using containerization for machine learning. In: Proceedings of the 2019 Workshop on DYnamic and Novel Advances in Machine Learning and Intelligent Cyber Security (2022)
3. Doriguzzi-Corin, R., Millar, S., Scott-Hayward, S., Martínez-del Rincón, J., Siracusa, D.: Lucid: A practical, lightweight deep learning solution for ddos attack detection. IEEE Transactions on Network and Service Management (2020)
4. Engelen, G., Rimmer, V., Joosen, W.: Troubleshooting an intrusion detection dataset: the cicids2017 case study. In: 2021 IEEE Security and Privacy Workshops (SPW) (2021)
5. Feng, Y., Shi, Q., Gao, X., Wan, J., Fang, C., Chen, Z.: Deepgini: prioritizing massive tests to enhance the robustness of deep neural networks. In: Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. ISSTA 2020 (2020)
6. Flood, R., Aspinall, D.: Measuring the complexity of benchmark nids datasets via spectral analysis. In: 2024 IEEE European Symposium on Security and Privacy Workshops (EuroSPW) (2024)
7. Flood, R., Engelen, G., Aspinall, D., Desmet, L.: Bad design smells in benchmark nids datasets. In: 2024 IEEE 9th European Symposium on Security and Privacy (EuroSP) (2024)
8. Friedman, D., Dieng, A.B.: The vendi score: A diversity evaluation metric for machine learning (2023)
9. Fujimoto, S., Gu, S.S.: A minimalist approach to offline reinforcement learning. In: Advances in Neural Information Processing Systems (2021)
10. Guo, J., Jiang, Y., Zhao, Y., Chen, Q., Sun, J.: Dlfuzz: Differential fuzzing testing of deep learning systems. In: Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (2018)

11. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06) (2006)
12. Harel-Canada, F., Wang, L., Gulzar, M.A., Gu, Q., Kim, M.: Is neuron coverage a meaningful measure for testing deep neural networks? In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE 2020 (2020)
13. Hu, Q., Ma, L., Xie, X., Yu, B., Liu, Y., Zhao, J.: Deepmutation++: A mutation testing framework for deep learning systems. In: 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE) (2019)
14. Humbatova, N., Jahangirova, G., Tonella, P.: Deepcrime: mutation testing of deep learning systems based on real faults. In: Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis. ISSTA 2021 (2021)
15. Jiang, X., Liu, S., Gember-Jacobson, A., Bhagoji, A.N., Schmitt, P., Bronzino, F., Feamster, N.: Netdiffusion: Network data augmentation through protocol-constrained traffic generation. Proc. ACM Meas. Anal. Comput. Syst. (2024)
16. Khosla, P., et al.: Supervised contrastive learning. Advances in neural information processing systems
17. Kim, J., Feldt, R., Yoo, S.: Guiding deep learning system testing using surprise adequacy. In: Proceedings of the 41st International Conference on Software Engineering. ICSE '19 (2019)
18. Kim, J., Feldt, R., Yoo, S.: Evaluating surprise adequacy for deep learning system testing. ACM Trans. Softw. Eng. Methodol. (2023)
19. Kumar, A., Zhou, A., Tucker, G., Levine, S.: Conservative q-learning for offline reinforcement learning. In: Advances in Neural Information Processing Systems (2020)
20. Lanvin, M., Gimenez, P.F., Han, Y., Majorczyk, F., Mé, L., Totel, E.: Errors innbsp;thenbsp;cicids2017 dataset andnbsp;thenbsp;significant differences innbsp;detection performances it makes. In: Risks and Security of Internet and Systems: 17th International Conference, CRiSIS 2022, Sousse, Tunisia, December 7-9, 2022, Revised Selected Papers (2023)
21. Liu, L., Engelen, G., Lynar, T.M., Essam, D.L., Joosen, W.: Error prevalence in nids datasets: A case study on cic-ids-2017 and cse-cic-ids-2018. 2022 IEEE Conference on Communications and Network Security (CNS) (2022)
22. Lorena, A.C., Garcia, L.P.F., Lehmann, J., Souto, M.C.P., Ho, T.K.: How complex is your classification problem? a survey on measuring classification complexity. ACM Comput. Surv. (2019)
23. Ma, L., Juefei-Xu, F., Xue, M., Li, B., Li, L., Liu, Y., Zhao, J.: Deepct: Tomographic combinatorial testing for deep learning systems. In: 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER) (2019)
24. Ma, L., Juefei-Xu, F., Zhang, F., Sun, J., Xue, M., Li, B., Chen, C., Su, T., Li, L., Liu, Y., Zhao, J., Wang, Y.: Deepgauge: multi-granularity testing criteria for deep learning systems. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. ASE '18 (2018)
25. Manocchio, L.D., Layeghy, S., Lo, W.W., Kulatilleke, G.K., Sarhan, M., Portmann, M.: Flowtransformer: A transformer framework for flow-based network intrusion detection systems. Expert Systems with Applications (2024)
26. Moustafa, N., Slay, J.: Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS)

27. Nie, A., Flet-Berliac, Y., Jordan, D.R., Steenbergen, W., Brunskill, E.: Data-efficient pipeline for offline reinforcement learning with limited data, neurIPS 2022
28. Owezarski, P.: Investigating adversarial attacks against random forest-based network attack detection systems. In: NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium (2023)
29. Paine, T.L., et al.: Hyperparameter selection for offline reinforcement learning. arXiv preprint (2020)
30. Pei, K., Cao, Y., Yang, J., Jana, S.: Deepxplore: Automated whitebox testing of deep learning systems. In: Proceedings of the 26th Symposium on Operating Systems Principles. pp. 1–18. ACM (2017)
31. Riccio, V., Humbatova, N., Jahangirova, G., Tonella, P.: Deepmetis: augmenting a deep learning test set to increase its mutation score. In: Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering. ASE '21 (2022)
32. Riccio, V., Tonella, P.: When and why test generators for deep learning produce invalid inputs: an empirical study. 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (2022)
33. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy, (ICISSP) (2018)
34. Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A.A.: Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Computers Security (2012)
35. Singla, A., Bertino, E., Verma, D.: Preparing network intrusion detection deep learning models with minimal data using adversarial domain adaptation. In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security. ASIA CCS '20 (2020)
36. Sudyana, D., Verkerken, M., D'Hooge, L., Lin, Y.D., Hwang, R.H., Lai, Y.C., Yudha, F., Wauters, T., Volckaert, B., De Turck, F.: Quality analysis in ids dataset: Impact on model generalization. In: 2024 IEEE Conference on Communications and Network Security (CNS) (2024)
37. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction (2018)
38. Tufano, M., Kimko, J., Wang, S., Watson, C., Bavota, G., Di Penta, M., Poshyvanyk, D.: Deepmutation: A neural mutation tool. In: 2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) (2020)
39. Wang, C., Finamore, A., Michiardi, P., Gallo, M., Rossi, D.: Data augmentation for traffic classification. In: Richter, P., Bajpai, V., Carisimo, E. (eds.) Passive and Active Measurement (2024)
40. Weiss, M., Chakraborty, R., Tonella, P.: A review and refinement of surprise adequacy. In: 2021 IEEE/ACM Third International Workshop on Deep Learning for Testing and Testing for Deep Learning (DeepTest) (2021)
41. Xie, X., Ma, L., Juefei-Xu, F., Xue, M., Chen, H., Liu, Y., Zhao, J., Li, B., Yin, J., See, S.: Deephunter: A coverage-guided fuzz testing framework for deep neural networks. In: Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (2019)
42. Yin, Y., Lin, Z., Jin, M., Fanti, G., Sekar, V.: Practical gan-based synthetic ip header trace generation using netshare. In: Proceedings of the ACM SIGCOMM 2022 Conference. SIGCOMM '22 (2022)