# Representation Performance of SimCLR with a Simpler Backbone Network on Intrusion Detection

Haonan Chen
*Faculty of Engineering*
*University of Ottawa*
Ottawa, Canada
hchen312@uottawa.ca

Sayed Us Sadat
*Faculty of Engineering*
*University of Ottawa*
Ottawa, Canada
ssada041@uottawa.ca

Mohammed Sufiyan Ali Banaganapalle
*Faculty of Engineering*
*University of Ottawa*
Ottawa, Canada
mbana080@uottawa.ca

Samee Mohammad Sayeed
*Faculty of Engineering*
*University of Ottawa*
Ottawa, Canada
ssaye028@uottawa.ca

***Abstract*—In the realm of computer and network security, the crucial task of detecting network intrusions is essential to safeguard systems from unauthorized activities. However, a significant challenge arises due to imbalanced datasets, where the occurrence of normal network traffic instances greatly outweighs that of attack instances. Existing conventional methods like oversampling and undersampling have proven ineffective in addressing this challenge. In response, this study proposes leveraging representation learning as a solution. The chosen starting point for this approach is a previously developed intrusion detection model based on SimCLR. Unlike the original method, which employed ResNet as the backbone network, our adaptation opts for a simpler backbone network. This choice is rooted in the belief that a simpler backbone network is better suited for intrusion detection, as ResNet was primarily designed for image data with significantly more features. The study's performance evaluation criteria consist of accuracy, precision, recall, and F1-score, mirroring the metrics used in the baseline model.**

***Index Terms*—intrusion detection; contrastive learning; SimCLR; representation learning**

## I. Introduction

Network intrusion detection is a common challenge in the field of Cybersecurity. Often, one encounters a situation where normal data significantly outweighs instances of attacks, resulting in an imbalanced dataset. This data imbalance creates difficulties in tasks like information retrieval and filtering, ultimately leading to a low-quality dataset. Consequently, machine learning and artificial intelligence models trained on such datasets perform poorly in real-world scenarios. To address this issue, various techniques have been proposed, with common practices including:

- Undersampling: This involves rebalancing the training data by reducing the majority class and increasing the proportion of minority samples.
- Oversampling: In this approach, more minority samples are generated from existing minority samples to balance the dataset.

However, according to the work [1] of Ravid and others, these methods are often ineffective and can sometimes even harm the performance of machine learning models. Therefore, in this paper, we will employ a different approach, contrastive learning, which has been proven to effectively handle imbalanced data problems, as demonstrated by the work [2] of Ziyu and Bingyi.

Unlike self-supervised feature encoding methods like Autoencoders (AE), which rely on reconstruction loss and do not consider other samples, contrastive learning involves selecting or generating sets of positive and negative samples from the same batch. The loss function in contrastive learning is typically a triplet loss calculated based on the distances between the encoded anchor sample and the encoded positive and negative samples.To enhance detection performance, some methods, such as those proposed by Wei and others [3], involve training the encoder along with the classifier, tailoring the sample representations to be more compatible with the classifier's requirements.

Our research aims to adapt a representation learning framework called SimCLR for contrastive learning. SimCLR provides a method for generating and selecting positive and negative samples within the same batch. A previous study [4] applied SimCLR to intrusion detection, but it utilized the original SimCLR with its backbone network, which is designed for datasets with many features. Historically, it was initially designed for image tasks, where feature dimensions are typically much larger than those in intrusion detection. For instance, when an image is resized to 128x128, it may have thousands of features, whereas intrusion detection data usually has only a few hundred. Given the smaller feature dimensions in intrusion detection, we hypothesize improvements can be made by using a simpler network, such as a Multi-Perceptron Network (MLP), as the backbone network.

In this study, we will begin by replicating the approach outlined in Chengcheng's work [4] that introduced a technique for generating augmented perspectives of a reference sample, which will serve as the basis for comparison. Following this, the subsequent steps will closely follow the original SimCLR framework. The replicated method will serve as a benchmark model for comparison against the model we intend to use: a simplified SimCLR with a Multilayer Perceptron (MLP) backbone. We will evaluate performance using metrics such as accuracy, recall, precision, and F1-score within a 5-fold cross-validation setup, and the final metrics will be calculated as the average across the 5-fold experiments.

## II. Review

### A. Intrusion Detection

In Cybersecurity, intrusion pertains to unauthorized activities that can harm an information system, potentially compromising its confidentiality, integrity, or availability [5]. Intrusion detection, in turn, refers to identifying and flagging such malevolent activities within an information system. Intrusion detection systems (IDS) typically fall into two categories:

- Signature-based intrusion detection systems (SIDS) rely on pattern matching. For example, they raise an alarm if they detect specific protocols or IP addresses in certain activities.
- Anomaly-based intrusion detection systems (AIDS), employ methods like machine learning, statistics, or knowledge-based approaches. These methods centre around learning patterns from data.

Conventional SIDS primarily operate by comparing incoming network packets to pre-existing signatures. Consequently, they face challenges when it comes to detecting novel attacks, requiring frequent additions of new signatures by administrators. Therefore, our approach in this work is rooted in AIDS. AIDS methods can be broadly categorized into three types:

- Statistical-based approaches involve constructing a model of normal behaviour by profiling typical activities. Suspicious activities with low probabilities are then identified as potential attacks.
- Machine learning-based techniques build models using extensive datasets to recognize patterns distinguishing attacks from regular activities.
- Knowledge-based methods, often referred to as expert systems, necessitate creating a knowledge base representing legitimate traffic profiles. Any activities falling outside this predetermined profile are classified as attacks.

Statistical methods are known for their high interpretability. Knowledge-based techniques, on the other hand, excel in significantly reducing false alarms. However, both these methods demand substantial domain expertise to create efficient models. In contrast, machine learning and other learnable approaches offer distinct advantages: they do not require extensive professional knowledge, can adapt to evolving threats and attack patterns, and entail less post-deployment maintenance. Additionally, in some learnable methods, such as representation learning and contrastive learning, the laborious task of feature engineering can be mitigated.

Enamul and others [6] have proposed a novel approach to solve the network intrusion detection problem using Least Square Support Vector Machine (LS-SVM), where they were successfully able to investigate the novelty of their methodology in terms of detection accuracy and computational efficiency. They referred to the proposed algorithm as an optimum allocation-based least square support vector machine (OA-LS-SVM) for IDS [6]. Their decision-making process unfolds over two distinct phases. Firstly, they proportioned the complete dataset into several predefined clusters. Next, their novel algorithm randomly samples from these clusters that resemble the dataset's characteristics. Following this, the LS-SVM is employed on these selected samples to identify potential intrusions. They have also claimed their experiment by providing evidence through tests on the KDD 99 dataset, proving the efficiency and robustness of their methodology. A research [7] on a knowledge-based model by Klaus and others has discovered that their methods can handle future alarms more efficiently. They investigated episode rules considering suitability and reported the challenges they faced and the unexpected results gained. Their results show that intrusion detection alarms are very homogeneous and repetitive. The work [8] by Robin and Vern is noteworthy as it uses machine learning techniques to identify the unique challenges associated with employing machine learning in network intrusion detection. They overcame certain Machine Learning method shortcomings such as outlier detection, high cost of errors, semantic gap, etc. They have also noted that diversity in the network also exists, which leads to misconceptions about what anomaly detection technology can realistically achieve in operational environments.

### B. Representation Learning & Related Applications

In contrast to other machine learning and artificial intelligence methods, techniques employed in representation learning typically involve training a model to create representations for each observation. These representations are then used for both training and testing. When dealing with new observations, acquiring representations using the trained representation learning model is necessary. Conversely, methods that do not learn latent spaces generally [9]:

- Utilize the original dataset's features directly.
- Often requires manual feature selection.
- Engage in feature engineering with expert knowledge.

The primary objective of representation learning is to enable a system to automatically learn representations for specific observations, reducing the need for manual feature engineering. Representation learning methods can be categorized into probabilistic models, manifold methods, and methods that directly map observations to representations. Within probabilistic models, we can distinguish:

- Directed Graphical Models, including Principal Components Analysis (PCA) and sparse coding.
- Undirected Graphical Models, where Restricted Boltzmann Machines (RBMs) are commonly applied.

Directed graphical models are frequently employed after a preprocessing stage to reduce noise in the data and address high-dimensional data challenges. Jiali and others [10] claim that integrating Bayesian networks can improve the FAIR model's capacity to manage uncertainty and generate more precise risk evaluations. They describe how Bayesian network architectures can be used to represent and analyze important

FAIR concepts, including threat event frequency, loss magnitude, and loss event frequency. To illustrate how it might be used to estimate the probability of loss events and their possible repercussions, they use their expanded FAIR model with Bayesian networks to a real-world Cybersecurity risk assessment scenario. The case study results are discussed in the paper, emphasising how the Bayesian network technique offers more complex and probabilistic risk assessments.

As for undirected graphical models, Tamer [11] demonstrated the robustness of RBMs as classifiers for detecting anomalies on new datasets. They applied RBMs to the 2018 ISCX dataset, achieving a 0.94 true negative rate for the negative class, which indicates an attack. Arnaldo and Miguel [12] contended that contemporary machine learning methods cannot extract complex structures due to their typically simple internal representations. This limitation does not apply to energy-based algorithms like RBMs. In their research, they developed a systematic approach for training RBMs, achieving a high specificity of 0.96 for the negative class (representing attacks).

Manifold methods refer to nonlinear dimensionality reduction techniques, with common methods including:

- Isomap
- Locally Linear Embedding
- Hessian Eigenmapping
- Spectral Embedding
- Local Tangent Space Alignment
- Multi-dimensional Scaling (MDS)
- t-distributed Stochastic Neighbor Embedding (t-SNE)

Guoping Hou and others [13] argued that dimension reduction methods like PCA and its derivatives are ineffective at capturing nonlinear relationships between features. They underperform compared to feature reduction methods that can identify such nonlinear relationships, especially in high-dimensional data. Their solution involved applying an improved Locally Linear Embedding (LLE) to preprocessed data before feeding it into a Backpropagation Neural Network (BPNN). As a result, they achieved a high detection rate of 94.4%, outperforming the baseline method by nearly 15%.

Ning and others [14] designed a manifold-based detection system capable of identifying adversarial examples generated by GAN models, which traditional machine learning methods cannot distinguish. Their approach combines a manifold component and a machine learning-based classifier to make determinations. Instead of using off-the-shelf manifold methods like Isomap, they designed a scoring criterion to combine the inconsistency between the Intrusion Detection System (IDS) model and the manifold evaluation. The results indicate that their method can successfully detect adversarial samples with an accuracy of approximately 90%.

The models that directly map observations to representations include:

- Auto-Encoders (AE)
- Regularized AE, such as Sparse AE, Denoising AE, and Contrastive AE

In intrusion detection, Youngrok and others [15] found that AEs can accurately and promptly detect unknown attack types, reducing the manual labeling of network flows. They tested various AEs with model structures on multiple datasets and observed that the latent size of an encoder significantly affects detection performance. Their approach consists of two phases: training and selecting the desired model through validation. In the testing phase, instead of a machine learning-based model, they applied a simpler thresholding method to classify normal and attack samples, leading to better performance differentiation. Ultimately, they determined that an AE with 7 layers and a latent size 9 performed the best, achieving an F1 score of 0.895 on the NSL-KDD dataset.

K. Narayana and others [16] developed a hybrid detection system based on a Sparse Auto-Encoder (SAE) with smoothed L1 regularization and a Deep Neural Network (DNN). They argued that adding sparsity constraints to the SAE helps represent input features better than an unregularized SAE, which is prone to overfitting. Their system achieved a detection rate of 99.98% on the NSL-KDD dataset and 99.99% on the UNSW-NB15 dataset.

Ivandro and others [17] combined a Denoising Auto-Encoder (DAE) with a one-hidden-layer Multi-Layer Perceptron (MLP) to detect anomalies in the CICIDS2018 dataset. Their approach resulted in a low false alarm rate and stable classification results. Furthermore, they reduced the input features to 1/10th of the original input, significantly reducing computation time for subsequent classifiers. They achieved an averaged F1 score of 99.69% and a low false alarm rate of 0.0277%.

### *C. Contrastive Representation Learning & Related Applications*

In contrastive representation learning, the primary objective is to create an embedding space where similar observations are brought closer while dissimilar ones are pushed apart [18]. This approach can be employed in both supervised and unsupervised scenarios. Irrespective of the context, the training process invariably entails selecting an anchor sample and adjusting its embedding to be similar to the embeddings of positive samples while ensuring dissimilarity from those of negative samples.

To implement contrastive learning in a representation learning model, a common practice involves replacing the loss function of learnable representation networks, such as Autoencoders, with a contrastive learning loss function, such as the triplet loss. This entails selecting a set of positive and negative samples for comparison with the anchor sample, and these samples are then fed into the loss function to calculate the loss within the same batch. The earliest form of a contrastive loss was defined by Chopra et al [19]. In this framework, you have a set of samples denoted as $x_i \in X$, with each corresponding to a label $y_i \in \{1, 2, ..., L\}$. The objective is to learn a representation learner, as defined by:

$$f_\theta(.) : \chi \to \mathbb{R}^d \tag{1}$$

The aim is to transform any input $x_i$ into a latent representation in which embeddings of samples belonging to the same class exhibit similarity while embeddings of samples from distinct classes exhibit dissimilarity. In pursuit of this goal, contrastive learning operates by considering pairs of inputs $(x_i, x_j)$. To fulfil this objective, the loss function is designed to measure the dissimilarity between embeddings of samples that share the same class:

$$\mathcal{L}_{same} = \sum_{i=0, y_i = y_j}^{n} \| f_\theta(x_i) - f_\theta(x_j) \|_2^2 \tag{2}$$

and the loss of the embeddings of samples with the different classes:

$$\mathcal{L}_{diff} = \sum_{i=0, y_i \neq y_j}^{n} max(0, \epsilon - \| f_\theta(x_i) - f_\theta(x_j) \|_2)^2 \tag{3}$$

Here, we have a hyperparameter denoted as $\epsilon$, which sets a minimum distance between a given sample and other samples. The contrastive loss is derived by combining the two equations mentioned above:

$$\mathcal{L}_{cont}(x_i, x_j, \theta) = \mathcal{L}_{same} + \mathcal{L}_{diff} \tag{4}$$

The concept of contrastive loss typically inspires loss functions developed later. The initial approach was the triplet loss, which aimed to train on images of the same person in various poses and angles. It involved selecting an anchor sample and treating a set of augmented versions of that anchor as positive samples, while other samples served as negatives. However, this method was computationally inefficient due to its reliance on only three samples. To enhance efficiency, the lifted structured loss was introduced to consider all paigenerating augmented views of the anchor sample becomes necessary loss was derived from the triplet loss, allowing for the inclusion of multiple negative samples in a single batch, further improving efficiency. The Noise Contrastive Estimation (NCE) loss was designed to estimate parameters in a statistical model to differentiate target data from noise. Inspired by NCE, InfoNCE employs categorical cross-entropy loss to distinguish positive samples from artificially generated noisy samples.

In the context of contrastive learning, there are two main approaches. The first involves supervised learning with pre-existing labels, enabling direct retrieval of positive and negative samples. In cases where labeled data is unavailable, such as when working with datasets lacking labels, it becomes necessary to generate augmented views of the anchor sample. In intrusion detection, both approaches have been utilized and tend to outperform methods relying solely on machine learning or artificial intelligence techniques.

In the work [3] of Wei and colleagues, they combined a classification loss and triplet loss to train the encoder network. Initially, they used an AE to encode categorical features and concatenated this with numerical features to input into the encoder network, generating a representation embedding. The loss was calculated by combining a classification loss and a triplet loss. Triplet samples, obtained using a triplet generator, were selected based on the anchor sample and used for calculating the triplet loss. The classification loss was determined by passing the representation embedding through a 2-layer Multi-Layer Perceptron (MLP) for observation classification. Their model's performance was compared to five feature selection-based methods and two deep learning methods, and their proposed model outperformed all selected models, with an improvement of around 5-30% in precision, recall, and accuracy.

In the work of Chengcheng and others [4], they adopted a more advanced contrastive learning framework called SimCLR, a state-of-the-art model in computer vision tasks. SimCLR generates augmented views of the anchor sample to serve as positive samples, while all other samples are considered negative. However, instead of directly applying contrastive loss to the latent embeddings, they trained a projector to contrast the projected embeddings. The projector was discarded for downstream tasks, and the latent embeddings were used directly. Notably, intrusion detection data augmentation methods differ from those used in image tasks, as they involve SMOTE and Gaussian noises to generate positive samples for the anchor. They maintained the representation encoder identical to SimCLR, using a ResNet as the encoder. The authors compared their method to directly using ResNet, Attention-LSTM, and CNN-LSTM, finding that their method performed the best, with approximately a 4% improvement in precision, recall, and F1 score, while requiring the least training time compared to others.

Rather than manually generating positive and negative samples by adding Gaussian noises and applying SMOTE, Jian and colleagues [20] used two autoencoders trained on normal and attack samples to generate positive and negative samples. These triplet samples (anchor sample, negative sample, positive sample) were then used to create a correlation matrix, which was fed into a CNN encoder to generate latent embeddings for contrast. A triplet loss function was applied to these latent embeddings. To demonstrate the effectiveness of the method, they conducted an embedding analysis, revealing that on both CICIDS17 and KDDCUP99 datasets, the normal embeddings were clearly separated from the attack embeddings. In the end, they compared their method to other neural network-based methods and found that their approach outperformed all other architectures with a 2-10% improvement in accuracy and F1-score. Furthermore, when compared to the latest state-of-the-art methods, they claimed that their model slightly outperformed other models.

## III. Background

### A. SimCLR

SimCLR, or Simple Contrastive Learning of Representations, is a framework for contrastive learning-based unsupervised visual representation learning. With the use of contrastive learning, it attempts to acquire meaningful representations of images without the need for explicit labels by differentiating between similar and dissimilar image pairs. The SimCLR framework, introduced by Ting Chen, Simon

Kornblith, Mohammad Norouzi, and Geoffrey Hinton [21], presents a simple yet powerful approach to contrastive learning of visual representations. The framework builds on discriminative approaches that learn representations using objective functions similar to those used for supervised learning, but train networks to perform pretext tasks where both the inputs and labels are derived from an unlabeled dataset.

- Contrastive Learning Objective: To learn representations, SimCLR employs a contrastive objective function. It minimizes agreement between views of various images (negative pairs) and increases agreement between differently augmented views of the same image (positive pairs).
- Data Augmentation: An essential component of SimCLR is augmentation. Using a variety of modifications (such as cropping, color distortion, rotation, etc.), it produces many views of the same image, producing positive pairs for training.
- Base Architecture: The fundamental architecture of SimCLR is usually a convolutional neural network (CNN). It creates high-dimensional representations of the visuals via encoding.
- Projection Head and Contrastive Loss: SimCLR maps the high-dimensional representations onto a lower-dimensional space by means of a projection head. After that, it uses a contrastive loss function to maximize agreement for positive pairs and minimize it for negative pairs, such as the NT-Xent (Normalized Temperature-scaled Cross-Entropy) loss.
- Training Process: Using a sizable dataset of unlabeled pictures, SimCLR iteratively optimizes the contrastive loss function to train the neural network.

The SimCLR framework has implications for the broader field of self-supervised learning, suggesting that despite the recent surge in interest, self-supervised learning remains undervalued . The framework's ability to learn meaningful representations without the need for specialized architectures or complex memory banks makes it a promising avenue for further exploration and application in the field of computer vision and representation learning . SimCLR's effectiveness in capturing meaningful visual features has implications for a wide range of applications, from image recognition to natural language processing .

Fig 1. [21] shows a fundamental framework for acquiring visual representations through contrast. To obtain two correlated views, two different data augmentation operators (t  T and t  T) are sampled from the same family of augmentations and applied to each data example. A contrastive loss is used to train a projection head g(·) and a base encoder network f (·) to optimize agreement. Once training is finished, we discard the projection head g(·) and use representation h and encoder f (·) for tasks that come after.

The SimCLR loss function is expressed as follows:

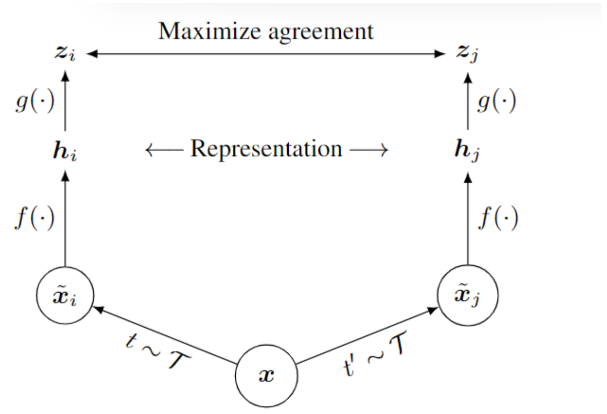

Fig. 1. A simple overview of SimClr

$$\mathcal{L}_{\text{SimCLR}} = -\frac{1}{2N}\sum_{i=1}^{N}\sum_{j=1}^{2}\log\left(\frac{\exp(\text{sim}(z_i, z_j))}{\sum_{k=1}^{2N} \nVdash_{[k \neq i]} \exp(\text{sim}(z_i, z_k))}\right) \tag{5}$$

In this equation, the numerator represents the similarity between the anchor sample and its corresponding views, while the denominator signifies the sum of similarities between the anchor sample and all other samples in the same batch. Unlike the triplet loss, where the anchor, positive, and negative samples are typically derived from the same instance, SimCLR's InfoNCE formulation treats all other samples within the batch as negative examples. This approach is computationally more efficient as it avoids the need for exhaustive comparisons, contributing to the overall effectiveness of the contrastive learning framework.

In summary, contrastive learning of visual representations has advanced significantly with the introduction of the SimCLR framework. SimCLR has shown impressive gains over earlier techniques by streamlining current algorithms and closely examining the consequences of its design decisions, underscoring the possibilities of self-supervised learning in computer vision.

### *B. Resnet*

ResNet, short for Residual Network, is a deep convolutional neural network architecture that introduced the concept of residual learning, which addresses the vanishing gradient problem in very deep networks. ResNet was proposed by Kaiming He, et al. in their paper "Deep Residual Learning for Image Recognition" in 2015. [22]

- Residual Learning: ResNet makes use of residual blocks that include identity mappings or shortcut connections, sometimes referred to as skip connections. By combining the output of a previous layer with the output of a deeper layer, these connections allow the network to learn residual mappings. By doing so, some layers can be avoided and the network is able to learn the remaining data without having to explicitly learn the full transformation.

- Architecture: The concept of stacking residual blocks is the foundation of the ResNet architecture. The structure consists of multiple layers with varying depths, such as ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152. ResNet versions with deeper architectures perform better across a range of tasks, particularly image categorization.
- Building Blocks: Two convolutional layers with batch normalization and ReLU activation, plus a shortcut connection, make up the fundamental building block of a ResNet. Even with more layers added, the network can retain accuracy because to the residual block.
- ResNet frequently employs Global Average Pooling (GAP) in place of fully connected layers in the final layers. In order to aggregate features prior to the final classification, GAP shrinks the spatial dimensions of the feature maps to a 1x1 size for each channel.
- Performance: ResNet outperformed previous architectures in terms of accuracy and convergence speed, achieving state-of-the-art performance on a number of picture classification benchmarks, including ImageNet.

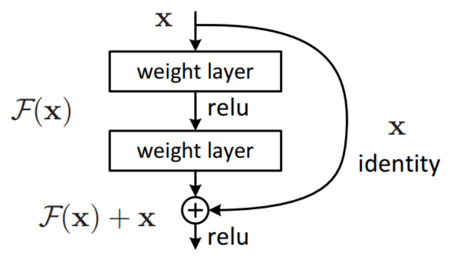
x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Fig. 2. Block Diagram for Residual Learning

Using "shortcut connections," feedforward neural networks can implement the formulation of F(x) +x as shown in Fig 2. [22] Connections that skip one or more layers are known as shortcut connections. [23] [24] [25] In this instance, the shortcut connections just carry out identity mapping, and the stacked layers' outputs are combined with their outputs as shown in Fig 2. Identity shortcut links do not increase computing complexity or introduce additional parameters. Without changing the solvers, the complete network may still be trained end-to-end using SGD with backpropagation, and it is simple to implement using widely available libraries .

The field of computer vision has greatly benefited from ResNet's contributions. ResNet made it possible to train far deeper neural networks by adding skip connections and residual blocks, which produced state-of-the-art results on several image recognition benchmarks, including ImageNet. More efficient and complex models in the field of deep learning have been made possible by its architecture and the idea of residual learning, which have impacted the design of later neural network architectures.

### C. MLP

A basic kind of artificial neural network that is frequently used in supervised learning tasks like regression and classification is called an MLP, or Multilayer Perceptron. [26] It consists of several layers of feedforward-organized neurons (nodes), with each layer's neurons coupled to all of the layers below it.

- Architecture: An MLP is made up of an output layer, an input layer, and one or more hidden layers. Multiple neurons are present in every layer—aside from the input layer—and they are all coupled to every other neuron in the layer above it. Hidden layer neurons introduce non-linearities into the data by using activation functions, which enables the network to learn intricate correlations.
- Feedforward Propagation: This is the method by which information moves from the input layer to the hidden layers and then to the output layer. After receiving inputs and calculating their weighted total, each neuron in a layer applies an activation function before sending the output to the neurons in the layer below.
- Training: Algorithms such as gradient descent and its variations, such as backpropagation, are used to train multilayer parsers. Using backpropagation, the network's weights are adjusted iteratively to minimize loss by computing the gradient of the loss function with respect to those weights.
- Applications: Because of their adaptability, MLPs have been used for a wide range of tasks, including regression issues, image recognition, and natural language processing. However, because of their limits in managing high-dimensional data, they may have difficulty with large-scale problems and complicated data linkages.

In conclusion, the fundamental neural networks known as Multilayer Perceptrons (MLPs) serve as the building blocks for a variety of deep learning models. They can solve a wide range of supervised learning problems and learn complicated relationships in data thanks to their architecture, which consists of numerous layers of interconnected neurons. Although they work well in many situations, their structural limitations make them unsuitable for processing high-dimensional data and may cause problems when dealing with complex patterns. Gaining an understanding of MLPs is a prerequisite for mastering deeper learning's more complex neural network structures.

## IV. Methodology

The methodology's structure is illustrated in Figure 3. Following the train-test split, we initiate the process by preprocessing the training set and creating augmented views based on the processed data. Subsequently, both the preprocessed data and the augment views are fed into the SimCLR framework for training the feature extractors. Once the training phase concludes, we adopt the preprocessing pipeline and the f1 extractor to apply them to the test set, generating representations. These representations are then employed in cross-validating selected classifiers to assess the extractor's performance. The framework employed in this study mirrors that outlined in the referenced paper [4]. Notably, our experimentation covers both the SimCLR with MLP backbone and replication of the ResNet-based approach as detailed in the paper.

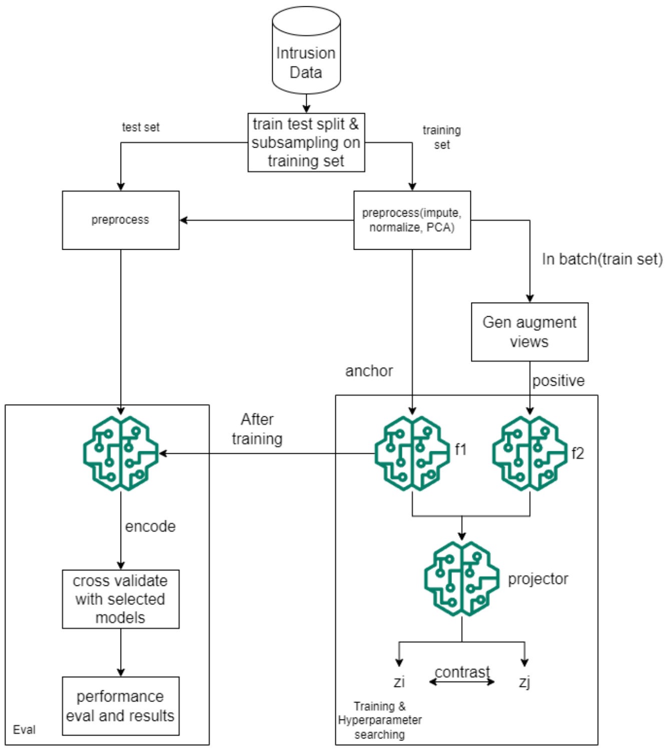
Intrusion Data
train test split & subsampling on training set
test set
training set
preprocess
preprocess(impute, normalize, PCA)
In batch(train set)
Gen augment views
anchor
positive
After training
f1
f2
encode
projector
cross validate with selected models
contrast
zi
zj
performance eval and results
Eval
Training & Hyperparameter searching

Fig. 3. Architecture overview

## A. Experiment settings

The dataset we used is CICIDS2017, which contains 2.8 million samples with 79 numeric features and 1 label for each sample. The label distribution can be found in Figure 4. The specifications of the experimental platform are listed below:

- Operating System: Ubuntu 22.04
- GPU: Nvidia 3090Ti
- RAM: 32GB
- CPU: AMD 5800X3D

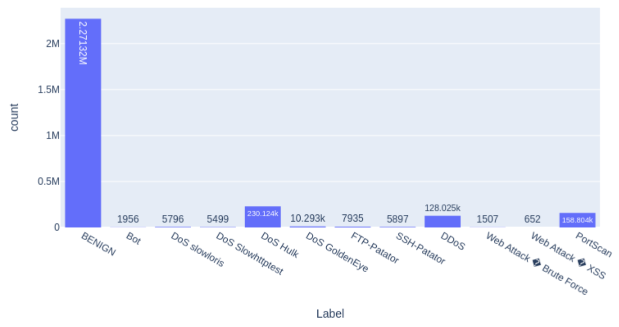

Fig. 4. Label distribution

## B. Train test split & subsampling

Following the customary practice, we allocated 80% of the data for the training set and 20% for the test set. However, during the hyperparameter tuning process, we encountered significant time constraints associated with training a feature extractor in the SimCLR framework. Specifically, it took approximately 16 minutes to train a ResNet-backed feature extractor on a subset of 20,000 samples. Consequently, we opted for subsampling within the training set to alleviate the time-intensive nature of the procedure.

The key steps involve subsampling attack samples, reducing them to half of their original size, except for samples with less than 20k instances. Additionally, benign samples are subsampled to match the quantity of all other attack samples. This approach aims to create a more balanced and manageable training set that accommodates the computational constraints imposed by SimCLR training process. Ultimately, this led to a reduction in the training samples from 2,261,000 to 445,000.

## C. Preprocessing

The preprocessing approach mirrors that of the referenced paper [4], encompassing three key steps:

- Imputation of NAN values involves replacing them with the most frequently occurring values. This assumes that NAN values typically fall within the normal range, resembling the records of benign samples. It's important to note that this assumption may not always hold true.
- Min-max normalization is applied to standardize the data.
- PCA (Principal Component Analysis) downscaling is employed, selecting the first 20 components for subsequent use. The explained variance graph, calculated based on the training samples, is depicted in Figure 5.

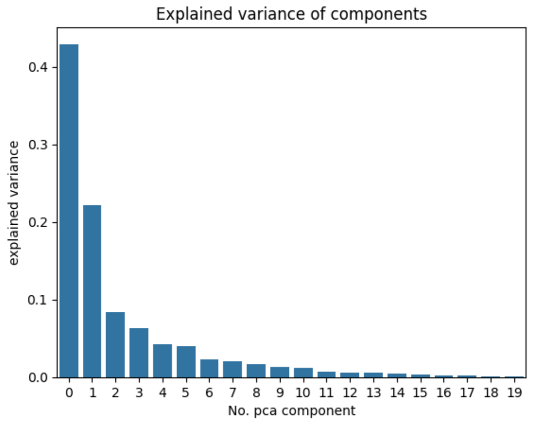

Fig. 5. Explained variance of pca components

## D. Augmentation methods

As outlined in the SimCLR framework, the generation of augmented views for training feature extractors is crucial. Traditional computer vision methods, such as cropping, resizing, and flipping, are commonly employed. However, these methods are not directly applicable to network data. Therefore, in the paper [4], the authors introduced a combination of three augmentation techniques specifically designed for network data:

- SMOTE sampling with the same multiplicity: This method involves generating a number of samples equal to the input, and each sample is calculated based on the five neighbors surrounding the original sample. Importantly, the exact same number of samples is obtained after the sampling process.
- Gaussian noise: A Gaussian noise with parameters $\mu = 0$ and $\sigma = 1$ is added to the SMOTE-sampled data.
- Sequence Inversion: This technique entails flipping all the samples in the original data. In other words, if the original data is represented as $x = [x_1, x_2, ..., x_n]$, it is transformed to $x' = [x_n, ..., x_2, x_1]$.

## E. Detailed SimCLR

In the initial SimCLR framework, a pair of networks, namely a symmetric feature extractor and a projector, is employed. However, in an effort to expedite training and reduce the number of sample pairs, the paper introduces a modification to the SimCLR framework. This alteration involves the use of asymmetric feature extractors: the extractor f1 is utilized for extracting preprocessed training data, while the extractor f2 is dedicated to extracting augment views. Subsequently, the same projector is employed to project these representations into another space for contrast.

The architectural details of our feature extractors, f1 and f2, are depicted in Figure 6 and 7. Since the original preprocessed training data has different channels compared to the vertically stacked augment views (which have 3 channels), we address this by incorporating a convolution layer before both extractors f1 and f2, as illustrated in Figure 6. This convolution layer helps standardize the input for both the ResNet and our MLP-backed extractor. In the case of ResNet-backed extractors, the extractor f1 is constructed using ResNet50 with a single-channel Conv1D layer as the base layer. On the other hand, the extractor f2 utilizes ResNet50 with a three-channel Conv1D layer as the base layer.

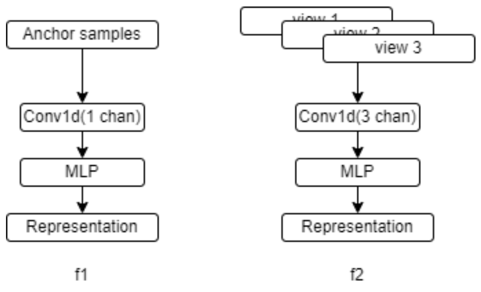

Fig. 6. Extractor architecture

As for the projector, a straightforward linear layer can be employed to map the input to another space. In the referenced paper [4], the projector network is designed with two linear layers interleaved with a layer-normalization layer and a ReLU activation, as illustrated in Figure 7.

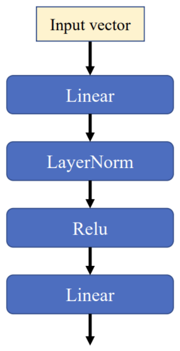

Fig. 7. Projector architecture

## F. Classifiers

Once the training process concludes, the feature extractor, denoted as f1, and the preprocessing pipeline will be sequentially applied to the test set. This sequential application aims to extract representations, which will then be fed into the classifiers, together with their corresponding labels, for the purpose of cross-validation. Subsequently, the performance of the extractors will be benchmarked.

Unless explicitly stated otherwise, all classifiers utilized in this context are default classifiers sourced from sklearn (version 1.3.1). These encompass tree methods such as Decision Tree and Random Forest, non-linear classifiers like MLP , linear classifiers including Logistic Regression and Gaussian Naive Bayes, and not-learning-based methods exemplified by 5-NN and 10-NN. The complete list of classifiers is provided below:

- MLP (max_iter=1500)
- Decision Tree
- Gaussian Naive Bayes
- Logistic Regression (max_iter=500)
- Random Forest
- 5-NN
- 10-NN

## G. Hyperparameter search

For the method mentioned in the paper [4], it's unstated which learning rate and the L2 regularization are applied and also the size of the representation embedding. Hence, in the work of reproduction, we first fixed the learning rate to 0.1 and the L2 regularization to $1 \times 10^{-4}$ to randomly search the embedding size. After getting the optimal embedding size, we fixed it to tune the learning rate and the regularization factor. At last, the following hyperparameters are determined:

- Embedding size: 34
- Learning rate: 0.010848038400629992
- Weight decay: 0.00012209180832556052

For the method with an MLP backbone, since all the hyperparameters are unknown, we first randomly searched all the parameters, then fixed the learning rate and the L2 regularization factor to search other parameters one by one or group by group since some parameters might be dependent, e.g., the conv1d's stride and the kernel size. The final hyperparameters are listed below:

- Learning rate: 0.014968162145540436
- Weight decay: 0.002904245255012199
- Hidden size: 65
- Number of hidden layers: 1
- Embedding size: 39
- Kernel size: 1
- Stride: 1

## V. Results

The outcomes are derived through cross-validation of chosen models, relying on test set representations extracted by both the MLP feature extractor and the ResNet extractor. In Figure 8, 9, 10, 11, 12, 13 and 14, the blue bars indicate the utilization of representations extracted by the ResNet feature extractor, while the green bars denote the use of representations extracted by the MLP feature extractor.

We observed marginal performance disparities in non-linear classifiers and tree classifiers (Fig 8, 9, 10), such as MLP, Random Forest, and Decision Tree. Similarly, slight differences were noted in non-learning-based models like KNN (Fig 13, 14). However, both feature extractors exhibited diminished performance in linear classifiers (Fig 11, 12), with all metric values decreasing by 0.1 to 0.6. Notably, the performance of the MLP-based extractor was significantly inferior to the ResNet-based one, with differences ranging from 0.1 to 0.2, whereas other classifiers showed a maximum performance difference of 0.05.

Upon examining training and inference metrics, see Table I, it was revealed that the training time for the MLP extractor was 42 percent less than that of the ResNet extractor, while the inference time was six times faster. Furthermore, the MLP extractor demanded less computing power due to its reduced parameter set, enhancing its compatibility with a broader array of devices.

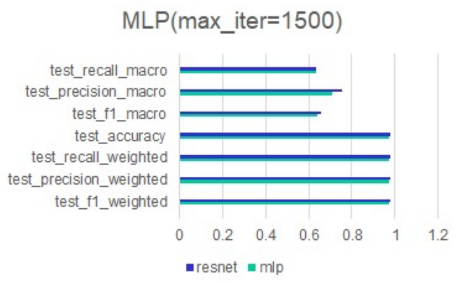

Fig. 8. Performance on MLP

## VI. Discussion

In general, our approach successfully attains the intended objective of replacing the SimCLR framework's backbone

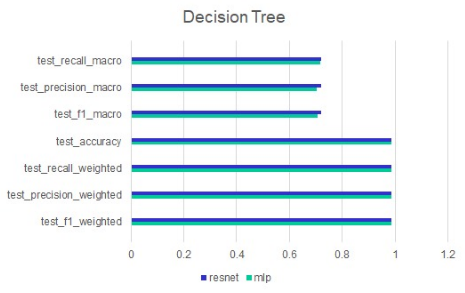

Fig. 9. Performance on Decision trees

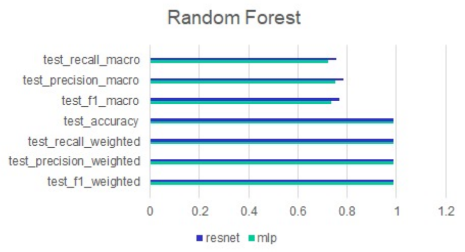

Fig. 10. Performance on Random forest

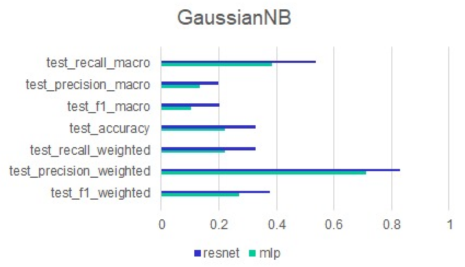

Fig. 11. Performance on GaussianNB

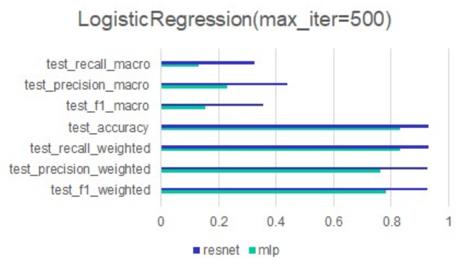

Fig. 12. Performance on Logistic Regression

network with a simpler one, aiming to achieve comparable or superior performance. This success is evident when combining non-linear classifiers, tree classifiers, or not-learning-based classifiers. However, the method falls short when applied to linear classifiers. The key advantage lies in our method's ability to deliver similar performance to the ResNet-based approach with significantly fewer parameters in non-linear classifiers, tree classifiers, and not-learning-based classifiers, with negligible performance differences. This makes our method

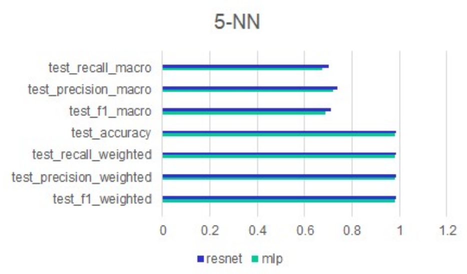

Fig. 13. Performance on 5-NN

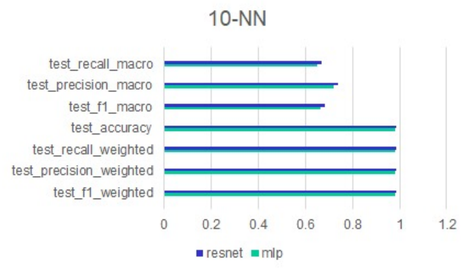

Fig. 14. Performance on 10-NN

suitable for deployment on resource-constrained devices with limited computational capabilities.

Nonetheless, a limitation arises concerning the method's poor performance on linear classifiers, rendering it unsuitable for such applications. While both the MLP-based method and the ResNet-based method exhibit suboptimal performance on linear classifiers, the MLP-based method performs significantly worse. Consequently, we advise against combining our method with any linear classifiers.

We hypothesize that the substantial performance gap between the MLP extractor and the ResNet-based one on linear classifiers may stem from the limited capacity of the MLP extractor, which consists of only two hidden layers. This limited architecture may hinder its ability to unfold data into lower dimensions, unlike the ResNet extractor, which boasts a sophisticated structure and a larger parameter space for training. While this assumption holds, it is essential to note that all classifiers used are default implementations from sklearn, and no hyperparameter tuning was conducted on them. The tuning efforts were solely directed at the feature extractors and projectors within the SimCLR framework. Therefore, the observed performance differences might be further influenced by hyperparameter tuning on the classifiers.

During hyperparameter tuning, we observed performance disparities between the ResNet extractor and the MLP extractor. Unexpectedly, the performance of the MLP extractor degraded when more than two additional hidden layers were introduced.. The optimal performance for the MLP extractor was achieved with two hidden layers with less hidden neurons or one hidden layer with more neurons. This result might be attributed to the absence of robust regularization in the MLP extractor, where only L2 regularization is applied, while the ResNet extractor benefits from BatchNorm and dropout layers, providing stronger regularization for improved training and performance.

TABLE I
COMPARISON OF COMPUTING RESOURCES CONSUMPTION

| Metrics | Feature extractor backbone | |
|---|---|---|
| | **mlp** | **resnet** |
| training time | ∼3.59hr | ∼6.24hr |
| inference time | ∼2.36s | ∼14.00s |
| params count | 8231 | 5338790 |
| mem consumption (f32) | ∼32K | ∼20MB |
| mem consumption (f16) | ∼16K | ∼10MB |
| mem consumption (int8) | ∼8K | ∼5MB |

## VII. Conclusion

Our study focuses on the novel exploration of utilizing a simpler backbone network within the SimCLR framework for intrusion detection systems (IDS). The primary goal was to ensure whether a less complex network like Multi-Layer Perceptrons (MLPs) could effectively replace the traditionally used, more complicated networks without compromising performance. The findings revealed that while this approach achieves comparable results with non-linear and non-learning methods, it slightly lags in performance when linear learning methods are applied. A key takeaway from this research is the significant reduction in computational resources required by the simpler extractor. This aspect is particularly relevant in scenarios where resource constraints are a critical consideration. Furthermore, the study's approach to handling imbalanced datasets through representation learning and using SimCLR with a modified network structure offers a fresh perspective. This method's potential to effectively deal with class imbalance in intrusion detection is a promising avenue for future developments. To conclude, this approach is particularly valuable in the current cybersecurity landscape, where there is a pressing need for efficient yet effective solutions. Our research aligns with the broader trend in machine learning and cybersecurity towards developing models that optimize computational resources while maintaining high accuracy and reliability.

## VIII. Future work

The promising results of using a simpler backbone network for intrusion detection suggest several directions for future research. Firstly, investigating a broader range of network architectures and datasets could provide deeper insights and validate the effectiveness of simpler networks in various contexts. Investigating how different network configurations affect performance could lead to more robust and efficient intrusion detection systems. This will uncover new insights into the scalability and adaptability of simpler networks in diverse environments. It's exciting because it pushes the boundaries of traditional network design, potentially leading to more efficient and versatile intrusion detection systems. Additionally, there is potential to refine and innovate data augmentation methods specific to intrusion detection. Refining data augmentation methods could significantly boost the performance of the simpler backbone network. These developments would contribute to the efficiency and effectiveness of intrusion detection systems and broaden the understanding of the applications and limitations of simpler neural network architectures in various machine learning domains.

## References

[1] Shwartz-Ziv, Goldblum, Li, Bruss, and Wilson, "On Representation Learning Under Class Imbalance," Sep. 2022.

[2] Jiang, Chen, Mortazavi, and Wang, "Self-Damaging Contrastive Learning," Jun. 2021, arXiv:2106.02990 [cs].

[3] Wang, Jian, Tan, Wu, and Huang, "Representation learning-based network intrusion detection system by capturing explicit and implicit feature interactions," *Computers & Security*, vol. 112, p. 102537, Jan. 2022.

[4] Li, Li, Zhang, Yang, Hu, and He, "Intrusion Detection for Industrial Control Systems Based on Improved Contrastive Learning SimCLR," *Applied Sciences*, vol. 13, no. 16, p. 9227, Jan. 2023, number: 16 Publisher: Multidisciplinary Digital Publishing Institute.

[5] Khraisat, Gondal, Vamplew, and Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, Jul. 2019.

[6] Kabir, Hu, Wang, and Zhuo, "A novel statistical technique for intrusion detection systems," *Future Generation Computer Systems*, vol. 79, pp. 303–318, Feb. 2018.

[7] Julisch and Dacier, "Mining intrusion detection alarms for actionable knowledge," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '02. New York, NY, USA: Association for Computing Machinery, Jul. 2002, pp. 366–375.

[8] Sommer and Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in *2010 IEEE Symposium on Security and Privacy*, May 2010, pp. 305–316, iSSN: 2375-1207.

[9] Bengio, Courville, and Vincent, "Representation Learning: A Review and New Perspectives," Apr. 2014, arXiv:1206.5538 [cs] version: 3.

[10] Wang, Neil, and Fenton, "A Bayesian network approach for cybersecurity risk assessment implementing and extending the FAIR model," *Computers & Security*, vol. 89, p. 101659, Feb. 2020.

[11] Aldwairi, Perera, and Novotny, "An evaluation of the performance of Restricted Boltzmann Machines as a model for anomaly network intrusion detection," *Computer Networks*, vol. 144, pp. 111–119, Oct. 2018.

[12] Gouveia and Correia, "A Systematic Approach for the Application of Restricted Boltzmann Machines in Network Intrusion Detection," in *Advances in Computational Intelligence*, Rojas, Joya, and Catala, Eds. Cham: Springer International Publishing, 2017, vol. 10305, pp. 432–446, series Title: Lecture Notes in Computer Science.

[13] Hou, Ma, and Zhang, "A New Method for Intrusion Detection using Manifold Learning Algorithm," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 11, no. 12, pp. 7339–7343, Dec. 2013, number: 12.

[14] Wang, Chen, Hu, Lou, and Hou, "MANDA: On Adversarial Example Detection for Network Intrusion Detection System," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, May 2021, pp. 1–10, iSSN: 2641-9874.

[15] Song, Hyun, and Cheong, "Analysis of Autoencoders for Network Intrusion Detection," *Sensors*, vol. 21, no. 13, p. 4294, Jan. 2021, number: 13 Publisher: Multidisciplinary Digital Publishing Institute.

[16] Narayana Rao, Venkata Rao, and P.V.G.D., "A hybrid Intrusion Detection System based on Sparse autoencoder and Deep Neural Network," *Computer Communications*, vol. 180, pp. 77–88, Dec. 2021.

[17] Lopes, Zou, Abdulqadder, Ruambo, Yuan, and Jin, "Effective network intrusion detection via representation learning: A Denoising AutoEncoder approach," *Computer Communications*, vol. 194, pp. 55–65, Oct. 2022.

[18] "Contrastive Representation Learning | Lil'Log."

[19] Chopra, Hadsell, and LeCun, "Learning a Similarity Metric Discriminatively, with Application to Face Verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. San Diego, CA, USA: IEEE, 2005, pp. 539–546.

[20] Luo, Zhang, Wu, Xu, Guo, and Shang, "A Multi-Channel Contrastive Learning Network Based Intrusion Detection Method," *Electronics*, vol. 12, no. 4, p. 949, Jan. 2023, number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[21] Chen, Kornblith, Norouzi, and Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[22] He, Zhang, Ren, and Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[23] Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[24] Ripley, *Pattern recognition and neural networks*. Cambridge university press, 2007.

[25] Venables and Ripley, *Modern applied statistics with S-PLUS*. Springer Science & Business Media, 2013.

[26] Hagan, Demuth, and Beale, *Neural network design*. PWS Publishing Co., 1997.