A Study on Self-Supervised Learning Objectives for Intrusion Detection

David de la Rosa*[0009−0006−4763−1071], María José del Jesus[0000−0002−7891−3059], and Francisco Charte[0000−0002−3083−8942]

Department of Computer Science. University of Jaén, Jaén, Spain
{drrosa, mjjesus, fcharte}@ujaen.es

Abstract. Intrusion detection systems are a critical component of modern cybersecurity architectures. Traditionally, supervised learning has been used to train models to accurately classify traffic flows and detect intrusions. However, high-quality labeled data is scarce in this field. This study analyzes the applicability of self-supervised learning in intrusion detection. We pretrain an encoder model with two different self-supervised objectives: Contrastive Learning and Instance Reconstruction. Then, we train both classical and neural network models on top of the latent features provided by the encoder. Results on the well-known UNSW-NB15 dataset demonstrate that self-supervised pretraining significantly outperforms traditional supervised baselines, particularly in regimes with limited labels, where only 1% of labeled data is available. Concretely, a small neural network trained on the latent features with just 1% of the labels achieved superior performance than other models trained on 100% of the data. Furthermore, the learned latent representations were found to be transferable between models, thereby enhancing the performance of non-neural classifiers such as SVM and KNN. Although achieving high detection rates for extreme minority classes remains challenging, our findings suggest that self-supervised learning is a robust and efficient alternative for developing high-performance network intrusion detection systems in scenarios with limited labeled data. The source code of this project is available in Github[1].

Keywords: Self-supervised learning · Contrastive learning · Intrusion Detection

1 Introduction

In the context of computer security, an intrusion is defined as either unauthorized access to an information system, or activities that could damage it. An intrusion attack deviates from standard operational behavior with the aim of gathering information, exploiting software vulnerabilities, establishing persistent access for control purposes, propagating malware, or reducing the availability of technological infrastructure. With the rise of big data and the constant development of

* Corresponding author: drrosa@ujaen.es
[1] Source code: https://github.com/DaviddlR/Lightning_SCARF.git

systems that store and process the information of millions of people, detecting these malicious activities is paramount.

Automatic Intrusion Detection (AID) [13] refers to systems that automatically detect intrusion attempts as early as possible. There are two categories within the AID field: signature-based systems rely on pattern-matching techniques to detect known attacks, while anomaly-based systems identify significant deviations between observations and the normal state of a computer system.

With the rise of Artificial Intelligence [1], intelligent models have been trained over the years for the task of AID [8]. Historically, AID models were trained in a supervised manner. In supervised learning, the model is trained using labeled samples covering each possible intrusion category. However, this paradigm poses several challenges in the computer security domain. First is the annotation work required. Labeling millions of flow traffic samples requires human expertise and a considerable amount of time. Another problem related to this is the labeling precision. If the model is fed erroneous annotations, the whole learning process will be affected, resulting in misclassifications after training. Lastly, in a supervised scenario, the labels predefine what the model learns, meaning it cannot learn anything new outside of them.

Despite their limitations, supervised systems have proven to be able to provide robust solutions in several problems in the tabular modality. Although unsupervised or Self-Supervised Learning (SSL) [10] approaches have surpassed classical machine learning algorithms [2] in domains such as computer vision and natural language processing, tree-based algorithms remain as the preferred solution for many problems involving tabular data [17, 19]. However, an SSL pretraining may be beneficial in several scenarios [21]. It consists of creating encoders able to produce a meaningful and compact latent representation capturing the essential features of the data. These encoders are trained via SSL following different approaches and objectives, such as reconstructing the original input [12], aligning different views of the instances [14], or predictive approaches [3]. As they are trained using self-supervised strategies, the encoder does not require labeled data during training, thus avoiding the aforementioned issues.

Although SSL has provided clear benefits, there is a gap in our understanding of which training objectives are most effective at capturing modern intrusions. The aim of this work is to conduct an experimental study to compare the effectiveness of various SSL objectives applied to intrusion detection, analyzing the effectiveness of Instance Reconstruction (IR) methods and Contrastive Learning (CL) approaches. The remaining of this manuscript is structured as follow. In Section 2 we will analyze current proposals for intrusion detection in tabular data to detect research gaps. The SSL objectives will be described in Section 3, while the experimentation on the UNSW-NB15 dataset [15] will be described in Section 4. Lastly, we present the conclusions in Section 5.

2 Related work

In this section we discuss some existing supervised proposals for AID. Furthermore, we analyze the applications of SSL in the context of tabular data and review several SSL works for AID.

The work [8] presents experiments involving several well-known supervised models on the UNSW-NB15 dataset. The authors used four different classifier models: Naive Bayes, Random Forest, J48 and ZeroR. Moreover, two clustering algorithms, K-Means and Expectation Maximization, were used to cluster the dataset into two groups depending on whether the target attribute was *attack* or *normal* network traffic. They found that combining clustering algorithms with J48 produced the best results for binary classification in that specific dataset. Similarly, an experimentation with the Random Forest and J48 algorithms is presented in the article [11]. Authors tested both supervised models on the NSL-KDD dataset, showing that their proposal is efficient with low false alarm rate and high detection rate.

However, as discussed in the previous section, the time and effort required to create a labeled dataset, coupled with the limited knowledge provided by labels, has pushed researchers to experiment with self-supervised approaches. Many studies have analyzed the performance of SSL proposals with different training objectives. The first category is CL for the tabular domain, as examined in the studies [4–6], in which similar views of the same tabular sample are encoded together in the latent space while pushing apart dissimilar instances. The second approach is IR [23, 24]. In this methodology, a sample is first corrupted and encoded to create its latent representation, which is then reconstructed in the original data space to measure the reconstruction loss. Another example is SAINT [18], which combines CL and an additional objective in which the model must predict the transformation applied to a tabular sample.

Following these proposals, some authors have tested SSL in the field of AID. For instance, the study [20] introduce a masked context reconstruction module into a transformer-based SSL scheme for binary classification in the UNSW-NB15 dataset. In the article [22], the BYOL framework [9], which is part of the view-alignment SSL paradigm, was adapted to work on tabular data for AID in the same dataset, introducing a transfer learning phase to many other datasets. A similar experiment was conducted in [7], adapting various view-alignment methodologies originally designed for vision tasks to the tabular domain. Lastly, the study [25] analyzes a contrastive pretraining on the UNSW-NB15 dataset.

Despite the successful integration of SSL into several AID proposals, some research gaps remain that need to be addressed. Firstly, even though many models were trained on the UNSW-NB15 dataset, the authors simplified the data by removing the existing classes of the dataset and transforming it into a binary classification dataset. In this study, we will analyze how our models perform under a multiclass classification scheme, discussing these classes that they are not able to identify. Secondly, the authors created models based solely on neural networks. However, pretraining could improve the performance of classifiers trained on latent representations provided by the encoder. We will discuss the

classification accuracy of several classic models to determine whether they benefit from SSL. Finally, we will analyze how our models perform in a simulated low-label scenario where we do not have access to the entire labelled dataset.

3 Self-supervised learning objectives for intrusion detection

In this section we define and describe the SSL objectives that we have tested for intrusion detection. Given the broad scope of SSL and the large number of proposals and families of methodologies, we have selected two of them: reconstruction-based approaches and CL objectives. Reconstruction-based RL proposes a simple objective in which a model must reconstruct a corrupted or masked version of the input. In contrast, CL seeks to align positive and negative views of the same instances. Positive views represent a data point that share the same semantic meaning than the original input, while negative views are these that represent different concepts. Both methodologies have been widely explored and have proved to yield meaningful representations in the tabular domain. For these reasons, we hypothesize that they may be suitable for intrusion detection datasets.

3.1 Instance reconstruction objective

The IR objective follows a generative approach in which an encoder-decoder architecture is trained to recover the original input from a corrupted version. The hypothesis is that, to accurately reconstruct the data from a latent representation, the model must have learn how to produce a representation that captures the key structural dependencies of the dataset. This workflow is depicted in Figure 1.

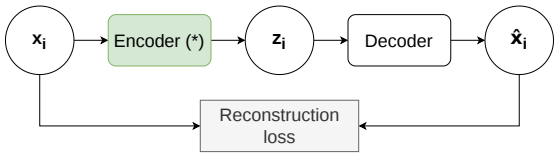


Fig. 1. Diagram representing the workflow of an IR model. x_i is the original instance, z_i is the latent representation and $\hat{x}_i$ is the reconstructed instance. Models highlighted with (*) are used after pretraining to obtain the representation of new samples.

As we are working in the tabular domain, the input space consists of a feature vector of as many attributes as the tabular instances have. Then, an MLP encoder maps this input to a compact latent representation, which is mapped back to the original data space by a decoder.

The next step is to define how the instance will be corrupted. Following the corruption procedure presented in the study [5], we apply the Swap Noise

procedure. It follows three main steps. First, the mask generation. A binary mask is generated for a given input. It highlights the features that will be corrupted according to a fixed corruption rate. Second, the feature sampling. To ensure that the corrupted values remain within the valid domain of each feature, we select values from the same feature columns but from different samples in the current batch. These samples values can be used to replace the original features according to the mask. Unlike simple Gaussian noise addition, this method ensures that the corrupted value is always a plausible value that actually exists in the data. Third, the reconstruction step. After the corruption process, the encoder-decoder architecture creates a latent representation and reconstruct it to the original data space. Lastly, the MSE loss, defined in equation (1), is computed between the original instance, x, and the reconstructed one, $\hat{x}$, serving as loss to optimize the models.

$$Loss = |x - \hat{x}|^2 \tag{1}$$

3.2 Contrastive learning objective

Unlike reconstruction-based approaches, the CL objective focuses on organizing and structuring the latent space by maximizing the agreement between different views of the same instance that share their semantic meanings while minimizing it for different instances. This framework typically uses a single encoder, thus we do not require a decoder. The workflow is depicted in the Figure 2.

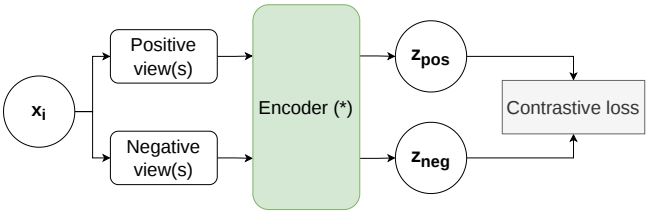


Fig. 2. Diagram representing the workflow of an CL model. x_i is the original instance while z_{pos} and z_{neg} are the latent representation of positive and negative views. Models highlighted with (*) are used after pretraining to obtain the representation of new samples.

A critical component in contrastive methodologies is a proper definition of positive and negative instances. We have to create positive views that, while being different from the original instance, share its semantic meaning. Following SCARF [4], we generate a second positive view by applying the same Swap Noise corruption procedure used in the reconstruction task. Regarding negative sampling, the most widely adopted procedure is to use the remaining instances of the current batch as negatives. Accordingly, for a given instance in a batch

of size N, we obtain a correlated view with its corrupted counterpart while all other $N-1$ samples within the batch are treated as negative views.

The contrastive objective demand that, at each step, a loss is computed so that the positive views are pulled together in the latent space while pushing them away from the remaining negative samples. With this objective, we employ the InfoNCE loss [16], defined as in equation (2), where (i, j) are the indexes of a positive pair, $sim()$ is the cosine similarity and τ is a temperature scaling parameter. This loss forces the encoder to extract invariant features that are robust to specific perturbations.

$$Loss = -log \frac{\frac{exp(sim(z_i, z_j))}{\tau}}{\sum_{k=1 \neq i}^{2N} \frac{exp(sim(z_i, z_k))}{\tau}} \tag{2}$$

4 Experimentation

4.1 Dataset and preprocessing

We test the proposed methodology for intrusion detection in the UNSW-NB15 dataset [15]. It was created to generate a hybrid of real normal activities and synthetic contemporary attack behaviors.

For our experiments, we use the partitioned version of this dataset, which is widely adopted in the literature to ensure reproducibility [22, 20]. This version comprises a training set of 175 341 samples and a test set of 82 332 samples. As is presented in Table 1, the dataset is characterized by a significant class imbalance. While the *Normal* and *Generic* categories represent the majority of the instances, other classes such as *Worms* or *Shellcode* are extremely scarce. This imbalance poses a key challenge for AI models, especially when the number of annotated instances is low. Our SSL proposals will be evaluated under the multiclass classification protocol, this is, our model must assign each of the test instances to one of the ten possible classes.

Category	Samples	Percentage	Category	Samples	Percentage
Normal	56 000	31.94%	**Reconnaissance**	10 491	5.98%
Generic	40 000	22.81%	**Analysis**	2 000	1.14%
Exploits	33 393	19.04%	**Backdoor**	1 746	1.00%
Fuzzers	18 184	10.37%	**Shellcode**	1 133	0.65%
DoS	12 264	7.00%	**Worms**	130	0.07%

Table 1. Representation of the classes of the UNSW-NB15 dataset and the number of instances per class.

Each instance in the dataset consists of 42 features. Notable examples include *sbytes* and *dbytes*, which indicate the source/destination-to-destination/source transaction bytes, *sttl* and *dttl*, which indicate the source/destination time-to-live, or *service*, which indicates the protocol used.

This dataset is heterogeneous, meaning that we have both numerical features (such as *sbytes*), and categorical features such as *service* or *state*. For that reason, a preprocessing step is necessary. Numerical features often exhibit high variance, so they are processed using Z-score normalization, which centers the distribution around a mean of zero and unit standard deviation. Categorical features are transformed using One-Hot encoding. This converts each category into a sparse binary vector, preventing the model from assuming an artificial ordinal relationship. Crucially, all transformation parameters are fitted exclusively on the training partition and then applied to the validation and test sets, preventing data leakage.

4.2 Methodology

After the data preprocessing steps, we can describe the experimental methodology of this work. The main objective is to determine whether SSL improves the classification results on intrusion detection tasks, specifically on the UNSW-NB15 dataset.

First, to prevent data leakage and to guarantee a correct evaluation of the models, we define three different partitions: train, test and validation sets. The training set is used to train the models. The validation set serves to obtain the optimal values of two key hyperparameters required for training: batch size and number of epochs. Lastly, the test set, not seen during training, is employed to verify the usability and generalization capabilities of the models.

To address the proposed objectives, we defined four different settings in which our models will be trained. The first is denominated *Supervised (100%)*. In this setting, models are trained in a supervised manner using the whole labeled dataset. The second is denominated *Supervised (1%)*. As labeled data may be scarce, we simulate this situation by only using 1% of the annotated data to train the models. This subset is sampled in a stratified manner to preserve class proportions. Third and four settings rely on SSL following the training objectives described in Section 3: a CL objective and an IR objective. Both of them follow the same training procedure. During pretraining, as they do not require labeled data, in both scenarios we fed the encoder with the complete training dataset (excluding the class column feature). After pretraining is complete, we use 1% of the labeled data to train a classification head on top of the features produced by the encoder.

As mentioned in Section 1, classical algorithms may surpass performance results of neural networks in the tabular domain. For that reason, we selected five different models apart from neural networks: K-Nearest Neighbors, Support Vector Machines, C4.5, Random Forest and XGB. In the first two settings these algorithms are trained directly on the preprocessed features, while in the last two settings they are trained on top of the features provided by the encoder, so we can verify if these models benefit from a self-supervised pretraining.

To further validate the results and ensure that our methodology is not biased toward randomness initialization, we defined a training and evaluation protocol

in which models are trained and evaluated thirty times, using a different initialization seed in each run. For each evaluation, we store the performance metrics so that we can easily compute 95% confidence intervals for each metric.

4.3 Experimentation setup

Before evaluating the models, we have to define several hyperparameters for the MLP model. As we are using the same SSL objective as in [4], we use the same hyperparameters for most of our experiments.

The architecture of the encoder is an MLP with four hidden layers of 256 neurons each. During pretraining, a projection head is used to further enhance the representation capabilities. The head has two hidden layers of 256 neurons each. This head is discarded after pretraining. We use a batch size of 256, higher than usual to benefit the contrastive objective. Lastly, the MLP classifier model is just one single layer of 256 dimensions followed by a linear layer to map these dimensions to the number of classes. For classical models, we use the default hyperparameters, only setting the number of estimators of Random Forest and XGB to 100.

Regarding the number of epochs, we used the validation dataset to analyze the training step in which it start to appear the overfitting issue. Figures 3 and 4 represent the training and validation loss for both classifier and encoder for 300 epochs. As can be seen, for the encoder, from 150 epochs onwards the validation loss start to increase while the training loss continues decreasing, being this a clear sign of overfitting. The same situation appears on the classifier MLP at 50 epochs. For these reasons, we set the number of epochs during pretraining to 150 and for the classification phase at 50.

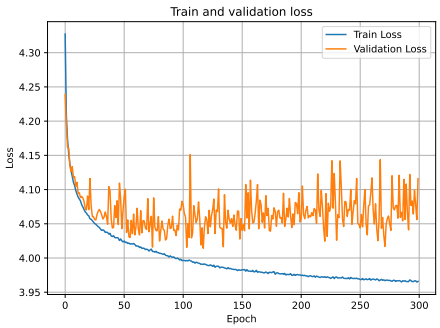


Fig. 3. Train and evaluation loss of the encoder during pretraining.

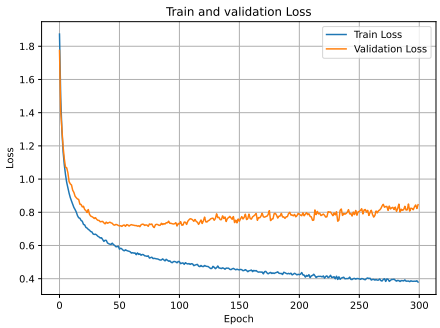


Fig. 4. Train and evaluation loss of the classifier model after pretraining.

4.4 Experimentation results

The results of our experiments are summarized in Table 2. We compare the performance of standard supervised learning on the whole labeled training dataset and on a scenario of scarce labeled data against the two SSL approaches presented in section 3.

Model	Metric	Sup.(100%)	Sup.(1%)	CL	IR
MLP	ACC	(0.686 - 0.707)	(0.663 - 0.689)	**(0.708 - 0.766)**	(0.688 - 0.715)
MLP	Macro F1	(0.334 - 0.363)	(0.311 - 0.330)	**(0.353 - 0.379)**	(0.323 - 0.366)
MLP	Weighted F1	(0.700 - 0.716)	(0.680 - 0.697)	**(0.722 - 0.756)**	(0.704 - 0.728)
KNN	ACC	(0.702 - 0.706)	(0.655 - 0.670)	**(0.700 - 0.727)**	(0.678 - 0.708)
KNN	Macro F1	**(0.395 - 0.403)**	(0.328 - 0.348)	(0.365 - 0.391)	(0.350 - 0.370)
KNN	Weighted F1	(0.727 - 0.729)	(0.679 - 0.690)	**(0.722 - 0.745)**	(0.704 - 0.728)
SVM	ACC	(0.681 - 0.681)	(0.646 - 0.674)	**(0.664 - 0.723)**	(0.628 - 0.719)
SVM	Macro F1	**(0.326 - 0.326)**	(0.288 - 0.296)	(0.314 - 0.323)	(0.298 - 0.314)
SVM	Weighted F1	(0.693 - 0.693)	(0.656 - 0.672)	**(0.679 - 0.716)**	(0.646 - 0.701)
C4.5	ACC	**(0.746 - 0.749)**	(0.685 - 0.728)	(0.660 - 0.696)	(0.657 - 0.685)
C4.5	Macro F1	**(0.470 - 0.488)**	(0.370 - 0.416)	(0.343 - 0.380)	(0.344 - 0.368)
C4.5	Weighted F1	**(0.764 - 0.767)**	(0.715 - 0.746)	(0.687 - 0.716)	(0.687 - 0.705)
RF	ACC	**(0.767 - 0.770)**	(0.729 - 0.745)	(0.719 - 0.744)	(0.700 - 0.726)
RF	Macro F1	**(0.450 - 0.454)**	(0.385 - 0.411)	(0.386 - 0.403)	(0.368 - 0.386)
RF	Weighted F1	**(0.777 - 0.779)**	(0.742 - 0.755)	(0.735 - 0.754)	(0.721 - 0.737)
XGB	ACC	**(0.777 - 0.780)**	(0.732 - 0.756)	(0.720 - 0.742)	(0.702 - 0.727)
XGB	Macro F1	**(0.479 - 0.492)**	(0.396 - 0.420)	(0.388 - 0.406)	(0.368 - 0.396)
XGB	Weighted F1	**(0.776 - 0.779)**	(0.742 - 0.763)	(0.737 - 0.753)	(0.723 - 0.739)

Table 2. Table presenting the accuracy (ACC), Macro F1-score and Weighted F1-score of five different models on four different settings: MLP (Multi-layer Perceptron), KNN (K-Nearest Neighbors), SVM (Support Vector Machine), C4.5, RF (Random Forest) and XGB (Extreme Gradient Boosting) on the settings Sup.(100%) (supervised trained with 100% of the data), Sup.(1%) (supervised trained with 1% of the data), CL and IR (Contrastive Learning and Instance Reconstruction pretraining). The best results are highlighted in bold, based on the mean value of the confidence interval.

The most significant finding is observed in the MLP model. While the supervised baseline with 100% of the labels achieves an ACC range of [0.686, 0.707], the CL pretraining, evaluated with only 1% of the labels, achieves a significantly higher ACC of [0.708, 0.766]. This fact suggests that the latent representations learned during the self-supervised phase are more robust and discriminative than those learned through standard supervised learning on raw features. This is reaffirmed when analyzing the results achieved by other models such as KNN or SVM, both improving the results compared with the supervised 1% setting. These results confirm that SSL is able to improve the performance results on situations in which the labeled data is scarce. However, for some models as C4.5, RF or XGB, neither CL or IR improved the results over the supervised approaches, highlighting that, in the tabular domain, some classical methods still outperforms neural networks and do not benefit from a SSL pretraining.

Regarding the comparison between both CL and IR, it is clear how CL consistently outperforms IR for every model that we tested. This disparity suggests that the discriminative objective of CL, which forces different instances apart in the latent space, is more effective for intrusion detection than the simple reconstruction task. IR focuses on perfect recovery, which might waste the capacity of the model on noise, whereas CL works in the latent space thus avoiding fine-grained details that are not needed for classification.

A critical challenge in the UNSW-NB15 dataset is the severe class imbalance, evidenced in the Table 2 by the large difference between Macro F1 and Weighted F1 score. Figure 5 provides details about the classification in each class. Class *Generic* shows near-perfect performance across all settings, while class *Normal*, representing the standard traffic flow, also achieves a high performance. The most striking result is found in classes *Normal* and *Reconnaissance*. In these categories, CL outperforms even the supervised 100% baseline. For classes *Exploits* and *Fuzzers*, while the supervised 100% model is slightly better than the remaining models, both SSL methods outperform the supervised 1% baseline, confirming that SSL successfully recovers a large portion of the performance lost when labels are scarce. Despite the overall success, classes *Analysis, Backdoor, Shellcode* and *Worms* remain a challenge for all configurations. The near-zero F1 scores in these classes highlight the difficulty of detecting attacks that have extremely few samples both with and without pretraining, suggesting that this situation still requires specializes techniques to address class imbalance issues.

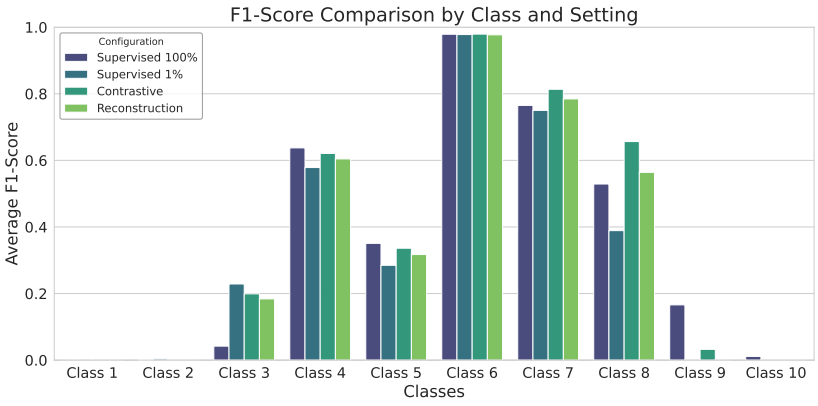


Fig. 5. Diagram representing the F1 score of each setting per class.

We have also measured the training and inference time of all the models, presented in Table 3. While the SSL framework introduces a substantial upfront cost during the pretraining phase, it drastically accelerates the downstream classifier training compared to training standard models from scratch. This efficiency gain is most pronounced in the SVM, where standard training requires 6 064 seconds

Type	Pretraining	Feature extr.	Training	Classification	Inference
SSL+MLP	726.52	1.39	26.69	1.25	2.64
SSL+KNN	726.52	1.39	0.0004	1.08	2.47
SSL+SVM	726.52	1.39	0.34	6.67	8.06
SSL+C4.5	726.52	1.39	0.21	0.01	1.40
SSL+RF	726.52	1.39	0.87	0.55	1.94
SSL+XGB	726.52	1.39	2.71	0.16	1.55
MLP	-	-	499.49	1.21	1.21
KNN	-	-	0.02	12.60	12.60
SVM	-	-	6 064.67	454.53	454.53
C4.5	-	-	2.64	0.01	0.01
RF	-	-	18.38	0.97	0.97
XGB	-	-	7.81	0.14	0.14

Table 3. Training and inference time, measured in seconds, of the different models used in this study. We measure the SSL pretraining time, feature extraction process time, training time of the classification model and the classification time. The last column measures inference time after training the models, this is, the sum of *Feature extr.* and *Classification* columns.

whereas SSL+SVM reduces this bottleneck to just 0.34 seconds. For inference, the models generally benefit from the low-dimensional embeddings provided by SSL, yielding faster downstream classification times.

5 Conclusions

In this study we have covered the impact of two SSL objectives, CL and IR, along with five different AI models, in the context of intrusion detection on the UNSW-NB15 dataset. Our results prove that SSL, particularly CL, provides a significant advantage, especially in low-label regimes. Notably, for the MLP architecture, the CL pre-training using only 1% of the available labels outperformed the fully supervised model trained on 100% of the data. It confirms that the corruption strategy along with the InfoNCE loss allow the model to learn a more discriminative and robust latent representation of the network traffic. Moreover, we proved that one could use these latent representations to train different ML architectures, such as SVM or KNN, improving the baseline results. The applications of the proposed methods are clear in the intrusion detection domain. In scenarios in which we do not have much labeled data, a CL pretraining can equal, or even surpass, fully-supervised models, thus minimizing the effort needed to train models in these scenarios.

Despite the benefits of SSL, this study identified certain limitations. First is the performance on extreme class imbalance. While SSL improved the performance on some minority classes, it was less effective for extremely rare categories such as *Worms* or *Backdoor*. Second is the tree-based dominance. In scenarios where labels are abundant, tree-based methods and XGB still outperformed SSL-based neural networks on raw tabular data, thus not fully bridging the gap between neural architectures and classical ML methods on this data modality.

This analysis can be extended by introducing several new approaches. For instance, hybrid SSL objectives involving a multi-task objective during pretraining

may help to capture both local features and global semantics of the instances. Moreover, to address the issue of rare attacks, several oversampling methodologies may be applied into the fine-tuning phase to improve the detection rates. Lastly, testing the approaches on other intrusion detection datasets may verify the correct generalization on SSL for cybersecurity.

Acknowledgments

The research carried out in this study is part of the project "Advances in the development of trustworthy AI models to contribute to the adoption and use of responsible AI in healthcare (TAIH)" financed by the Spanish Ministry of Science, Innovation and Universities with code PID2023-149511OB-I00.

References

1. AI, H.: A definition of AI: Main capabilities and scientific disciplines. Ethics guidelines for trustworthy AI pp. 1–6 (2019)
2. Alloghani, M., Al-Jumeily, D., Mustafina, J., Hussain, A., Aljaaf, A.J.: A systematic review on supervised and unsupervised machine learning algorithms for data science. Supervised and Unsupervised Learning for Data Science pp. 3–21 (2020)
3. Assran, M., Duval, Q., Misra, I., Bojanowski, P., Vincent, P., Rabbat, M., LeCun, Y., Ballas, N.: Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture. In: 2023 Computer Vision and Pattern Recognition (CVPR). pp. 15619–15629. IEEE, Vancouver, Canada (2023)
4. Bahri, D., Jiang, H., Tay, Y., Metzler, D.: Scarf: Self-Supervised Contrastive Learning using Random Feature Corruption. In: 10th International Conference on Learning Representations (ICLR). ICLR, Online (2022)
5. Chen, S., Wu, J., Hovakimyan, N., Yao, H.: ReConTab: Regularized Contrastive Representation Learning for Tabular Data. In: 37th Neural Information Processing Systems (NeurIPS). Curran Associates Inc., New Orleans, Louisiana, United States (2023)
6. Chen, S., Wu, J., Wang, Y., Ji, C., Xie, T., Cociorva, D., Sharps, M., Levasseur, C., Brunzell, H.: TabDeco: A Comprehensive Contrastive Framework for Decoupled Representations in Tabular Data. In: 38th Neural Information Processing Systems (NeurIPS). Curran Associates, Inc., Vancouver, Canada (December 10-15 2024)
7. Fard, H., Schalau, T., Wunder, G.: An investigation into the performance of non-contrastive self-supervised learning methods for network intrusion detection. In: Information and Communications Security. pp. 208–227. Springer Nature Singapore, Singapore (2025)
8. Farnaaz, N., Jabbar, M.: Random forest modeling for network intrusion detection system. Procedia Computer Science **89**, 213–217 (2016)
9. 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. In: 34th Neural Information Processing Systems (NeurIPS). pp. 21271 – 21284. Curran Associates Inc., Vancouver, BC, Canada (2020)

10. Gui, J., Chen, T., Zhang, J., Cao, Q., Sun, Z., Luo, H., Tao, D.: A Survey on Self-Supervised Learning: Algorithms, Applications, and Future Trends. IEEE Transactions on Pattern Analysis and Machine Intelligence **46**(12), 9052–9071 (2024)
11. Hammad, M., El-medany, W., Ismail, Y.: Intrusion detection system using feature selection with clustering and classification machine learning algorithms on the unsw-nb15 dataset. In: 2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT). pp. 1–6 (2020)
12. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked Autoencoders Are Scalable Vision Learners. In: 2022 Computer Vision and Pattern Recognition (CVPR). pp. 15979–15988. IEEE, New Orleans, LA, USA (2022)
13. Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J.: Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity **2**(1), 1–22 (2019)
14. Le-Khac, P.H., Healy, G., Smeaton, A.F.: Contrastive Representation Learning: A Framework and Review. IEEE Access **8**, 193907–193934 (2020)
15. 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 (2015)
16. Oord, A.v.d., Li, Y., Vinyals, O.: Representation Learning with Contrastive Predictive Coding. arXiv:1807.03748 (2018)
17. Shwartz-Ziv, R., Armon, A.: Tabular data: Deep learning is not all you need. Information Fusion **81**, 84–90 (2022)
18. 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. arXiv:2106.01342 (2021)
19. Van Breugel, B., Van Der Schaar, M.: Why tabular foundation models should be a research priority. arXiv preprint arXiv:2405.01147 (2024)
20. Wang, W., Jian, S., Tan, Y., Wu, Q., Huang, C.: Robust unsupervised network intrusion detection with self-supervised masked context reconstruction. Computers & Security **128**, 103131 (2023)
21. Wang, W.Y., Du, W.W., Xu, D., Wang, W., Peng, W.C.: A survey on self-supervised learning for non-sequential tabular data. Machine Learning **114**(16) (2025)
22. 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), 9486949 (2021)
23. Wu, J., Chen, S., Zhao, Q., Sergazinov, R., Li, C., Liu, S., Zhao, C., Xie, T., Guo, H., Ji, C., Cociorva, D., Brunzell, H.: SwitchTab: Switched Autoencoders Are Effective Tabular Learners. In: 38th AAAI Conference on Artificial Intelligence (AAAI-24). pp. 15924–15933. AAAI, Vancouver, Canada (February 20-27 2024)
24. Yoon, J., Zhang, Y., Jordon, J., van der Schaar, M.: VIME: Extending the Success of Self- and Semi-supervised Learning to Tabular Domain. In: 34th Neural Information Processing Systems (NeurIPS). pp. 11033–11043. Curran Associates, Inc., Online (December 06-12 2020)
25. Yue, Y., Chen, X., Han, Z., Zeng, X., Zhu, Y.: Contrastive learning enhanced intrusion detection. IEEE Transactions on Network and Service Management **19**(4), 4232–4247 (2022)