An Investigation into the Performance of Non-Contrastive Self-Supervised Learning Methods for Network Intrusion Detection *

Hamed Fard[1][0009−0007−2365−4313], Tobias Schalau[1][0000−0001−6881−1852], and Gerhard Wunder[1][0009−0001−0850−8816]

Freie Universität, Berlin Germany
{h.habibi.fard,g.wunder}@fu-berlin.de

Abstract. Network intrusion detection, a well-explored cybersecurity field, has predominantly relied on supervised learning algorithms in the past two decades. However, their limitations in detecting only known anomalies prompt the exploration of alternative approaches. Motivated by the success of self-supervised learning in computer vision, there is a rising interest in adapting this paradigm for network intrusion detection. While prior research mainly delved into contrastive self-supervised methods, the efficacy of non-contrastive methods, in conjunction with encoder architectures serving as the representation learning backbone and augmentation strategies that determine what is learned, remains unclear for effective attack detection. This paper compares the performance of five non-contrastive self-supervised learning methods using three encoder architectures and six augmentation strategies. Ninety experiments are systematically conducted on two network intrusion detection datasets, UNSW-NB15 and 5G-NIDD. For each self-supervised model, the combination of encoder architecture and augmentation method yielding the highest average precision, recall, F1-score, and AUCROC is reported. Furthermore, by comparing the best-performing models to two unsupervised baselines, DeepSVDD, and an Autoencoder, we showcase the competitiveness of the non-contrastive methods for attack detection. Code at: https://github.com/renje4z335jh4/non_contrastive_SSL_NIDS

Keywords: Network Intrusion Detection · Self-Supervised Learning · Data Augmentation.

1 Introduction

In the face of a rising tide of security threats targeting the internet and computer networks, the need for developing flexible and adaptive security approaches is

* Hamed Fard and Gerhard Wunder were supported by the Federal Ministry of Education and Research of Germany (BMBF) in the program of "Souverän". Digital. Vernetzt.", joint projects "UltraSec: Sicherheitsarchitektur für eine UWB-basierte Anwendungsplattform" under project identification number 16KIS1682, and "6G-RIC: 6G Research and Innovation Cluster" under project identification number 16KISK025.

of paramount importance. The swift evolution of network technologies has increased the complexity and severity of attacks [14]. In light of this dynamic landscape, the adoption of Network Intrusion Detection System (NIDS) [8] has become prevalent as an effective strategy to counter the expanding threat scenario. NIDS that use supervised methods have been the subject of extensive research over the past two decades [11, 34]. However, the requirement for extensive labeled data in training poses challenges due to cost and time implications. [1]

Self-Supervised Learning (SSL) has become increasingly prominent in recent years, offering an effective remedy for the labeled data scarcity challenge across various domains. By leveraging the underlying structure and patterns in the data, SSL learns meaningful representations without the need for labeled data. Subsequently, these acquired representations are useful in other downstream tasks, such as anomaly detection [15]. Many recent SSL methods, particularly those relying on a joint-embedding architecture, share a common goal: learning representations that remain invariant under various distortions (data augmentations). In other words, these methods seek to generate similar embeddings for different augmented views of the same sample [32]. Contrastive methods define positive and negative sample pairs through data augmentation, seeking to bring the output embeddings of positive pairs into closer proximity while simultaneously pushing negative pairs further apart [6]. This process requires comparing each sample with many others to work effectively. However, discarding negative samples and solely minimizing the distance between positive pairs during training can lead to the learned representation collapsing into a constant solution, where all inputs map to the same output [4]. To overcome this limitation, the computer vision (CV) community has introduced a set of SSL models, including BYOL [13], SimSiam [7], Barlow Twins [36], VICReg [5], and W-MSE [10]. These models are collectively referred to as non-contrastive methods because they require no negative samples, differing primarily in how they avoid representation collapse. Nonetheless, non-contrastive SSL models rely on two crucial elements: a) data augmentations, which are essential for regulating the degree of invariance beneficial for downstream tasks, and b) encoder architectures that act as the representation learning backbone in SSL models. Unlike CV, where specific data augmentations or encoder architectures have been established, and different non-contrastive SSL models are commonly compared to each other [5], to the best of our knowledge, the practice of comparing these models, along with determining suitable augmentation strategies or encoder architectures for non-contrastive SSL in NID, remains unclear. Therefore, this paper aims to investigate the interplay between augmentation, encoder, and non-contrastive SSL models by proposing a two-stage pipeline. In the initial stage, an informative representation of only normal network traffic data without labels is learned, involving augmentations, encoders, and non-contrastive SSL models. In the sec-

[1] Alternatively, semi-supervised learning methods have shown promising performance while utilizing a few labeled samples [2]. However, this paper focuses on self-supervised learning, predicated on the assumption of label-free training.

ond stage, a K-means detector is introduced to distinguish between benign and attack data. For augmentation, four strategies from prior NID research and two new strategies from the tabular domain, not previously applied in NID, are utilized. Regarding the encoders, three different architectures serve as the representation learning backbone for the five non-contrastive models mentioned above. Ninety different combinations are systematically investigated, stemming from the permutations of augmentation techniques, encoder architectures, and non-contrastive SSL models. The best results are reported, with the most suitable combination of augmentation methods and encoder architectures being determined for each non-contrastive SSL model. Performance is assessed using the metrics precision, recall, F1-score, and AUCROC on two publicly available NID datasets. To the best of our knowledge, we are the first to conduct a comparative analysis of non-contrastive SSL models in NID, specifically examining their performance under different augmentation methods and encoder architectures.

The paper is structured as follows: In Section 2, we review previous studies, emphasizing similarities and distinctions between our work and prior research. Section 3 outlines the augmentation methods, encoder architectures, SSL models, and the K-means detector employed in our study. Our experimental setup is detailed in Section 4. The results of our study are presented in Section 5. The paper concludes with Section 6.

2 Related Work

In this section, prior related research on non-contrastive SSL is discussed with a focus on the used SSL model, augmentation strategy, and encoder architecture.

Wang et al. [31] were the first to adopt the BYOL model from CV for the task of NID. Their approach involves transforming network traffic samples into grayscale images, incorporating standard computer vision augmentations such as flipping, cropping, and introducing a method called *Random Shuffle*, which shuffles values within each sample. However, the authors did not provide a clear demonstration of the specific benefits of this augmentation strategy for the detection task. They also argued against using *Gaussian Noise* for augmentation. In addition, the paper investigates the impact of six different encoders, all rooted in the field of CV, and notes that the BoTNet attains the highest performance metric. The direct application of a CV pipeline for NID has also been questioned in [19], where a zero-masking strategy (also referred to as *Zero Out Noise*) is proposed for augmentation instead of relying solely on CV augmentations. BYOL is also employed in android malware detection, as demonstrated in [33], utilizing a TextCNN as an encoder and incorporating two augmentation strategies: *Gaussian Noise* and row- or column-wise feature masking. Recently, BYOL has been adapted for the encrypted network classification task in [29], where data augmentation operates by dividing a flow of packets into sub-flows and using one sub-flow as an augmented version of another. These sub-flows are created through an incremental sampling strategy described in [29]. Apart from BYOL, VICReg is the only other non-contrastive SSL model applied in the domain of

NID in [21], where the authors used the*Swap Noise* augmentation strategy from the tabular domain [30]. This technique involves randomly swapping a small portion of the columns between two samples to generate noisy augmented samples for training. For the encoder, an MLP is employed.

In summary, previous research exploring the performance of non-contrastive SSL models for NID commonly employed a single model. They either replicated an entire CV pipeline or adopted a singular augmentation strategy along with a lone encoder. In addition, these studies consistently incorporated a supervised linear classifier in their detection stage, implying that access to a labeled subset of the dataset was assumed during the finetuning stage. In contrast, this paper conducts a) a comparative analysis of the performance among different non-contrastive SSL models using six distinct augmentation strategies and three diverse encoder architectures, and b) employs an unsupervised linear classifier (K-means) in the detection stage, rendering it label-free.

3 Method

3.1 Augmentations

As previously emphasized, the choice of augmentation methods is pivotal in shaping the SSL objective, as it dictates what the non-contrastive SSL models learn. In this section, we explain the considered augmentation methods.

Swap Noise. Given a traffic network sample $i \sim D$ from dataset D with $i \in \mathbb{R}^{d_D}$, where d_D is the number of features of sample i. To generate an augmented version of this sample i', each feature of i is randomly replaced with a feature at the same position from other samples in D with probability p sampled from a Bernoulli distribution [30, 35, 3, 28]. This procedure is given by

$$i' = i \odot (1 - m) + j \odot m, \tag{1}$$

where $m \in \{0, 1\}^{d_D}$ is a binary mask vector with each element drawn from a Bernoulli distribution, j is a feature vector where each feature is randomly sampled from the original data within the same feature and $\odot$ is the element-wise multiplication.

Zero Out Noise. Similar to *Swap Noise*, features are randomly replaced by zeros in this augmentation. The generation of an augmented sample i' from a sample i is then given by

$$i' = i \odot (1 - m), \tag{2}$$

where m is again a binary vector sampled from a Bernoulli distribution with parameter p [30, 19].

Gaussian Noise. In addition to these replacement methods, we can add Gaussian noise onto randomly selected feature values [30, 22]. The noise is sampled from a normal distribution with $\epsilon = N(\mu, \sigma^2)$, where $\mu \in \mathbb{R}$ is the mean and $\sigma^2 \in \mathbb{R}_{>0}$ is the variance. Formally, this is given by

$$i' = i + \overrightarrow{\epsilon} \odot m, \tag{3}$$

where $\overrightarrow{\epsilon}$ is a vector of values, with each value sampled from the normal distribution $N(\mu, \sigma^2)$ and m is the binary mask vector sampled from a Bernoulli distribution with parameter p.

Random Shuffle. The former augmentation methods alter the features' values. In contrast to the former augmentations, we can randomly shuffle the features' positions within a sample i to generate the augmented version of the sample [31]. For this shuffling, a version of the Fisher-Yates algorithm given in Appendix B is used to generate the augmented version of sample i.

Subsets. Instead of creating two views, in this augmentation the features of D are split into k subsets, before being fed into the encoder f_θ [30]. Each subset consists of a set of features that can overlap with a neighbor subset with a defined percentage of the subset's number of features. We randomly shuffle the dataset features at the beginning of each run to remove any negative bias created by the order of features before building the subsets.

The subsets can be seen as different views of the original sample fed into the model. In contrast to the previously mentioned augmentation methods, *Subsets* automatically creates more than one view and thus is not required to be executed multiple times. With $k > 2$ as the number of subsets, more than two views are obtained. Therefore, we compute the pairwise loss of all views and take the mean as the final loss. At test time, the samples must be split into subsets similar to those at training time, because the trained encoder is adjusted to the subset's shape. Therefore, each subset $s_1, s_2, \ldots, s_k$ is processed by the encoder f_θ to generate a representation for each subset $y_1, y_2, \ldots, y_k = f_\theta(s_1), f_\theta(s_2), \ldots, f_\theta(s_k)$. Then, these representations of the subsets are aggregated using the representation's element-wise mean, forming the final representation for the downstream task.

Mixup. Each former-explained augmentation method operates in the input space, implying that the augmentations take a raw network traffic sample and transform it into one or multiple views of this sample. In contrast, the *Mixup* augmentation operates in the representation space [35, 28]. Thus, the encoders simply receive two copies of the input sample and generate the representations $y = f_\theta(i), y' = f_\theta(i)$. Then, *Mixup* creates a convex combination between y and another randomly selected representation of the current batch y_j. This augmentation is mathematically described by

$$\bar{y} = \alpha * y + (1 - \alpha) * y_j \tag{4}$$

Similarly, the second representation y' is augmented with a different randomly selected representation of the batch.

The different augmentation strategies are further illustrated in Figure 1.

3.2 Encoders

Therefore, each subset $s_1, s_2, \ldots, s_k$ is processed by the encoder f_θ to generate a representation for each subset $y_1, y_2, \ldots, y_k = f_\theta(s_1), f_\theta(s_2), \ldots, f_\theta(s_k)$. Then,

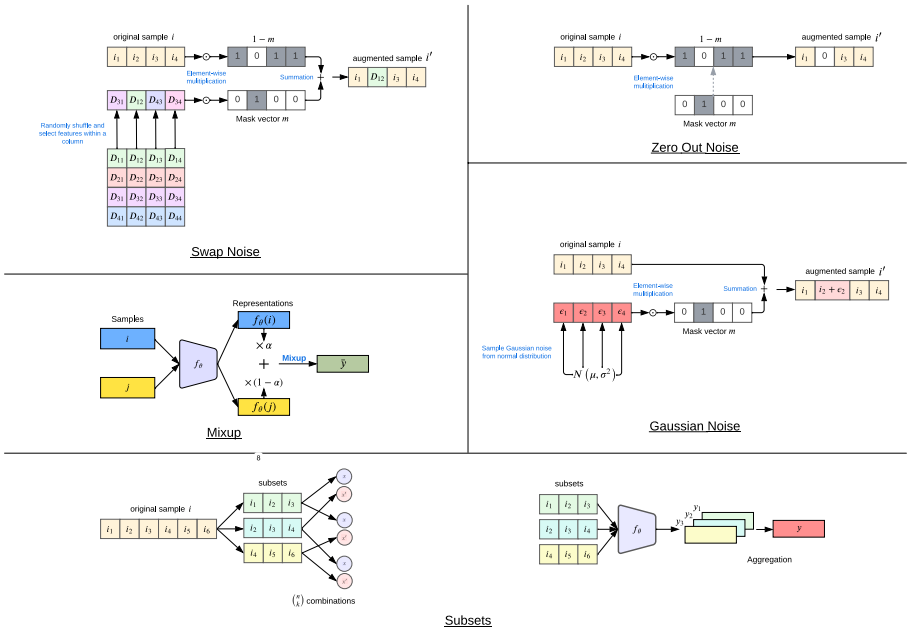

Fig. 1. Visualisation of different augmentation strategies. Swap Noise: each feature of sample i is randomly replaced with a feature from the same position in other samples, with probability p from a Bernoulli distribution. m is a binary mask vector with elements drawn from a Bernoulli distribution, and $\odot$ represents element-wise multiplication. Zero Out Noise: to generate the augmented view i', features of sample i are multiplied element-wise by 1 minus the binary mask vector. Gaussian Noise: $\overrightarrow{\epsilon}$ is a vector of values, with each value sampled from the normal distribution. This vector is element-wise multiplied with a binary mask vector and summed with the original sample vector i to generate the augmented sample. Mixup: operates in the representation space where the encoder f_θ receives two copies of the representations $y = f_\theta(i)$ and $y' = f_\theta(i')$. Mixup creates a convex combination between y and another randomly selected representation of the current batch y_j. Similarly, the second representation y' is augmented with a different randomly selected representation of the batch. Subsets: dataset features are split into k subsets before being fed into the encoder f_θ. Each subset can overlap with a neighboring subset by a defined percentage of features. For $k > 2$, more than two views are obtained. Each subset is processed by the encoder f_θ to generate representations $y_1, y_2, \ldots, y_k$. These representations are aggregated using their element-wise mean, forming the final representation for the downstream task.

these representations of the subsets are aggregated using the representation's element-wise mean, forming the final representation for the downstream task.

The representations generated by the SSL model's encoder play a crucial role in improving the performance of the downstream task. To this end, three different encoder architectures are chosen to serve as the representation learning backbone for the non-contrastive SSL models. The details of each architecture are explained in the following.

CNN. For the Convolutional Neural Networks (CNNs) architecture, the procedure described in [19] is adhered to, where the authors reshaped a network traffic sample with d_D features to $H \times W \times C = 1 \times d_D \times 1$. Through this processing, we can use a $2D$ convolutional layer with filters of height $H = 1$ and arbitrary width W. The precise architecture is given in Appendix 8.

MLP. The encoder consists of four fully connected layers, with the first layer being the input layer, followed by two hidden layers and the output layer. After each of the first three layers, we perform batch normalization followed by applying a ReLU activation function before feeding the output of the current layer to the next layer. The embedding dimension is set to 256 for all layers. Consequently, the input dimension of the first input layer equals the number of features from the input sample, whereas all remaining layers have an input dimension of 256. The MLP architecture is adapted from [3].

FT-T. The Feature Tokenizer Transformer (FT-Transformer) is a supervised transformer model introduced for tabular data consisting of a Feature Tokenizer and a Transformer [12]. We decided to take advantage of the FT-Transformer because, unlike typical Transformers used e.g. in [31], the FT-Transformer is capable of handling numerical and categorical data. Given that the pre-training of the encoder is label-free, we exclude the classification token utilized by the prediction head of the FT-Transformer to integrate it into our pipeline. The Feature Tokenizer creates an embedding of an input network traffic sample by separately processing numerical and categorical features. Afterward, the embeddings of all features are stacked upon each other, which forms the final embedding of the input sample. The embedding is further processed by the Transformer part of the encoder, which consists of a stack of Transformer layers. The structural architecture of the Transformer layers remained unaltered, following the specifications in the original paper [12]. In our implementation, we set the embedding dimension of the feature tokenizer to 32 and the number of self-attention heads for the Multi-Head Self-Attention mechanism to 4. In addition, we chose 4 Transformer layers, encoding the input embedding. The output of the Transformer encoder is then flattened for the subsequent computations. Furthermore, we added a dropout of 0.1 into each attention and feedforward sub-layer, similar to the original implementation [12].

The introduced encoders replace the original encoders in the implementations of the SSL models in our framework. This adaption allows the models to operate on network traffic data. To ensure a fair and competitive comparison, the architecture of the encoders described above is kept identical for each SSL model.

3.3 Non-Contrastive SSL Models

As mentioned in Section 1, non-contrastive SSL models minimize the distance of representations obtained from different augmented versions of a sample, which is realized through a joint embedding architecture. However, a shortcoming of joint embedding architectures is a phenomenon known as collapse, where the two branches ignore the inputs and produce constant output vectors (trivial solutions). In the this subsection, we delve into the distinct features of these models with a particular emphasis on their role in preventing collapse, which are illustrated in Figure 2.

The two augmented views x, x' are fed to an encoder f with weights θ which yields the representations $y = f_\theta(x)$, $y' = f_\theta(x')$. Then, y and y' are further processed by the network h (referred to as the projector) with weights ϕ. In our case, f can be any one of three encoders described in Section 3.2 and h is an MLP (two fully-connected layers with batch normalization and ReLU activation) with embedding dimension 256 for each model to ensure a fair comparison. After this step, different criteria are applied to the projector embeddings z and z':

BYOL. Collapse is prevented through architectural modifications, where in one branch, the weights θ_m for the encoder f and ϕ_m for the projector h are the estimated moving averages of their respective weights θ and ϕ in the other branch. One branch incorporates an additional predictor, denoted as g with weights ψ, to map the output of one network to the other, resulting in an asymmetric architecture. Finally, the output embeddings of the two branches are feature-wise normalized (F-normed) [2], and the similarity loss is computed as the mean-squared error (mse) between them [13].

SimSiam. The estimated moving average operation from BYOL is omitted because it was found to be unnecessary for preventing representation collapse [7]. Similar to BYOL, SimSiam includes a predictor network in one branch and a stop-gradient operation in the other branch [5]. The stop-gradient operation allows the branch with the predictor to be optimized with the projector output of the other branch as the target, but not the other way around.

Barlow Twins. The design of the objective function, rather than architectural modifications, is responsible for preventing collapse. The objective function assesses the cross-correlation matrix between the outputs of the two branches with the goal of minimizing the deviation from the identity matrix [36]. It comprises two key terms: an invariance term (inv) that aims to set the diagonal elements of the cross-correlation matrix to 1 and a decorrelation term (c), which decorrelates pairs of different dimensions within the batch-wise normalized (B-Norm) embeddings [5], i.e., it aims to set the off-diagonal elements of the cross-correlation matrix to 0.

VICReg. Similar to Barlow Twins, VICReg avoids collapse through its objective function, which balances three essential components: a variance term, a covariance term, and an invariance term. The variance and covariance of each branch undergo independent regularization through v and cov, respectively. The

[2] In our implementation, F-norm always refers to ℓ_2 normalization.

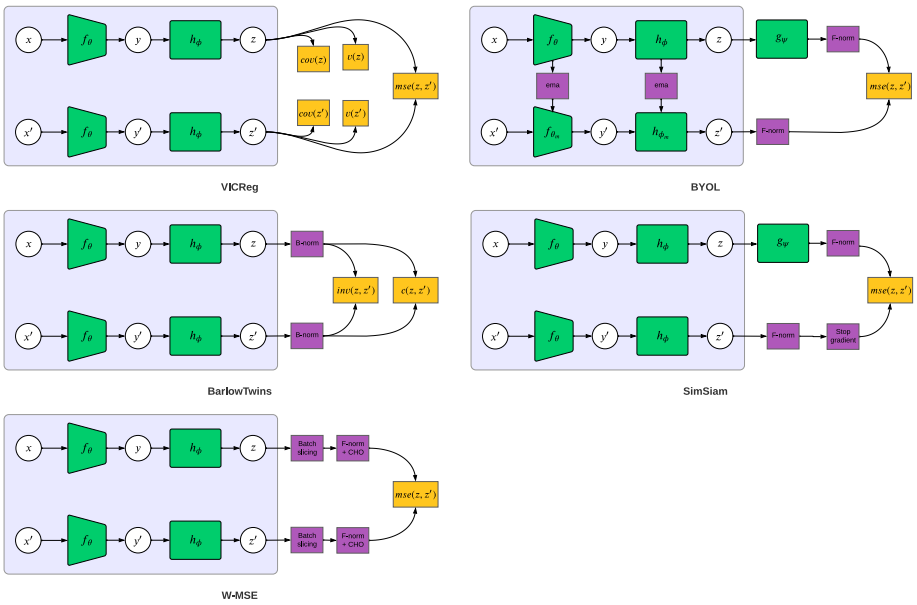

Fig. 2. Comparison of the different non-contrastive SSL models. The two augmented views x, x' are fed to an encoder f (can be an MLP, CNN, or FT-T) with weights θ which yields the representations $y = f_\theta(x)$, $y' = f_\theta(x')$. Then, y and y' are further processed by the network h with weights ϕ. h is an MLP (two fully-connected layers with batch normalization and ReLU activation). After this step, different criteria are applied to the projector embeddings z and z'. VICReg: regularizes the variance and covariance of each branch independently with v and cov, respectively. The invariance term is determined as the mean-squared distance between each pair of vectors z and z'. The final loss is the weighted sum of these three terms. BYOL: one branch incorporates an additional predictor, denoted as g with weights ψ, to map the output of one network to the other, resulting in an asymmetric architecture. The output embeddings of the two branches are feature-wise normalized (F-normed) and the similarity loss is computed as the mean-squared error (mse) between them. Barlow Twins: its objective function assesses the cross-correlation matrix between the outputs of the two branches and has two terms: an invariance term (inv) that aims to set the diagonal elements of the cross-correlation matrix to 1 and a decorrelation term (c), which decorrelates pairs of different dimensions within the batch-wise normalized (B-Norm) embeddings. SimSiam: adds a predictor network in one branch and a stop-gradient operation in the other, omitting BYOL's moving average. W-MSE: applies batch slicing and a Cholesky decomposition-based whitening transformation to F-normed embeddings. The loss is the mean-squared error (mse) between whitened, normalized embeddings of the two branches.

variance regularization term v imposes a constraint on the variance along the batch dimension, ensuring it exceeds a specified threshold for every embedding dimension. Simultaneously, the covariance regularization term cov is defined as the sum of the squared off-diagonal coefficients of the covariance matrix. Moreover, both the variance and covariance regularization terms are computed independently for each branch. The invariance term is determined as the mean-squared distance between each pair of vectors z and z'. The final loss is the weighted sum of these three terms [5].

W-MSE. Similar to VICReg and Barlow Twins, W-MSE prevents collapse through its objective function. This involves a batch slicing operation that reorganizes batches of projector output embeddings z and z' into smaller sub-batches [16]. Following this, a Cholesky decomposition-based whitening transformation is applied to the F-normed embeddings of each sub-batch [5]. Finally, the loss is computed as the mean squared error between the whitened, normalized embeddings of the two branches [10].

3.4 Classifier

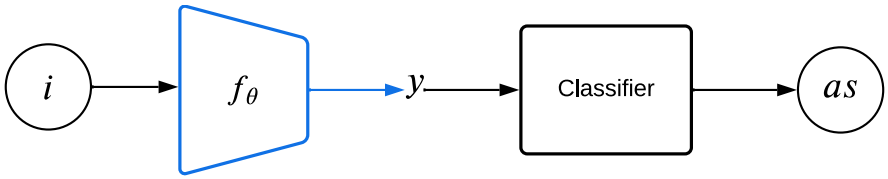

Fig. 3. K-means classifier generates an anomaly score (as) for a network traffic sample i.

After the training phase of the non-contrastive SSL models has completed, only the encoder f_θ is kept, and all other parts of each model are discarded [17]. The weights of the encoder are frozen, and a simple classifier is trained on top of the frozen representation of the data, as shown in Figure 3. Training a linear classifier on top of a pre-trained encoder and using a labeled set of the dataset for fine-tuning on the downstream task is the most common approach in computer vision, which is also used in NID by [31, 33, 29, 21]. However, in this paper, no access to labeled data for fine-tuning is assumed, and the learned representations of the encoders are evaluated via a simple K-means algorithm with a single cluster center in an unsupervised manner, following [26]. The K-means classifier trained on top of the encoder calculates an anomaly score from a given feature representation $y = f_\theta(i)$. The anomaly score ac is defined as the Euclidean distance E between the cluster center and the representation of the test sample ts processed by the encoder f_θ given as

$$as = \|E\|_2 = \|f_\theta(ts) - cc\|_2 \tag{5}$$

where cluster center cc is given by:

$$cc = \frac{1}{n_D} \sum_{j=1}^{n_D} f_\theta(i_j) \tag{6}$$

$D = \{i_1, i_2, \ldots, i_{n_D}\}$ are the n_D data samples used for training the unsupervised classifier.

4 Experiments

In the preceding sections, the four key components of our pipeline were explained: the augmentation methods, encoder architectures, non-contrastive SSL models, and the unsupervised classification head, yielding a complete pipeline suitable for a NID task. We treat every unique combination as an individual experiment, resulting in a total of 90 experiments ($3 \times 6 \times 5$). Each combination is then hyper-optimized, trained, and evaluated on two NIDS datasets under identical conditions to ensure a fair performance comparison. In this section, the evaluation protocol, metrics, datasets, and hyperparameter optimization strategy are examined. The complete pipeline is given in our repository[3].

4.1 Evaluation Protocol and Metrics

Ensuring a fair and consistent comparison of different models is crucial. Unfortunately, in scientific publications, this is not always achieved due to inconsistent hyperparameter tuning or the use of misleading metrics for evaluation, such as accuracy in an unbalanced test set. Moreover, various choices for the class of interest, discrepancies in the training protocol, and different ratios of anomalies in the training set are barriers to comparative evaluations. Therefore, we adhere to the evaluation protocol of [1], where the training set consists exclusively of normal data and the test set includes both normal data and anomalies, with the split ratio as specified in [1]. The positive class is consistently defined as the anomalous class, which forms the basis for the performance metrics. These include the threshold-independent metric AUROC and threshold-dependent metrics, precision, recall (detection rate), and F1-score, computed with an optimal threshold. To account for statistical uncertainty, each combination of augmentation method, encoder architecture, and SSL model is treated as a distinct experiment and executed in 10 runs. The mean and standard deviation are subsequently calculated over all runs, ensuring robust performance evaluations and more accurate estimations of each model's performance.

4.2 Datasets

To benchmark the different SSL models in the pipeline, two NIDS datasets are utilized: UNSW-NB15 [23] and 5G-NIDD [25]. The UNSW-NB15 dataset is well-recognized in the NID domain, proving more suitable for modern NID than the

[3] `https://github.com/renje4z335jh4/non_contrastive_SSL_NIDS`

Table 1. General information on the datasets after preprocessing,

Dataset	Number of Samples	Number of Features	Attack Ratio
UNSW-NB15	154 098	196	0.4437
5G-NIDD	1 215 655	58	0.6072

NSL-KDD dataset [9]. The 5G-NIDD dataset is collected using the 5G Test Network (5GTN) in Finland. A key feature of this dataset is the generation of benign traffic by actual mobile devices in the network, as opposed to simulated traffic. The benign traffic consists of HTTP, HTTPS, SSH, and SFTP traffic. In addition, it includes two attack categories: Port Scan (including SYN Scan, TCP Connect Scan, UDP Scan) and Dos/DDoS (covering ICMP flood, UDP flood, SYN flood, HTTP flood, Slow rate DoS - Slowloris, Slow rate DoS - Torshammer) attacks.

Both datasets were initially cleaned by removing NaN values, dropping duplicated features and samples, and normalizing the values of the features. Furthermore, categorical features were one-hot encoded. Thus, the datasets only consist of numerical features that fit each encoder's input type. Finally, the malicious samples were merged for the NID task. A script for this general preprocessing is provided in our repository[4] [5]. Table 1 summarizes the information of the datasets after the preprocessing step.

4.3 Hyperparameter Optimization

The implementation comprises three sets of variable hyperparameters: model-specific parameters, augmentation parameters, and general training parameters such as learning rate and epochs. Due to the absence of established references guiding the appropriate configuration of these parameters within our pipeline, and recognizing that directly adopting hyperparameter values from original papers may result in biased and non-competitive outcomes, optimization of hyperparameters was conducted for each dataset, model, augmentation, and encoder combination using Tune [18].

The ADAM optimizer was consistently employed for model optimization both in hyperparameter optimization and final runs. During the initial optimization phase, the learning rate was set to 1e−4 and the maximum number of epochs was fixed at 200. Subsequently, optimal model and augmentation parameters were determined using BayesOptSearch [24] with 200 trials, except for the W-MSE model and subset augmentation, which utilized a BasicVariantGenerator due to an integer search space. Following this, the identified optimal parameters were established, and a grid search was conducted on common learning rates (1e−2, 1e−3, 1e−4, 1e−5), each with three trials. Finally, the optimal number

[4] `https://github.com/renje4z335jh4/non_contrastive_SSL_NIDS/blob/main/src/data/process.py`

[5] Aside from the feature preprocessing steps described above, no further feature selection methods were applied

of epochs was determined by training the models with the previously obtained parameters for 200 epochs and three runs. The epoch with the highest average metric over the three runs was considered as the optimal number of epochs. Optimal hyperparameters for each combination are available in our repository[6].

4.4 Experimental Environment

A 64-Bit computer with Debian version 11.7 is used to execute the experiments. As hardware components, the Intel(R) Core(TM) i9-10980XE CPU 3.00GHz with 32 GB RAM and an NVIDIA RTX A5000 GPU with 24 GB VRAM are given.

5 Results

Our experimental results, detailed in Tables 2 and 3, showcase the combinations of augmentation strategy and encoder architecture that yielded the highest average precision, recall, F1-score, and AUCROC metrics for each SSL model on the UNSW-NB15 and 5G-NIDD datasets, respectively.

UNSW-NB15. For the UNSW-NB15 dataset, BYOL exhibits comparably lower performance than other models. Notably, the *Gaussian Noise* augmentation strategy, which was previously deemed unsuitable by [31] yields the best result for BYOL! On the contrary, the *Random Shuffle* augmentation, proposed in [31], consistently underperforms when combined with any encoder or SSL model. As a result, it is not featured in Table 2. The poor performance of the *Random Shuffle* augmentation, compared to other augmentation methods, becomes more evident in Tables 4 and 5, where the performance of BYOL and SimSiam models are compared across all augmentation methods on the UNSW-NB15 dataset. Another absent augmentation strategy in Table 2 is *Swap Noise*, used by the authors in [21] in conjunction with VICReg and an MLP encoder. In our experiments, VICReg attains the highest average precision, F1-Score, and AUCROC with the *Subsets* augmentation. Similarly, Barlow Twins demonstrates the best average performance metrics when combined with the MLP encoder and the *Subsets* augmentation method. In addition, the *Zero Out Noise* augmentation, combined with the SimSiam model and FT-Transformer, yields the highest detection rate.

5G-NIDD. VICReg, combined with a CNN encoder and the *Gaussian Noise* augmentation, achieved the highest average precision, recall, and F1-Score among all other non-contrastive SSL models. This showcases the viability of *Gaussian Noise* as an augmentation strategy. One possible reason for the exclusion of this strategy in [31] could be attributed to either insufficient hyperparameter optimization or its combination with other CV augmentations, such as horizontal flip, vertical flip, random crop, or even *Random Shuffle*, leading to suboptimal

[6] `https://github.com/renje4z335jh4/non_contrastive_SSL_NIDS/blob/main/hyperopt/best_config.yml`

Table 2. Comparison of non-contrastive SSL models with the highest average performance metrics (all with standard deviation) on the UNSW-NB15 dataset.

Model	ENC	AUG	Precision	Recall	F1-Score	AUROC
BYOL	FT-T	GN	0.720 ± 0.021	0.776 ± 0.024	0.747 ± 0.022	0.704 ± 0.037
SimSiam	FT-T	ZON	0.762 ± 0.049	$\mathbf{0.823 \pm 0.053}$	0.791 ± 0.051	0.762 ± 0.048
VICReg	MLP	S	$\mathbf{0.788 \pm 0.059}$	0.810 ± 0.062	$\mathbf{0.798 \pm 0.056}$	$\mathbf{0.786 \pm 0.071}$
BarlowTwins	MLP	S	0.783 ± 0.066	0.809 ± 0.053	0.795 ± 0.056	0.764 ± 0.105
W-MSE	CNN	M	0.763 ± 0.031	0.806 ± 0.019	0.784 ± 0.018	0.756 ± 0.043

Table 3. Comparison of non-contrastive SSL models with the highest average performance metrics (all with standard deviation) on the 5G-NIDD dataset.

Model	ENC	AUG	Precision	Recall	F1-Score	AUROC
BYOL	CNN	SN	0.867 ± 0.015	0.911 ± 0.017	0.888 ± 0.013	0.775 ± 0.017
SimSiam	CNN	S	0.841 ± 0.070	0.877 ± 0.070	0.858 ± 0.069	0.724 ± 0.134
VICReg	CNN	GN	$\mathbf{0.932 \pm 0.010}$	$\mathbf{0.961 \pm 0.026}$	$\mathbf{0.946 \pm 0.009}$	0.908 ± 0.005
BarlowTwins	MLP	M	0.916 ± 0.012	0.909 ± 0.012	0.912 ± 0.009	$\mathbf{0.925 \pm 0.009}$
W-MSE	MLP	M	0.836 ± 0.054	0.891 ± 0.057	0.863 ± 0.056	0.756 ± 0.077

performance. Similar to the results in Table 2, *Random Shuffle* did not achieve competitive results, irrespective of the SSL model it was employed with. Consequently, it is not featured in Table 3 as well. Barlow Twins, in conjunction with the MLP encoder and *Mixup* augmentation, achieved the highest AUCROC. In addition, *Mixup* is the augmentation method that, along with the MLP encoder, achieved the highest metrics for W-MSE on this dataset.

Notably, *Mixup* is the only augmentation method used to operate in the representation space, resulting in competitive outcomes on both datasets. This suggests that incorporating augmentation in the representation space can serve as a practical alternative to augmentations in the input space. Furthermore, the FT-Transformer encoder, which, in combination with BYOL and the SimSiam model, achieved the highest average performance metrics for these models in Table 2, is notably absent in Table 3. i.e., for all other combinations of augmentation and SSL models, the CNN and MLP encoders consistently outperform the FT-Transformer on both datasets. This observation indicates that the choice and optimization of augmentation techniques and hyperparameters may exert a more significant influence than the use of deeper and more complex architectures as backbone encoders. Experimental findings presented in [33] align with this perspective, indicating that employing deeper ResNet architectures as encoders in their BYOL model for android malware detection resulted in diminishing returns in accuracy.

Moreover, comparing the results in Tables 2 and 3 shows that when the augmentation involves *Subsets*, irrespective of the SSL model or encoder used, it leads to high uncertainty across performance metrics. Another drawback of this augmentation is its computational complexity during training time, as the com-

putation of loss involves combinations of projections, which limits the number of subsets for data splitting [30].

As detailed in Section 3.3, BYOL and SimSiam share a similar architecture, differing primarily in the absence of an estimated moving average in SimSiam compared to BYOL. This architectural distinction is a key feature that sets the two models apart. To emphasize the differences between these two SSL models, we also present the performance metrics for both models with the FT-Transformer as its encoder and integrate all augmentation strategies on the UNSW-NB15 dataset. The corresponding results are detailed in Table 4 and 5.

Table 4. Results of the BYOL model with the FT-T as the encoder utilizing different augmentation strategies on the UNSW-NB15 dataset.

Model	ENC	AUG	Precision	Recall	F1-Score	AUROC
BYOL	FT-T	M	0.679 ± 0.014	0.733 ± 0.014	0.705 ± 0.014	0.644 ± 0.039
BYOL	FT-T	RS	0.625 ± 0.005	0.676 ± 0.006	0.650 ± 0.005	0.520 ± 0.012
BYOL	FT-T	S	0.679 ± 0.023	0.733 ± 0.026	0.705 ± 0.024	0.634 ± 0.043
BYOL	FT-T	GN	$\mathbf{0.720 \pm 0.020}$	$\mathbf{0.775 \pm 0.023}$	$\mathbf{0.746 \pm 0.022}$	$\mathbf{0.703 \pm 0.037}$
BYOL	FT-T	SN	0.671 ± 0.007	0.725 ± 0.010	0.697 ± 0.008	0.629 ± 0.013
BYOL	FT-T	ZON	0.680 ± 0.016	0.738 ± 0.016	0.708 ± 0.016	0.649 ± 0.029

Table 5. Results of the SimSiam model with the FT-T as the encoder utilizing different augmentation strategies on the UNSW-NB15 dataset.

Model	ENC	AUG	Precision	Recall	F1-Score	AUROC
SimSiam	FT-T	M	0.657 ± 0.023	0.711 ± 0.024	0.680 ± 0.024	0.591 ± 0.042
SimSiam	FT-T	RS	0.657 ± 0.044	0.711 ± 0.046	0.683 ± 0.045	0.551 ± 0.047
SimSiam	FT-T	S	0.664 ± 0.058	0.718 ± 0.062	0.690 ± 0.059	0.569 ± 0.630
SimSiam	FT-T	GN	0.720 ± 0.021	0.777 ± 0.020	0.747 ± 0.020	0.712 ± 0.039
SimSiam	FT-T	SN	0.672 ± 0.075	0.714 ± 0.062	0.692 ± 0.066	0.597 ± 0.093
SimSiam	FT-T	ZON	$\mathbf{0.761 \pm 0.048}$	$\mathbf{0.823 \pm 0.053}$	$\mathbf{0.791 \pm 0.050}$	$\mathbf{0.762 \pm 0.048}$

In Table 4, notable variations in AUCROC are observed for the BYOL model depending on the augmentation method. Specifically, employing *Random Shuffle* leads to an AUCROC of 0.52, while utilizing *Gaussian Noise* significantly improves the AUCROC to 0.70. A similar trend is evident in the performance of SimSiam, as presented in Table 5. When *Random Shuffle* is employed, an AUCROC of 0.55 is attained. However, opting for the *Gaussian Noise* augmentation with SimSiam results in higher average performance metrics compared to the BYOL model utilizing the same augmentation method.

Comparison with unsupervised baselines: Before concluding with the results section, the aim is to compare the non-contrastive models with the highest average performance metrics against two well-known unsupervised meth-

ods: DeepSVDD and AE. DeepSVDD, a deep learning-based one-classification method, entails mapping input data into a hypersphere with the objective of minimizing the hypersphere's volume. This mapping situates normal samples inside the hypersphere, while anomalies reside outside. AE, a vanilla autoencoder with a reconstruction-based objective, employs an MLP architecture for the encoder and decoder. It is noteworthy that AE has demonstrated superior performance over other sophisticated reconstruction-based methods across diverse datasets, including NID datasets, as detailed in [1]. To ensure fairness and comparability of results, the hyperparameters of the baselines were tuned using Tune, following the same procedure described in Section 4.3. The results of this comparison are outlined in Tables 6 and 7 for the UNSW-NB15 and 5G-NIDD datasets, respectively. In the UNSW-NB15 dataset, VICReg achieves the highest average precision, while for other metrics, the AE consistently outperforms non-contrastive SSL models. DeepSVDD shows less favorable results. A similar pattern is observed for the 5G-NIDD dataset, where AE attains the highest average performance metrics. Although the highest average performance metrics achieved by the non-contrastive SSL models are comparable to those of AE, the results underscore the significance of tuning baseline hyperparameters. Previous studies investigating the performance of non-contrastive SSL models often fail to specify the extent of hyperparameter tuning in their baseline comparisons, potentially creating a misleading sense of confidence.

Table 6. Comparison of non-contrastive SSL models with the highest average performance metrics (all with standard deviation) on the UNSW-NB15 dataset against unsupervised models on the same dataset.

Model	ENC	AUG	Precision	Recall	F1-Score	AUROC
SimSiam	FT-T	ZON	0.762 ± 0.049	0.823 ± 0.053	0.791 ± 0.051	0.762 ± 0.048
VICReg	MLP	S	$\mathbf{0.788 \pm 0.059}$	0.810 ± 0.062	0.798 ± 0.056	0.786 ± 0.071
DeepSVDD	—	—	0.683 ± 0.021	0.735 ± 0.025	0.708 ± 0.023	0.656 ± 0.047
AE	—	—	0.786 ± 0.013	$\mathbf{0.837 \pm 0.029}$	$\mathbf{0.811 \pm 0.018}$	$\mathbf{0.793 \pm 0.024}$

Table 7. Comparison of non-contrastive SSL models with the highest average performance metrics (all with standard deviation) on the 5G-NIDD dataset against unsupervised models on the same dataset.

Model	ENC	AUG	Precision	Recall	F1-Score	AUROC
VICReg	CNN	GN	0.932 ± 0.010	0.961 ± 0.026	0.946 ± 0.009	0.908 ± 0.005
BarlowTwins	MLP	M	0.916 ± 0.012	0.909 ± 0.012	0.912 ± 0.009	0.925 ± 0.009
DeepSVDD	—	—	0.895 ± 0.060	0.937 ± 0.055	0.915 ± 0.057	0.865 ± 0.117
AE	—	—	$\mathbf{0.939 \pm 0.027}$	$\mathbf{0.965 \pm 0.018}$	$\mathbf{0.951 \pm 0.020}$	$\mathbf{0.932 \pm 0.020}$

6 Conclusion

In this paper, we explore the interplay between augmentation methods, encoders, and non-contrastive SSL models. We propose a two-stage pipeline: first, learning a useful representation of normal network traffic in a self-supervised manner; second, freezing the pre-trained encoder weights and using a K-means algorithm to distinguish between benign and attack data. Our empirical findings revealed the poor performance of the *Random Shuffle* method across all SSL models. In contrast, *Gaussian Noise*, previously deemed unsuitable by [31], yielded the best average performance metric for the BYOL model on the UNSW-NB15 dataset. This could be due to insufficient hyperparameter optimization or its combination with other CV augmentations, such as horizontal flip, vertical flip, random crop, or even *Random Shuffle*, leading to suboptimal performance. *Mixup* combined with Barlow Twins achieved the highest AUCROC on the 5G-NIDD dataset, highlighting the effectiveness of augmentation in the representation space as an alternative to augmentations in the sample space. *Subsets*, used with three different SSL models on both datasets, exhibited competitive performance. Notably, in combination with VICReg, it achieved the highest precision, F1-score, and AUCROC on the UNSW-NB15 dataset, though this method led to increased uncertainty in performance metrics. *Zero Out Noise* and *Swap Noise* showed competitive performance only with BYOL and SimSiam models, respectively, and only on the UNSW-NB15 dataset.

While our experiments underscore the importance of augmentation methods, they also reveal two significant drawbacks: a) these methods are not specifically tailored for NID, and b) they do not necessarily satisfy the domain constraints of NID, meaning they are not function-preserving and may lead to the generation of unrealistic samples [27]. Designing NID-specific augmentation methods that satisfy domain constraints for SSL methods is a promising avenue for future research.

Regarding SSL models, VICReg and Barlow Twins consistently attained higher average performance metrics than other SSL models. The asymmetric architectural design choices in SimSiam and BYOL did not offer an advantage over these methods. For encoders, the FT-Transformer demonstrated competitive performance only as the backbone for BYOL and SimSiam models, and only on the UNSW-NB15 dataset. In all other cases, the conceptually simpler MLP and CNN architectures proved more viable.

Finally, this paper compares the performance of non-contrastive SSL models to DeepSVDD and AE. The results show that non-contrastive SSL models outperformed DeepSVDD, while AE achieved higher average performance metrics than non-contrastive SSL models. This difference may be due to the use of a naive K-means detector. Future work could explore the utilization of improved distance metrics, such as the Mahalanobis distance [20], in the K-means detector. Additionally, incorporating more sophisticated unsupervised detectors, such as Isolation Forest or OCSVM, might address the minor performance gap observed between non-contrastive SSL models and reconstruction-based approaches.

A Encoder Structure

Table 8. Architecture of the CNN encoder. *conv* is a convolutional layer followed by a ReLU activation function, and pooling represents a pooling layer. For each layer, the kernel size, number of filters (only for convolutional layers), input shape, and output shape are given for an example network traffic sample with 196 features.

Layer	**Kernel**	**Filter**	**Input**	**Output**
conv1	1×2	32	$1 \times 196 \times 1$	$1 \times 195 \times 32$
conv2	1×2	64	$1 \times 195 \times 32$	$1 \times 194 \times 64$
conv3	1×2	128	$1 \times 194 \times 64$	$1 \times 193 \times 128$
pooling	1×3	–	$1 \times 193 \times 128$	$1 \times 64 \times 128$
conv4	1×2	256	$1 \times 64 \times 128$	$1 \times 63 \times 256$
pooling	1×2	–	$1 \times 63 \times 256$	$1 \times 31 \times 256$
conv5	1×2	512	$1 \times 31 \times 256$	$1 \times 30 \times 512$
pooling	1×4	–	$1 \times 30 \times 512$	$1 \times 7 \times 512$

B Augmentation

Algorithm 1 Pseudocode of Fisher-Yates inspired *Random Shuffle* augmentation method.

Require: $i_j = \{f_j^{(1)}, f_j^{(2)}, \ldots, f_j^{(d_D)}\}$ ▷ network traffic sample i_j composed of d_D features
  for $k = d_D - 1$ to 0 **do**
    $p \leftarrow random\_integer(0, k)$
    $i_j^{(p)}, i_j^{(k)} = i_j^{(k)}, i_j^{(p)}$
  end for

References

1. Alvarez, M., Verdier, J.C., Nkashama, D.K., Frappier, M., Tardif, P.M., Kabanza, F.: A Revealing Large-Scale Evaluation of Unsupervised Anomaly Detection Algorithms (Apr 2022), http://arxiv.org/abs/2204.09825, arXiv:2204.09825 [cs]
2. Apruzzese, G., Laskov, P., Tastemirova, A.: Sok: The impact of unlabelled data in cyberthreat detection. In: 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P). pp. 20–42. IEEE (2022)
3. Bahri, D., Jiang, H., Tay, Y., Metzler, D.: SCARF: Self-Supervised Contrastive Learning using Random Feature Corruption (Mar 2022), http://arxiv.org/abs/2106.15147, arXiv:2106.15147 [cs]

4. Balestriero, R., Ibrahim, M., Sobal, V., Morcos, A., Shekhar, S., Goldstein, T., Bordes, F., Bardes, A., Mialon, G., Tian, Y., et al.: A cookbook of self-supervised learning. arXiv preprint arXiv:2304.12210 (2023)
5. Bardes, A., Ponce, J., LeCun, Y.: Vicreg: Variance-invariance-covariance regularization for self-supervised learning. arXiv preprint arXiv:2105.04906 (2021)
6. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A Simple Framework for Contrastive Learning of Visual Representations (Jun 2020), http://arxiv.org/abs/2002.05709, arXiv:2002.05709 [cs, stat]
7. Chen, X., He, K.: Exploring Simple Siamese Representation Learning (Nov 2020), http://arxiv.org/abs/2011.10566, arXiv:2011.10566 [cs]
8. Denning, D.E.: An intrusion-detection model. IEEE Transactions on software engineering (2), 222–232 (1987)
9. Divekar, A., Parekh, M., Savla, V., Mishra, R., Shirole, M.: Benchmarking datasets for Anomaly-based Network Intrusion Detection: KDD CUP 99 alternatives. In: 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS). pp. 1–8 (Oct 2018). https://doi.org/10.1109/CCCS.2018.8586840, http://arxiv.org/abs/1811.05372, arXiv:1811.05372 [cs, stat]
10. Ermolov, A., Siarohin, A., Sangineto, E., Sebe, N.: Whitening for self-supervised representation learning. In: International Conference on Machine Learning. pp. 3015–3024. PMLR (2021)
11. Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., Vázquez, E.: Anomaly-based network intrusion detection: Techniques, systems and challenges. computers & security **28**(1-2), 18–28 (2009)
12. Gorishniy, Y., Rubachev, I., Khrulkov, V., Babenko, A.: Revisiting Deep Learning Models for Tabular Data (Oct 2023), http://arxiv.org/abs/2106.11959, arXiv:2106.11959 [cs]
13. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P.H., Buchatskaya, E., Doersch, C., Pires, B.A., Guo, Z.D., Azar, M.G., Piot, B., Kavukcuoglu, K., Munos, R., Valko, M.: Bootstrap your own latent: A new approach to self-supervised Learning (Sep 2020), http://arxiv.org/abs/2006.07733, arXiv:2006.07733 [cs, stat]
14. Group, C.: 2023 cyberthreat defense report. https://www.humansecurity.com/hubfs/HUMAN_Report_2023-Cyberthreat-Defense-Report.pdf (2023)
15. Hojjati, H., Ho, T.K.K., Armanfard, N.: Self-supervised anomaly detection: A survey and outlook. arXiv preprint arXiv:2205.05173 (2022)
16. Huang, L., Yang, D., Lang, B., Deng, J.: Decorrelated batch normalization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 791–800 (2018)
17. Jaiswal, A., Babu, A.R., Zadeh, M.Z., Banerjee, D., Makedon, F.: A Survey on Contrastive Self-supervised Learning (Feb 2021), http://arxiv.org/abs/2011.00362, arXiv:2011.00362 [cs]
18. Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J.E., Stoica, I.: Tune: A research platform for distributed model selection and training. arXiv preprint arXiv:1807.05118 (2018)
19. Lotfi, S., Modirrousta, M., Shashaani, S., Amini, S., Shoorehdeli, M.A.: Network intrusion detection with limited labeled data. arXiv preprint arXiv:2209.03147 (2022)
20. Mahalanobis, P.C.: On the generalized distance in statistics. Proceedings of the National Institute of Sciences (Calcutta) **2**, 49–55 (1936)
21. Menon, A.S., Nair, G.: Vicra: Variance-invariance-covariance regularization for attack prediction. In: 2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS). pp. 1075–1080. IEEE (2023)

22. Mirza, B., Syed, T.: SELF-SUPERVISION FOR TABULAR DATA BY LEARNING TO PREDICT ADDITIVE GAUSSIAN NOISE AS PRETEXT (2021)
23. 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). pp. 1–6. IEEE, Canberra, Australia (Nov 2015). https://doi.org/10.1109/MilCIS.2015.7348942, http://ieeexplore.ieee.org/document/7348942/
24. Nogueira, F.: Bayesian Optimization: Open source constrained global optimization tool for Python (2014–), https://github.com/fmfn/BayesianOptimization
25. Samarakoon, S., Siriwardhana, Y., Porambage, P., Liyanage, M., Chang, S.Y., Kim, J., Kim, J., Ylianttila, M.: 5G-NIDD: A Comprehensive Network Intrusion Detection Dataset Generated over 5G Wireless Network (Dec 2022). https://doi.org/10.48550/arXiv.2212.01298, http://arxiv.org/abs/2212.01298, arXiv:2212.01298 [cs]
26. Sehwag, V., Chiang, M., Mittal, P.: SSD: A Unified Framework for Self-Supervised Outlier Detection (Mar 2021), http://arxiv.org/abs/2103.12051, arXiv:2103.12051 [cs]
27. Sheatsley, R., Hoak, B., Pauley, E., Beugin, Y., Weisman, M.J., McDaniel, P.: On the robustness of domain constraints. In: Proceedings of the 2021 ACM SIGSAC conference on computer and communications security. pp. 495–515 (2021)
28. Somepalli, G., Goldblum, M., Schwarzschild, A., Bruss, C.B., Goldstein, T.: SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training (Jun 2021), http://arxiv.org/abs/2106.01342, arXiv:2106.01342 [cs, stat]
29. Towhid, M.S., Shahriar, N.: Encrypted network traffic classification using self-supervised learning. In: 2022 IEEE 8th International Conference on Network Softwarization (NetSoft). pp. 366–374. IEEE (2022)
30. Ucar, T., Hajiramezanali, E., Edwards, L.: SubTab: Subsetting Features of Tabular Data for Self-Supervised Representation Learning (Oct 2021), http://arxiv.org/abs/2110.04361, arXiv:2110.04361 [cs, stat]
31. Wang, Z., Li, Z., Wang, J., Li, D.: Network Intrusion Detection Model Based on Improved BYOL Self-Supervised Learning. Security and Communication Networks **2021**, 1–23 (Oct 2021). https://doi.org/10.1155/2021/9486949, https://www.hindawi.com/journals/scn/2021/9486949/
32. Weng, X., Huang, L., Zhao, L., Anwer, R., Khan, S.H., Shahbaz Khan, F.: An investigation into whitening loss for self-supervised learning. Advances in Neural Information Processing Systems **35**, 29748–29760 (2022)
33. Yang, S., Wang, Y., Xu, H., Xu, F., Chen, M.: An android malware detection and classification approach based on contrastive lerning. Computers & Security **123**, 102915 (2022)
34. Yang, Z., Liu, X., Li, T., Wu, D., Wang, J., Zhao, Y., Han, H.: A systematic literature review of methods and datasets for anomaly-based network intrusion detection. Computers & Security **116**, 102675 (2022)
35. Yoon, J., Jordon, J., Zhang, Y.: VIME: Extending the Success of Self- and Semi-supervised Learning to Tabular Domain (2020)
36. Zbontar, J., Jing, L., Misra, I., LeCun, Y., Deny, S.: Barlow twins: Self-supervised learning via redundancy reduction. In: International Conference on Machine Learning. pp. 12310–12320. PMLR (2021)