# SECL: A Zero-Day Attack Detector and Classifier based on Contrastive Learning and Strong Regularization

Robin Duraz, David Espes, Julien Francq, Sandrine Vaton

# SECL: A Zero-Day Attack Detector and Classifier based on Contrastive Learning and Strong Regularization

Robin Duraz
robin.duraz@ecole-navale.fr
Chaire of Naval Cyberdefense, Lab-STICC
Brest, France

David Espes
david.espes@univ-brest.fr
UBO, Lab-STICC
Brest, France

Julien Francq
julien.francq@naval-group.com
Naval Group (Naval Cyber Laboratory, NCL)
Ollioules, France

Sandrine Vaton
sandrine.vaton@imt-atlantique.fr
IMT Atlantique, Lab-STICC
Brest, France

## ABSTRACT

Intrusion Detection Systems (IDSs) always had difficulties in detecting Zero-Day attacks (ZDAs). One of the advantages of Machine Learning (ML)-based IDSs, which is their superiority in detecting ZDAs, remains largely unexplored, especially when considering multiple ZDAs. This is mainly due to the fact that ML-based IDSs are mainly using supervised ML methods. Although they exhibit better performance in detecting known attacks, they are by design unable to detect unknown attacks because they are limited to detecting the labels present in the dataset they were trained on. This paper introduces SECL, a method that combines Contrastive Learning and a new regularization method composed of dropout, Von Neumann Entropy (VNE) and Sepmix (a regularization inspired from mixup). SECL is close to, or even better than supervised ML methods in detecting known attacks, while gaining the ability to detect and differentiate multiple ZDAs. Experiments were performed on three datasets, UNSW-NB15, CIC-IDS2017 and WADI, effectively showing that this method is able to detect multiple ZDAs while achieving performance similar to supervised methods on known attacks. Notably, the proposed method even has an overall better performance than a supervised method knowing all attacks on the WADI dataset. These results pave the way for better detection of ZDAs, without reduction of performance on known attacks.

## CCS CONCEPTS

• **Security and privacy → Intrusion detection systems**; • **Human-centered computing → HCI design and evaluation methods**; • **Computing methodologies → Machine learning**.

## KEYWORDS

Intrusion Detection Systems; Zero-Day Detection; Zero-Day Classification; Contrastive Learning; Semi-Supervised Learning; Open-World


*ARES 2024, July 30-August 2, 2024, Vienna, Austria*

ACM ISBN 979-8-4007-1718-5/24/07…$15.00
https://doi.org/10.1145/3664476.3664505

**ACM Reference Format:**
Robin Duraz, David Espes, Julien Francq, and Sandrine Vaton. 2024. SECL: A Zero-Day Attack Detector and Classifier based on Contrastive Learning and Strong Regularization. In *The 19th International Conference on Availability, Reliability and Security (ARES 2024), July 30-August 2, 2024, Vienna, Austria.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3664476.3664505

## 1 INTRODUCTION

Standard approaches in Intrusion Detection based on signatures have always been quite efficient in detecting properly identified patterns. However, signature-based approaches are by design limited and struggle to detect cyberattacks when there is a (small) change in behavior with regard to the reference patterns, and even less needs to be said about Zero-Day Attacks (ZDAs). Machine Learning (ML) approaches, however, are by design able to detect cyberattacks with small changes in behavior, and possibly also ZDAs.

Traditionally, ML approaches to detect cyberattacks rely on either supervised methods or anomaly-based methods. Supervised methods learn with labeled datasets and exhibit a high performance on known attacks but are unable to detect ZDAs. Anomaly-based methods are able to detect attacks, including ZDAs, but are unable to distinguish them. More recent research has focused on Open-Set Learning (OSL) methods. They are generally based on supervised methods that constrain the problem to a multi-class problem for known classes, and allow to reject samples that are outliers to all known classes. Rejected samples are then all identified as belonging to a big anomaly class.

However, current attack methodologies are generally complex and composed of multiple steps, e.g., reconnaissance, lateral movement, privilege escalation, data exfiltration, impact[1], etc. Therefore, a human expert that will handle alerts raised by an IDS needs attacks, both known and ZDAs, to be properly distinguished. Being able to distinguish between different steps of ZDAs allows to much more quickly decide on the actions to take to respond to a raised alert, and thus should not be overlooked.

When using ML-based IDSs to detect multiple ZDAs in a real-world scenario, there are two possibilities:

- **Scenario 1**: They are completely new. In this case, the traffic never existed and the attacks can only be detected during testing.

[1] https://attack.mitre.org/

- **Scenario 2**: They have been present for some time, and have remained unidentified. In this case, the traffic exists, is unlabeled but can still be used to train ML-based IDSs.

Both of these scenarios are very distinct and, ideally, both should be considered from the perspective of training ML-based IDSs. In scenario 2, an IDS that was already trained using a labeled dataset would be able to be updated with unlabeled data to better differentiate ZDAs. This is similar to Incremental Learning (IL), where new classes are incrementally learned, but extends IL to use unlabeled data.

Multiple ML methods and datasets [10, 13] exist to develop and evaluate ML-based IDSs. Besides supervised methods, ML methods can be unsupervised, semi-supervised or self-supervised. Both unsupervised and self-supervised methods do not require any labeled data while semi-supervised methods can leverage both labeled and unlabeled data. Self-supervised methods are a subcategory of unsupervised methods that train in a supervised way with self-created labels. Unsupervised methods, other than anomaly-based, are generally able to distinguish between more than two classes and can group together "similar" instances. When labeled data is available, however, their performance is generally lacking compared to supervised methods. Ideally, the chosen approach should be able to both leverage labeled data to reach a performance similar to supervised methods on known classes, and be able to detect and distinguish ZDAs.

Contrastive Learning (CL) has gained popularity in recent years by decreasing the difference in performance between supervised and unsupervised Deep Learning (DL) on image applications [15]. CL is based on Neural Networks (NNs) that will learn a new representation of the data, to be used in downstream tasks. It typically relies on self-supervised learning, which is based on augmentations. Augmentations are transformations of the data that retain the semantic information, e.g., color shifts in images still represent the same object. These augmentations are used to create samples that share the same semantic information and are called positives. Other samples or augmentations of other samples are considered as negatives samples. The goal is then to learn a representation space where anchors are close to their positives and far away from their negatives. However, augmentations are much more difficult to define in the case of IDS datasets where modifications of some information, e.g., ports and protocols, can be too complex or even counterproductive because they do not retain the semantic information.

Consequently, there are two main difficulties to use CL to train IDS. First, as stated previously, augmentations are impossible to properly define. Secondly, CL, as any ML method for IDSs, also suffers from the high imbalance in the datasets, and tends to overfit on more prevalent classes, e.g., typically normal traffic.

The proposed approach, SECL (for Sepmix rEgularized Contrastive Learning) solves both of these difficulties. It will combine CL with a supervised contrastive loss [17] to remove the need to define proper augmentations. Additionally, it will use a new regularization method that combines dropout, Sepmix (for Separation through Mixup), and VNE. All components of this regularization method impact different parts of the approach and are all required. Dropout prevents co-adaption of neurons inside all layers of NNs and reduces overfitting. VNE forces SECL to learn richer representations by penalizing when eigenvalues of the NN's last layer are not well distributed. Finally, Sepmix creates virtual samples between classes to force SECL to reduce intra-class differences and obtain more compact representations for known classes. SECL will be tested on three well known IDS/Industrial Control Systems datasets: UNSW-NB15, CIC-IDS2017 and WADI. To the best knowledge of the authors, this is the first paper that considers the ability to detect and classify both known and multiple new unknown attacks and shows performance similar to supervised methods while having a relatively high performance (in both detection and classification) on new unknown attacks.

The rest of the paper is organized as follows: Section 2 presents related works. Section 3 describes the proposed approach, while Section 4 presents the experimental setup. Section 5 presents and analyzes the results. Finally, Section 6 concludes the paper and discusses future avenues of research.

## 2 RELATED WORK

### 2.1 Open-World vs. Open-Set Learning

The problem of trying to detect and classify new unknown classes is a very difficult problem that has been first formalized in [3], and is named Open-World Learning (OWL). It extends Open-Set Learning (OSL) [27] that considered all unknowns as a single anomaly class by considering multiple unknown classes. It also extends it by applying IL to incrementally add in a supervised manner the multiple new classes that were detected. In [4], it is shown that although small steps are taken in the direction of OWL, mainly with advances in OSL, this is not sufficient and much remains to be done. Since then, recent works on image applications [5, 30] have shown that most recent ML methods are slowly gaining the ability to detect and distinguish unknown classes.

Much of the work in Intrusion Detection and detection of new classes has focused on OSL. In [7, 14, 19, 26], it is shown that using OSL methods can lead to detection of unknown classes with a rate ranging between 20% and sometimes up to 90%. However, these approaches only consider a single anomaly class, and sometimes are tested on relatively small datasets. Furthermore, they are always tested by leaving out one attack of the dataset, which restricts the distribution of unknown attacks to that of a single class. As such, this is unclear if and how these methods would scale when considering multiple different new classes, even if considering them as a single anomaly class.

### 2.2 Contrastive Learning for ZDAs

CL appears to be a promising solution to detect ZDAs, but this is a relatively new ML topic for cybersecurity, and research using CL in Intrusion Detection is relatively scarce. In [34], it is shown that using a contrastive loss alongside a more common cross-entropy loss can achieve state-of-the-art results in Intrusion Detection. Unfortunately, the detection of new classes is not addressed. In [23], CL is used to perform Intrusion Detection and new class detection, where it shows similar performance ranges to [7, 14, 19, 26]. Unfortunately, it also remains in the OSL setting.

An important component of CL that requires careful consideration is how anchors, positives and negatives are chosen for training.

This generally relies on methods that will refine the selection of positives and negatives with regard to their respective anchor, to select cases that will better contribute to learning. The supervised contrastive loss in [17] allows to use multiple positives and multiple negatives per anchor, and requires to have at least one of each. Current approaches generally try to select hard negatives (the closest negative) [16] or hard negatives and easy positives (the closest positive) [33]. A method based on [33] will be used in this paper.

Finally, since CL is a self-supervised method, it is possible to leverage labeled data and perform semi-supervised learning. While semi-supervised learning can be framed as in [11] to reduce the amount of labeled samples, it is also possible to consider semi-supervised learning when there are classes that are present in the data but are unlabeled. As such, the previously mentioned scenario 2 can be considered as an incremental semi-supervised learning problem where new classes without labels are added to improve detection of both known classes and ZDAs.

## 2.3 Regularization methods

In order to be able to generalize and detect new unknown classes, it is also important to prevent the model from overfitting. Regularization methods in Deep Learning (DL) are numerous [32], and are generally effective to regularize for in-distribution (known classes) samples. However, it is still unclear if they will be sufficient to enable detection of unknown classes.

Multiple regularization methods are used with CL for applications on images [21], such as Cutout, mixup [35], CutMix, AugMix. Excluding mixup, these methods are using operations specific to images and are not applicable to Intrusion Detection data. However, mixup has been shown to work in multiple application domains, including tabular data [9, 20].

Additional methods such as VNE [18] regularization have shown multiple benefits such as better generalization, learning representations of better quality or preventing representation collapse in self-supervised learning. In the proposed approach, VNE will be used, as well as Sepmix, a regularization inspired from mixup.

## 2.4 Datasets

In order to train and evaluate ML-based IDSs, labeled datasets are required. Furthermore, it is important for these datasets to closely resemble realistic traffic with sufficiently diverse cyberattacks. KDD'99 [1] and NSL-KDD [31] are the two most used datasets [13]. However, these datasets, and particularly the former, are heavily criticized because of their age and various other problems such as redundancy [8].

The UNSW-NB15 [25] and CIC-IDS2017 [28] datasets are more recent and based on quite complete environments. Both being more recent datasets, it also ensures that the environment and simulated traffic are more representative of nowadays' real-world traffic.

It is also possible to use ICS (Industrial Control System) datasets where cyberattacks are often more specific and normal traffic less diverse. Because ICSs are now often connected through networks or even the internet, security of ICSs is currently much more important than it was a decade ago and it can be beneficial to include them when testing ML-based IDSs. One such dataset is the WADI dataset [2], which represents a water treatment plant.

Details of the three datasets used in this paper, UNSW-NB15, CIC-IDS2017 and WADI, are available in Table 1.

# 3 PROPOSED APPROACH

The proposed approach consists of a CL algorithm to learn better representations that will be used by a combination of a K-Means and a K-Nearest Neighbors algorithm to cluster and assign a label to these representations. The difficulty of the task in scenario 1 comes from the fact that an unknown number of classes do not exist in the training dataset, and they only appear during testing, as shown in Figure 1. The goal is thus to, along with correctly classifying known classes, detect all unknown classes and be able to classify them. In scenario 2, this unknown number of classes is present in the training dataset but is unlabeled.

During training, the proposed approach uses the supervised contrastive loss to circumvent the need for data augmentation, a memory bank to compensate for the high imbalance in the data, Sepmix (a regularization method inspired from mixup) and VNE. The contrastive model is composed of a Contrastive Encoder (CE)

Table 1: Datasets details

| Dataset | Number of instances per class | Total |
|---|---|---|
| **UNSW-NB15** | Normal: 2218761, Generic: 215481, Exploits: 44525, Fuzzers: 24246, DoS: 16353, Reconnaissance: 13987, Analysis: 2677, Backdoor: 2329, Shellcode: 1511, Worms: 174 | 2540047 |
| **CIC-IDS2017** | Benign: 2273097, DoS Hulk: 231073, Portscan: 158930, DDoS: 128027, DoS GoldenEye: 10293, FTP-Patator: 7938, SSH-Patator: 5897, DoS Slowloris: 5796, DoS Slowhttptest: 5499, Botnet: 1966, Web Attack Brute Force: 1507, Web Attack XSS: 652, Infiltration: 36, Web Attack SQL Injection: 21, Heartbleed: 11 | 2830743 |
| **WADI** | Normal: 947347, Attack_3−4: 1742, Attack_10: 1620, Attack_1: 1502, Attack_5: 852, Attack_6: 808, Attack_9: 700, Attack_8: 672, Attack_7: 632, Attack_2: 592, Attack_13: 578, Attack_14: 204, Attack_15: 89 | 957338 |

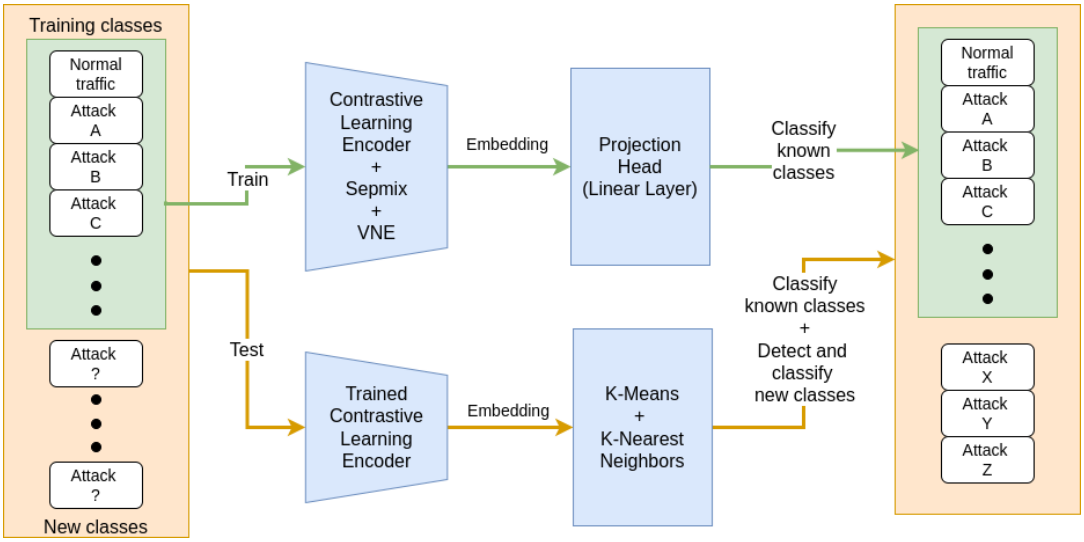


**Figure 1: Proposed approach with CL, Sepmix and VNE. The number of new classes is unknown during training.**

and a projection head. During testing, this CE will output representations that will be used by a combination of a K-Means and K-Nearest Neighbors algorithms to detect and classify known and unknown classes. An algorithm describing a single training step of a batch (a group) of samples is shown in Algorithm 1, with all three hyperparameters' ($\lambda$, $\alpha_{mem}$ and $\alpha_{VNE}$) values being found using grid search. All components of the approach and their use in Algorithm 1 will be more detailed in Section 3.1, and Section 3.2.

SECL, the approach proposed in this paper, will be tested on both scenario 1 and 2.

## 3.1 Contrastive Learning

The performance of CL is directly related to how data is augmented to create positives. Manual methods would require finding values or ranges of values where semantic information is retained for each feature. Furthermore, this would also differ for different datasets, and need to be changed every time data changes or the IDS is retrained, which makes such approach unusable in practice. On the other hand, the supervised contrastive loss as introduced in [17] allows to define positives and negatives without introducing bias and will adapt as data changes. With regard to an anchor, positives are samples of the same class while negatives are samples of other classes.

This supervised contrastive loss $\mathcal{L}_{supcon}$ is shown in Eq. 1, where $i \in I \equiv \{1...N\}$ is the index of a sample in a batch of size $N$ and represents the chosen anchor, $y_i$ is the label of sample $i$. With $\mathcal{F}$ representing the CE and $g$ being the projection head, $h_i = \mathcal{F}(x_i)$ are the representations, with $H = \{h_1, h_2, ..., h_N\}$ the representation matrix, and $z_i = g(h_i)$ the outputs of the model. $A(i) \equiv I \setminus \{i\}$, and $P(i) \equiv \{p \in A(i) : y_p = y_i\}$ represents the set of positives (samples of the same class). $z_p$ thus represent positives with regard to $z_i$ and $z_a$ are all outputs excluding $z_i$. Finally, $\tau$ is a temperature hyperparameter. The goal of this loss is to penalize when negative samples are closer to an anchor than positive samples.

**Algorithm 1:** A training step (for a single batch)

**Data:** X, y the data and labels from a batch,
$\mathcal{F}$ the Contrastive Encoder,
g the projection head,
$\lambda$ the hyperparameter for Sepmix,
$\alpha_{mem}$ the hyperparameter for the memory bank,
$\alpha_{VNE}$ the hyperparameter for VNE

```
/* Compute the representation matrix                    */
H ← F(X)
/* If memory is full, it is updated following
   the method described in Section 3.1                  */
memory ← save to memory bank(H, α_mem)
/* Mixed representations are computed using the
   memory and batch representations and using
   the λ hyperparameter, as described in Eq. 4          */
mixed_H, mixed_y ← get mixed samples(H, memory, λ)
/* Get positives and negatives used for loss
   computation                                          */
positives, negatives ←get positives and negatives(H,
  y, mixed_H, mixed_y)
/* The loss is computed using the projection of
   representations, positives and negatives of
   each sample, as in Eq. 5                             */
loss ← L_supcon(g(H), g(positives), g(negatives))
/* VNE is computed as in Eq. 2 added to the loss,
   as in Eq. 6                                          */
L_SECL ← loss − α L_VNE(g(H))
/* Update F and g                                       */
backward(L_SECL)
```

$$\mathcal{L}_{supcon} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} log \frac{exp(z_i \cdot z_p / \tau)}{\sum_{a \in A(i)} exp(z_i \cdot z_a / \tau)} \quad (1)$$

While this loss presents some desirable properties, e.g., generalization to an arbitrary number of positives, higher gradients for hard positives and negatives, it also gains some of the undesirable properties of supervised methods, such as the tendency to overfit.

As any contrastive loss, this loss relies on the definition of positives and negatives. Anchors will be selected from the training batch while the sets of positives and negatives will be selected from the samples created through Sepmix (Sepmix will be detailed in Section 3.2). In this way, the positives will generally be further away from the anchor, while negatives will often be closer to the anchor. This allows to further reduce intra-class difference and increase inter-class difference, which is necessary to properly differentiate new attacks from normal traffic.

To train DL methods, data goes through NNs as batches of multiple samples. CL approaches are very dependent on the size of batches, this influences the number of positives and negatives that can be used for computation. Batches are generally very big to increase diversity of samples for each class within a single batch. Unfortunately, the proportion of attack traffic is relatively low in Intrusion Detection datasets, which leads to batches often having a single sample or none at all of a specific attack. In these cases, the attack does not contribute to the loss and is much more difficult to learn because there are no positives within the same batch. Increasing the size of batches can quickly become impossible because of resources constraints. Therefore, a solution is to use a memory bank of samples that retains part of the samples of each class, as in [12, 24]. However, using memory in this way is not trivial, because the memory saves representations (outputs of the CE). Therefore, memory needs to be updated frequently to reflect the changes in the CE brought by the learning process. Unfortunately, updating memory also cannot be too frequent, because this, along with other regularization methods, can make training more unstable. While advanced methods such as described in [12] are possible, this requires training of an additional encoder and is both slower and much more complex to train. Such an approach would possibly interfere with other regularization methods and make training much more unstable. Therefore, it has been chosen to use a simpler method for a more stable training process: update the memory with probability 0.1 for each class, once memory is full. Memory is filled until a sufficient number $N$ of samples of each class is in the memory bank. Experiments performed led to $N = 20$ to ensure stability of the training process.

During training, a projection head (a simple linear layer) is used to project the representations into a layer that has a neuron per known class, similarly to common DL algorithms. It is conjectured in [6] that using a contrastive loss induces a loss of information. As such, using a projection head allows for the information to be lost mainly in the projection head, thus creating richer representations before projection.

During testing, this projection head is removed and replaced by a combination of a K-Means and a K-Nearest Neighbors algorithm to detect and classify an unknown number of classes. While a K-Means algorithm alone is able to detect and classify by assigning labels with the computed clusters, the addition of K-Nearest Neighbors allows to assign labels by determining the labels of samples closest to cluster centers. K-Means will cluster similar representations together, and K-Nearest Neighbors will be used to assign a label to each cluster via a majority vote.

## 3.2 Regularization methods

While the proposed approach without regularization reaches performance similar to supervised methods on known classes, it remains largely unable to detect and possibly classify unknown attacks, hence the need for regularization. One of the main reasons is that normal traffic being much more prevalent and its distribution often overlapping with the distribution of other classes, any ML approach tends to overfit on known attacks and defaults to identifying anything unknown as normal traffic. This will be mitigated using Sepmix, as described later in this section. Furthermore, learned representations tend to be of insufficient quality to properly differentiate unknown classes from known classes. Both dropout and VNE will be used to increase the quality of learned representation by making the use of neurons inside the NN more balanced.

A commonly used regularization method to train NNs is the mechanism of dropout [29]. This randomly zeroes-out different neurons in NNs at each step of the training process which reduces co-adaptation of neurons, i.e., the fact that neurons are activated by the same information. For CL, this essentially reduces the risk of representation collapse as well as improves the quality of all intermediate representations. In the proposed approach, as is often used in the literature, a dropout of 0.2 has shown the best performance.

VNE has been used in [18] and shows a high effectiveness in enforcing the use of all neurons in a NN, especially in the last layer, thus improving generalization. This effectively forces the CE to learn representations of higher quality that will be able to better differentiate unknown from known classes. Otherwise, unknown classes sometimes collapse into a single known class: normal traffic. In order to compute VNE, there are two steps involved. First, the autocorrelation $C_{auto}$ of the representation matrix $Z = \{z_1, z_2, ..., z_N\}$, with $z_i$ being the representation of $x_i$ through CE and the projection head, is computed. Then, with $\lambda_i$ being the $i$-th eigenvalue of $C_{auto}$, the VNE loss is computed as in Eq. 2.

$$\begin{aligned} C_{auto} &= Z^T Z / N \\ \mathcal{L}_{VNE} &= -\sum_j \lambda_j \, log \, \lambda_j \end{aligned} \quad (2)$$

To reduce overfitting of CL approaches, Sepmix, that is inspired from mixup is used. Mixup [35] is a method that helps in having a more linear behaviour in the space between different classes. Originally, mixup creates new representations, as well as new targets, as shown in Eq. 3, by selecting randomly two indices $i$ and $j$, with $\lambda \sim Beta(\alpha, \alpha)$ being sampled for each pair of indices, with $Beta$ being the Beta distribution and $\alpha \in [0, \infty]$ being a hyperparameter.

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1 - \lambda) x_j \\ \tilde{y} &= \lambda y_i + (1 - \lambda) y_j \end{aligned} \quad (3)$$

While mixup encourages the NN model to behave linearly between samples and can improve generalization, this is not the reason it is used here. The goal here is to better separate different classes, and especially normal traffic from different attack classes. As such, mixup is repurposed into Sepmix as shown in Eq. 4, where $i$ is the index of a randomly selected sample, and $c$ is the closest sample that is of a different class. $\lambda \in [0, 1]$ is also fixed instead of being sampled from a $Beta$ distribution and becomes a hyperparameter. The goal is thus to bring closer to $\tilde{x}$ any sample $x_j$ if $y_j = y_i$ and push further away any sample $x_k$ if $y_k \neq y_i$.

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1 - \lambda) x_c \\ \tilde{y} &= y_i \end{aligned} \tag{4}$$

Sepmix essentially creates virtual samples between classes that will be used in the supervised contrastive loss to make representations more compact. As a consequence, this also allows to artificially increase the distance between samples of different classes, especially in the case of normal traffic that often overlaps with all other classes. In the case of scenario 2 where unidentified attacks exist in the training data but are unlabeled, they can be used as the closest sample with Sepmix to improve performance and generalization. Therefore, Sepmix allows to leverage unlabeled data to both make known classes more compact, and better separate them from unknown classes.

The supervised contrastive loss then uses the new mixed samples to represent positives and negatives. This new loss is shown in Eq. 5, where $i \in I \equiv \{1...N\}$ is the index of a sample in a batch of size $N$ and represents the chosen anchor, $mI$ represents the batch of newly created mixed samples, $mP(i) \equiv \{p \in mI : y_p = y_i\}$ represents the set of mixed positives. $z_i$ is defined analogously to Eq. 1, while $z_p$ are all mixed positives with regard to $z_i$ and $z_a$ are all mixed samples excluding $z_i$.

$$\mathcal{L}_{msupcon} = \sum_{i \in I} \frac{-1}{|mP(i)|} \sum_{p \in mP(i)} log \frac{exp(z_i \cdot z_p / \tau)}{\sum_{a \in mI} exp(z_i \cdot z_a / \tau)} \tag{5}$$

Finally, adding VNE to the loss shown in Eq. 5, the complete loss formulation is shown in Eq. 6, with $\alpha$ influencing the impact of $\mathcal{L}_{VNE}$. If $\alpha > 0$, this forces the method to increase the rank of $Z$ (this increases independence of the different $z_i$). Practically, this reduces the number of neurons in CE that will be equal to 0, which leads to representations of higher quality.

$$\mathcal{L}_{SECL} = \mathcal{L}_{msupcon} - \alpha \mathcal{L}_{VNE} \tag{6}$$

# 4 EXPERIMENTAL SETUP

## 4.1 Working Environment

All experiments were performed on a Linux machine with 64Gb RAM, an 8 core AMD Ryzen 9 5900HX CPU and a NVIDIA 3080 GPU.

All models were run using Python 3.11, *PyTorch*[2] 2.2.0, *PyTorch Lightning*[3] 2.1.4 and *PyTorch Metric Learning*[4] 2.4.1. All implementations and instructions to reproduce experiments will be available on GitLab[5] before publication.

## 4.2 Dataset Pre-processing

All datasets were split using a stratified scheme into 70% train (60% train and 10% validation) and 30% test sets.

For the WADI dataset, features such as Row number, date and timestamps were removed. Four features (`2_LS_001_AL`, `2_LS_002_AL`, `2_P_001_STATUS`, `2_P_002_STATUS`) were removed because they do not have any values. Attack labels were attributed using the recorded beginning and end times of the attacks. Finally, features having a small number of unique values (including most features regarding motor valves, generally named `X_MV_XXX_STATUS`, with `X` being a number) were one-hot encoded. The resulting dataset has 124 features.

For the UNSW-NB15 dataset, features such as IP addresses, timestamps, `attack_cat` were removed, while categorical features or features having a small number of unique values, were one-hot encoded. The resulting dataset has 229 features.

For the CIC-IDS2017 dataset, two features and 5792 instances were removed because of problematic or missing values. A further eight features were removed because they only had one value. The resulting dataset has 70 features.

## 4.3 Unknown classes setup

In order to simulate new unknown classes and still retain the ability to easily evaluate the performance of the tested approaches, the easiest method is to simulate unknown classes by removing them from the train sets. For the scenario 1, the unknown classes were completely removed from the train sets. For the scenario 2, unknown classes data was kept but labels were removed by assigning the label $-1$. During training, all samples with a label of $-1$ can only be mixed with other samples that possess a correct label by using Sepmix.

## 4.4 Evaluation methodology

To evaluate the performance of the proposed approach in both scenarios, results were averaged on two runs with different initializations of the three datasets, i.e., different classes were randomly selected to be removed. In order to keep a sufficient number of classes for training, only up to a third of the attack classes were removed: 5 for WADI and CIC-IDS2017, and 3 for UNSW-NB15. This is a big difference compared to OSL approaches that limit themselves to leaving out a single class from the dataset, which makes it unclear if their approach would be able to detect multiple ZDAs, even without distinguishing them.

In both scenarios, the proposed approach was compared to a supervised baseline. This is a NN using a cross-entropy loss with the same architecture as CE in the proposed approach (without the projection head) *trained knowing all classes (even those unknown*

[2] https://pytorch.org/
[3] https://www.pytorchlightning.ai/
[4] https://github.com/KevinMusgrave/pytorch-metric-learning
[5] https://gitlab.com/RobinKD/secl

*for SECL). The comparison to this supervised baseline gives an upper bound to the performance that the approach has to get close to.*

Metrics used to evaluate performance are the same as what is generally used in multi-class supervised learning. Accuracy will be used to give a general idea of the performance. F1-score is also chosen because it relays information about two metrics important in Intrusion Detection: the Detection Rate (or Recall) and the False Alarm Rate (the opposite of Precision). Therefore, the higher the F1-score, the higher the Detection Rate and the lower the False Alarm Rate. Results given for F1-scores will be micro-averaged, i.e., averaged while taking into account class proportions.

## 4.5 Hyperparameter tuning

Multiple hyperparameters are used in SECL and need to be adjusted for optimal performance. In this case, there are six hyperparameters that have the biggest impact on performance and need to be carefully selected:

- The contrastive loss temperature. Varying this hyperparameter will have an impact on the loss, artificially increasing or decreasing it, thus also impacting computed gradients in the same way. The lower the temperature, the higher the loss, and the more representations will be pushed together in the case of positives and pushed apart in the case of negatives. Experiments showed that a temperature around 0.1 was optimal.
- The architecture of the CE. It needs to be complex enough to learn rich representations, while not being too complex that training becomes too unstable. The problem of instability is further amplified by regularization mechanisms and thus the architecture has to be chosen carefully. Experiments showed that a five-layered network (512, 1024, 2048, 4096, 2048), with dropout layers and ReLU non-linearities between each layer produced rich enough representations with enough stability to train successfully.
- The $\lambda$ used for Sepmix. The selected $\lambda$ influences how close the mixed sample is to the original sample: the higher the $\lambda$, the closer it is. Experiments showed that $\lambda$ lower than 0.5 tends to impede learning because mixed samples are too far from the original sample and the training process becomes too complex. Values ranging between 0.6 and 0.9 have a similar impact, depending on the chosen $\alpha$ for VNE.
- The $\alpha$ used for VNE. Both positive and negatives values are possible, but only positive values force the CE to learn richer representations. Values ranging between 0.1 and 0.3 tend to have a similar impact, depending on the chosen $\lambda$.
- Both K for K-Means and K-Nearest Neighbors. Experiments showed that fixing K-Nearest Neighbors' K to a small number, e.g., 5 allowed to reduce the impact of normal traffic's high prevalence. For K-Means, the K value can be obtained using the elbow method.

When using a lower value for Sepmix $\lambda$, e.g., 0.6, this will increase the effect of regularization and VNE's $\alpha$ is best chosen smaller, e.g., 0.1. The opposite is also true.

The hyperparameter values used for experiments shown in Section 5 are: 0.1 for temperature, (512, 1024, 2048, 4096, 2048) for the CE architecture, 0.75 for $\lambda$, 0.3 for $\alpha$, 5 for the K of K-Nearest Neighbors and the K for K-Means has been obtained with the elbow method.

Table 2: Accuracy of baselines on all datasets

| Dataset | Baseline | |
|---|---|---|
| | Dummy | Supervised |
| **WADI** | 0.9895 | 0.9994 |
| **UNSW-NB15** | 0.8735 | 0.9882 |
| **CIC-IDS2017** | 0.8030 | 0.9955 |

**Values were rounded to the fourth decimal**

# 5 RESULTS

First, in order to properly evaluate the performance of the proposed approaches, the supervised baseline *trained knowing all classes* need to be evaluated. Additionally, a Dummy baseline (only predicting as normal traffic) has been added to show prevalence of normal traffic. Any method should at least be able to improve over this Dummy baseline. Results obtained are shown in Table 2.

It can be seen from the Dummy baseline that normal traffic is highly prevalent, especially for the WADI dataset. Although detrimental to performance and evaluation, this high imbalance is a fundamental characteristic of Intrusion Detection problems and should be expected from IDS datasets to properly represent realistic traffic.

For the proposed approach, SECL, the performance depends on the scenario, as well as the number of classes that were removed or unlabeled for training. Table 3 shows the performance of SECL on all combinations of datasets and scenarios with regard to the number of classes that were removed or unlabeled for training. The first observation is that performance of SECL is better in scenario 2 compared to scenario 1 for all datasets. This means that SECL is able to leverage the unlabeled data through Sepmix to better differentiate known attacks and normal traffic from unknown attacks.

A second result is that SECL's accuracy remains mostly stable and tends to only slightly degrade as the number of unknown attacks increases across all datasets, although this decrease appears to generally be lower in scenario 2. Nevertheless, this reduction is somewhat unsurprising because the fewer the classes, the lower the quality of the learned representations will be.

In order to better compare the proposed approach with both baselines, more detailed results are presented in Section 5.1 and Section 5.2 using F1-score that takes into account both Precision and Recall.

## 5.1 Scenario 1

As a reminder, the scenario 1 is the scenario in which new attacks are completely unknown, and thus are not in the training dataset. Therefore, results in this Section show the ability of SECL to detect multiple ZDAs after an initial training using a supervised dataset.

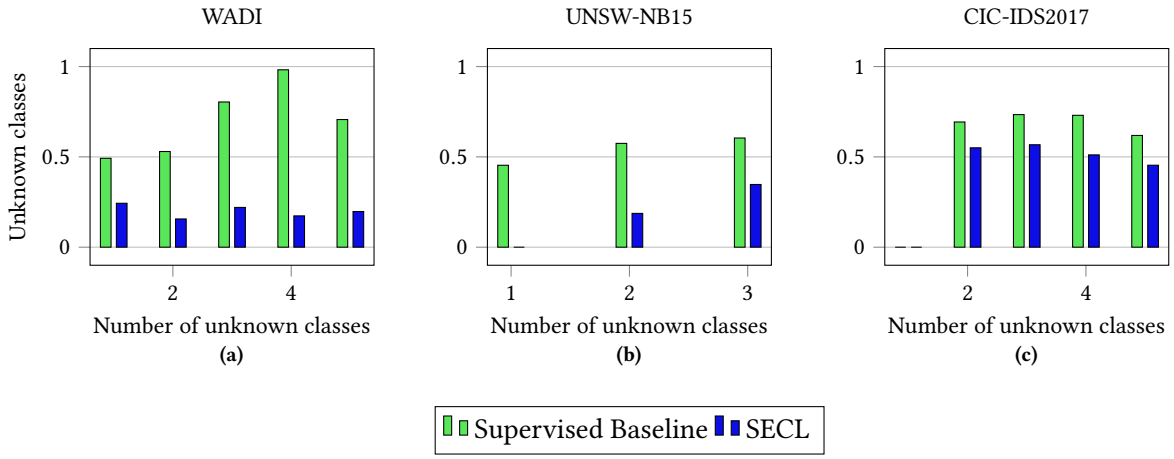


**Figure 2: F1-score of SECL against the supervised baseline in scenario 1 on WADI, UNSW-NB15 and CIC-IDS2017, depending on the number of unknown classes**

In order to get a better picture of the actual performance and ability to generalize of SECL, F1-score of SECL and the supervised baseline on the attacks unknown to SECL are shown in Figure 2.

From Figure 2a and Figure 2c, it is shown that while SECL is able to detect part of the new attacks consistently, although not at the level of a fully supervised model. In Figure 2b, SECL almost completely missed unknown attacks when 1 attack was removed. This might partly be a bias induced by the fact that particularly hard to detect classes were removed, as exemplified by the lower F1-score of the supervised baseline.

From Table 4, it is visible that considering both known classes and unknown attacks, SECL is consistently close to the supervised baseline, and sometimes even better for WADI.

SECL is consistently able to detect ZDAs, except in cases where the supervised baseline also struggles in detecting attacks it learned. Furthermore, it can even be better at detecting known attacks than a supervised model that is trained knowing all classes. Although most important to detect ZDAs, regularization can also impact and improve detection of known classes, especially when these classes have outliers.

Finally, it seems that the performance might decrease as the number of unknown attacks increases, although this is less apparent on UNSW-NB15 and CIC-IDS2017. While SECL might indeed be less effective in detecting new attacks when their number increases, another possibility is that removing new attacks completely from the datasets makes the training datasets less diverse, and thus SECL learns lower quality representations. Using realistic data with a much higher number of classes might not lead to this decrease in performance.

## 5.2 Scenario 2

As a reminder, the scenario 2 is the scenario in which new attacks are present in the training dataset, but are unlabeled. This scenario shows how an IDS initially trained with a supervised dataset would be able to incrementally learn using unlabeled data, collected while in operation, to better detect both known classes and ZDAs. While

Table 3: Accuracy of SECL for both scenarios on all datasets, depending on the number of unknown classes

| Dataset | Scenario | Number of unknown classes | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| **WADI** | Scenario 1 | 0.9988 | 0.9990 | 0.9978 | 0.9964 | 0.9955 |
| | Scenario 2 | 0.9995 | 0.9993 | 0.9994 | 0.9991 | 0.9987 |
| **UNSW-NB15** | Scenario 1 | 0.9855 | 0.9843 | 0.9803 | X | X |
| | Scenario 2 | 0.9866 | 0.9855 | 0.9859 | X | X |
| **CIC-IDS2017** | Scenario 1 | 0.9949 | 0.9769 | 0.9877 | 0.9805 | 0.9835 |
| | Scenario 2 | 0.9955 | 0.9949 | 0.9910 | 0.9938 | 0.9935 |

**Values were rounded to the fourth decimal. Experiments stopped at three classes for UNSW-NB15.**

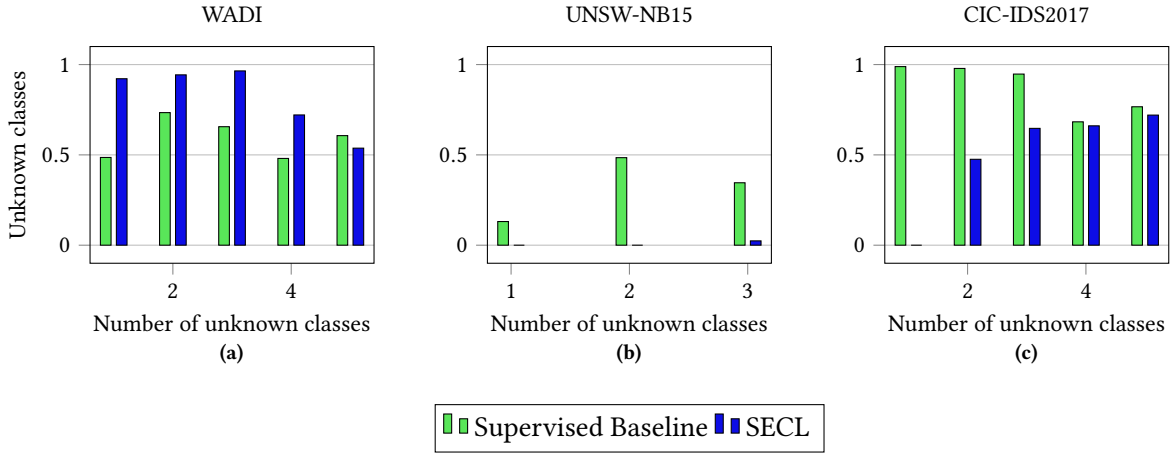


**Figure 3: F1-score of SECL against the supervised baseline in scenario 2 on WADI, UNSW-NB15 and CIC-IDS2017, depending on the number of unknown classes**

this is harder to learn on these attacks than by relying on labels, it allows to leverage their data through Sepmix to potentially learn to differentiate them without the need to identify them. If a sample from a known class and a sample from a ZDA are used by Sepmix to create a virtual sample, all other than this known class will try to move away from the virtual sample, thus improving detection of ZDAs without actually needing to identify them. Detection of ZDAs depending on the number of unknown classes during training are shown in Figure 3.

From Figure 3a, it is shown that SECL is almost always better at detecting unknown attacks that the supervised baseline that was trained knowing the labels. In Figure 3c, SECL also seems quite effective at detecting unknown attacks, although results are below or close to the supervised baseline.

Table 5 shows that SECL is consistently better than the supervised baseline for WADI, and quite close for the other two datasets. This means that being able to leverage unknown attacks during training, even without labels, allows to learn representations of much higher quality and increases the ability of the approach to generalize and detect ZDAs.

A conjecture about why SECL is able to better detect both known and unknown attacks on WADI than a supervised method is that SECL overfits less and is thus more robust to outliers. Because the WADI dataset resembles time-series, traffic at the beginning of an attack is much closer to normal traffic than when the attack has completely impacted the system. As such, if a supervised method is unable to learn from the beginning of the attack, it will generally miss it because of overfitting. SECL, however, will still be able to distinguish the beginning of attacks because they are simulated through Sepmix which can mix normal traffic and attack traffic. Dropout and VNE further help in reducing overfitting.

Finally, results obtained by SECL on all classes show that the performance seems even more stable as the number of unlabeled attacks increases than it was in scenario 1. This is a very promising result, because this means that SECL might be able to leverage a

Table 4: F1-score of SECL and the supervised baseline for scenario 1 on all datasets, depending on the number of unknown classes

| Dataset | Model | Number of unknown classes | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| **WADI** | Baseline | 0.9974 | 0.9974 | 0.9974 | 0.9974 | 0.9974 |
| | SECL | 0.9985 | 0.9987 | 0.9971 | 0.9950 | 0.9941 |
| **UNSW-NB15** | Baseline | 0.9874 | 0.9874 | 0.9874 | X | X |
| | SECL | 0.9840 | 0.9824 | 0.9784 | X | X |
| **CIC-IDS2017** | Baseline | 0.9950 | 0.9950 | 0.9950 | 0.9950 | 0.9950 |
| | SECL | 0.9943 | 0.9749 | 0.9869 | 0.9800 | 0.9825 |

**Values were rounded to the fourth decimal. Experiments stopped at three classes for UNSW-NB15.**

very high number of unlabeled classes to further enrich the learned representations and better detect both known classes and ZDAs.

### 5.3 Ablation study of the regularization method

To determine the influence of each component of the new regularization method combining dropout, VNE and Sepmix, experiments were run removing either one of the three components.

Table 6: F1-score of SECL, removing different components of the regularization method.

| Ablation | Attacks | |
|---|---|---|
| | Unknown | All |
| **No VNE** | 0.7877 | 0.7793 |
| **No Sepmix** | 0.8968 | 0.7867 |
| **No dropout** | 0.9452 | 0.8227 |
| **SECL** | 0.9906 | 0.9083 |

**Values were rounded to the fourth decimal**

Table 6 shows results on detection of ZDAs on WADI with three classes unlabeled in the training set (scenario 2). It can clearly be seen that removing any component of the new regularization method greatly decreases performance, be it for detection of ZDAs or for all attacks.

All three components seem to have a very big impact on the F1-score for detecting both unknown attacks and all classes. Of the three components, dropout seems to have the lowest impact. This is expected because it is known to help regularizing in-distribution samples. While it also seems to help in detecting ZDAs, its effect is more limited than both Sepmix and VNE. Although Sepmix and VNE seem to have a similar impact on all classes, Sepmix seems to have less impact on detecting ZDAs than VNE. An hypothesis is that increasing the quality of the learned representations have more effect than actually making classes more compact and further apart. While Sepmix allows to both make known classes more compact and separate them from ZDAs, this does not actually increase the quality of the representations. On the opposite, VNE that explicity forces SECL to learn richer representations has a much higher impact on detecting ZDAs.

## 6 CONCLUSION AND FUTURE WORK

Contrary to state-of-the-art Intrusion Detection Systems based on Open-Set Learning that are able to only deject a single unknown class, this paper introduced SECL, a method designed to detect and classify Zero-Day attacks using Contrastive Learning and a new regularization method that combines dropout, Sepmix, and VNE. All three components are differently but conjointly helping SECL to better detect ZDAs while retaining a performance on known classes similar to supervised methods.

Two scenarios were considered: unknown attacks are completely absent during training, and unknown attacks are present in the training data, but were never identified, and thus are unlabeled. Results on scenario 1 show that the proposed approach is quite consistently able to detect unknown attacks, even as their number increases. For scenario 2, the proposed approach is able to leverage unlabeled attacks during training and is consistently close to, if not even better, in performance, than a fully supervised method trained knowing all classes.

Furthermore, results obtained by SECL on scenario 2 suggest that SECL might be able to incrementally learn using a high number of unlabeled classes to further increase performance in detecting both known attacks and ZDAs. Therefore, SECL provides a first step towards building next-generation IDSs, able to detect both known and Zero-Day attacks by relying on the ever-increasing volume of traffic to incrementally train.

As future work, three main improvements are considered. First, the proposed setup leverages a combination of K-Means and K-Nearest Neighbors algorithms during testing, which is computationally slower than inference in Deep Learning methods. A possible extension would be to perform clustering through the Contrastive Encoder, as shown in [22]. Secondly, the proposed approach would

Table 5: F1-score of SECL and the supervised baseline for scenario 2 on all datasets, depending on the number of unknown classes

| Dataset | Model | Number of unknown classes | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| **WADI** | Baseline | 0.9974 | 0.9974 | 0.9974 | 0.9974 | 0.9974 |
| | SECL | 0.9994 | 0.9992 | 0.9992 | 0.9989 | 0.9984 |
| **UNSW-NB15** | Baseline | 0.9874 | 0.9874 | 0.9874 | X | X |
| | SECL | 0.9852 | 0.9838 | 0.9845 | X | X |
| **CIC-IDS2017** | Baseline | 0.9950 | 0.9950 | 0.9950 | 0.9950 | 0.9950 |
| | SECL | 0.9947 | 0.9944 | 0.9905 | 0.9930 | 0.9927 |

**Values were rounded to the fourth decimal. Experiments stopped at three classes for UNSW-NB15.**

be closer to real-world adoption once Incremental Learning is added, in order to more easily integrate newly identified attacks. Finally, it would be interesting to see if integrating additional cybersecurity knowledge into the training process, for example, with CVSS scores, would help in better detecting and recognizing known attacks and ZDAs that are more severe for the monitored environment.

## ACKNOWLEDGMENTS

This work is supported by the Chair of Naval Cyber Defence and its partners Ecole Navale, ENSTA-Bretagne, IMT-Atlantique, Naval Group and Thales.

# REFERENCES


[1] 1999. KDD Cup 99 Data. [Online]. Available:http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[2] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P. Mathur. 2017. WADI. In *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*. https://doi.org/10.1145/3055366.3055375

[3] Abhijit Bendale and Terrance Boult. 2015. Towards Open World Recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2015.7298799

[4] T. E. Boult, S. Cruz, A.R. Dhamija, M. Gunther, J. Henrydoss, and W.J. Scheirer. 2019. Learning and the Unknown: Surveying Steps Toward Open World Recognition. *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (2019), 9801–9807. https://doi.org/10.1609/aaai.v33i01.33019801

[5] Kaidi Cao, Maria Brbic, and Jure Leskovec. 2022. Open-World Semi-Supervised Learning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=O-r8LOR-CCA

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. *CoRR* (2020). arXiv:2002.05709v3 [cs.LG] http://arxiv.org/abs/2002.05709v3

[7] Andrea Corsini and Shanchieh Jay Yang. 2023. Are Existing Out-Of-Distribution Techniques Suitable for Network Intrusion Detection?. In *2023 IEEE Conference on Communications and Network Security (CNS)*. 1–9. https://doi.org/10.1109/cns59707.2023.10288685

[8] Gideon Creech and Jiankun Hu. 2013. Generation of a new IDS test dataset: Time to retire the KDD collection. In *2013 IEEE Wireless Communications and Networking Conference (WCNC)*. 4487–4492. https://doi.org/10.1109/wcnc.2013.6555301

[9] Sajad Darabi, Shayan Fazeli, Ali Pazoki, Sriram Sankararaman, and Majid Sarrafzadeh. 2021. Contrastive Mixup: Self- and Semi-Supervised Learning for Tabular Domain. *CoRR* (2021). arXiv:2108.12296 [cs.LG] http://arxiv.org/abs/2108.12296v2

[10] Mohamed Amine Ferrag, Leandros Maglaras, Sotiris Moschoyiannis, and Helge Janicke. 2020. Deep Learning for Cyber Security Intrusion Detection: Approaches, Datasets, and Comparative Study. *Journal of Information Security and Applications* 50, 102419 (2020). https://doi.org/10.1016/j.jisa.2019.102419

[11] Kazuki Hara and Kohei Shiomoto. 2020. Intrusion Detection System using Semi-Supervised Learning with Adversarial Auto-encoder. In *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*. 1–8. https://doi.org/10.1109/noms47738.2020.9110343

[12] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2019. Momentum Contrast for Unsupervised Visual Representation Learning. *CoRR* (2019). arXiv:1911.05722v3 [cs.CV] http://arxiv.org/abs/1911.05722v3

[13] Hanan Hindy, David Brosset, Ethan Bayne, Amar Kumar Seeam, Christos Tachtatzis, Robert Atkinson, and Xavier Bellekens. 2020. A Taxonomy of Network Threats and the Effect of Current Datasets on Intrusion Detection Systems. *IEEE Access* 8 (2020), 104650–104675. https://doi.org/10.1109/access.2020.3000179

[14] A. Hussain, F. Aguiló-Gost, E. Simó-Mezquita, E. Marín-Tordera, and X. Massip. 2023. An NIDS for Known and Zero-Day Anomalies. In *2023 19th International Conference on the Design of Reliable Communication Networks (DRCN)*. 1–7. https://doi.org/10.1109/drcn57075.2023.10108319

[15] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2020. A Survey on Contrastive Self-Supervised Learning. *CoRR* (2020). arXiv:2011.00362 [cs.CV] http://arxiv.org/abs/2011.00362v3

[16] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. 2020. Hard Negative Mixing for Contrastive Learning. *CoRR* (2020). arXiv:2010.01028 [cs.CV] http://arxiv.org/abs/2010.01028v2

[17] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised Contrastive Learning. *CoRR* (2020). arXiv:2004.11362 [cs.LG] http://arxiv.org/abs/2004.11362v5

[18] Jaeill Kim, Suhyun Kang, Duhun Hwang, Jungwook Shin, and Wonjong Rhee. 2023. VNE: An Effective Method for Improving Deep Representation by Manipulating Eigenvalue Distribution. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 3799–3810. https://doi.org/10.1109/cvpr52729.2023.00370

[19] Satoru Koda and Ikuya Morikawa. 2023. OOD-Robust Boosting Tree for Intrusion Detection Systems. In *2023 International Joint Conference on Neural Networks (IJCNN)*. 01–10. https://doi.org/10.1109/ijcnn54540.2023.10191603

[20] Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. 2021. i-Mix: A Domain-Agnostic Strategy for Contrastive Representation Learning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=T6AxtOaWydQ

[21] Dominik Lewy and Jacek Mańdziuk. 2022. An Overview of Mixing Augmentation Methods and Augmentation Strategies. *Artificial Intelligence Review* 56, 3 (2022), 2111–2169. https://doi.org/10.1007/s10462-022-10227-z

[22] Zengyi Li, Yubei Chen, Yann LeCun, and Friedrich T. Sommer. 2022. Neural Manifold Clustering and Embedding. *CoRR* (2022). arXiv:2201.10000 [cs.LG] http://arxiv.org/abs/2201.10000v1

[23] Manuel Lopez-Martin, Antonio Sanchez-Esguevillas, Juan Ignacio Arribas, and Belen Carro. 2022. Supervised Contrastive Learning Over Prototype-Label Embeddings for Network Intrusion Detection. *Information Fusion* 79 (2022), 200–228. https://doi.org/10.1016/j.inffus.2021.09.014

[24] Nicolas Michel, Romain Negrel, Giovanni Chierchia, and Jean-François Bercher. 2022. Contrastive Learning for Online Semi-Supervised General Continual Learning. *CoRR* (2022). arXiv:2207.05615 [cs.LG] http://arxiv.org/abs/2207.05615v1

[25] Nour Moustafa and Jill Slay. 2015. 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)*. 1–6. https://doi.org/10.1109/milcis.2015.7348942

[26] Mohanad Sarhan, Siamak Layeghy, Marcus Gallagher, and Marius Portmann. 2023. From Zero-Shot Machine Learning To Zero-Day Attack Detection. *International Journal of Information Security* 22, 4 (2023), 947–959. https://doi.org/10.1007/s10207-023-00676-0

[27] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult. 2013. Toward Open Set Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 7 (2013), 1757–1772. https://doi.org/10.1109/tpami.2012.256

[28] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. 108–116. https://doi.org/10.5220/0006639801080116

[29] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a Simple Way To Prevent Neural Networks From Overfitting. *Journal of Machine Learning Research* 15, 56 (2014), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

[30] Yiyou Sun and Yixuan Li. 2023. Opencon: Open-World Contrastive Learning. *Transactions on Machine Learning Research* (2023). https://openreview.net/forum?id=2wWJxtpFer

[31] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. 2009. A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. https://doi.org/10.1109/cisda.2009.5356528

[32] Yingjie Tian and Yuqi Zhang. 2022. A Comprehensive Survey on Regularization Strategies in Machine Learning. *Information Fusion* 80 (2022), 146–166. https://doi.org/10.1016/j.inffus.2021.11.005

[33] Hong Xuan, Abby Stylianou, and Robert Pless. 2020. Improved Embeddings with Easy Positive Triplet Mining. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2463–2471. https://doi.org/10.1109/wacv45572.2020.9093432

[34] Yawei Yue, Xingshu Chen, Zhenhui Han, Xuemei Zeng, and Yi Zhu. 2022. Contrastive Learning Enhanced Intrusion Detection. *IEEE Transactions on Network and Service Management* 19, 4 (2022), 4232–4247. https://doi.org/10.1109/tnsm.2022.3218843

[35] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2017. Mixup: Beyond Empirical Risk Minimization. *CoRR* (2017). arXiv:1710.09412 [cs.LG] http://arxiv.org/abs/1710.09412v2