Local Contrastive Feature Learning for Tabular Data

Zhabiz Gharibshah
Dept. of EECS, Florida Atlantic University
Boca Raton, FL 33431, USA
zgharibshah2017@fau.edu

Xingquan Zhu
Dept. of EECS, Florida Atlantic University
Boca Raton, FL 33431, USA
xzhu3@fau.edu

ABSTRACT

Contrastive self-supervised learning has been successfully used in many domains, such as images, texts, graphs, etc., to learn features without requiring label information. In this paper, we propose a new local contrastive feature learning (LoCL) framework, and our theme is to learn local patterns/features from tabular data. In order to create a niche for local learning, we use feature correlations to create a maximum-spanning tree, and break the tree into feature subsets, with strongly correlated features being assigned next to each other. Convolutional learning of the features is used to learn latent feature space, regulated by contrastive and reconstruction losses. Experiments on public tabular datasets show the effectiveness of the proposed method versus state-of-the-art baseline methods.

CCS CONCEPTS

• **Information systems** → **Data mining**.

KEYWORDS

Contrastive learning, self-supervised learning, tabular data

ACM Reference Format:
Zhabiz Gharibshah and Xingquan Zhu. 2022. Local Contrastive Feature Learning for Tabular Data. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557630

1 INTRODUCTION

Tabular data, using rows (instances) and columns (features) to represent objects, are ubiquitous in nearly all applications [3, 7, 17]. Feature engineering is a traditional method to analyze the data and produce informative features for predictive modeling. Recently, self-supervised learning combined with deep learning methods to learn feature representations from unlabeled data has shown considerable success in different domains, especially for images, graphs and texts [4, 19, 22–24]. Some studies have conducted to extend this success to tabular data where data samples are represented by vectors with different value types [18, 21]. In practice, decision tree-based models like XGBoost are still known as strong non-gradient based models with a comparable or even superior performance [10, 15]. However, some advantages with deep learning methods like attention mechanism, pre-training parameters and providing an end-to-end data processing paradigm for training make them appealing for learning efficient feature representations [1, 16].

Lack of clear feature relationships in tabular data, fully connected dense neural networks are typically used as a parametric method for training to consider the impact of all features on the target values in supervised setting [8, 12]. Some methods have been proposed to enable deep feature learning in contrastive learning paradigm for tabular data, however, they all use dense layer network [2, 5, 14, 21]. The main drawback of dense layer is that they learn global patterns using all features. In many datasets (or learning tasks), patterns only involve a small number of features (not all features are useful). On the other hand, in real-world datasets, features are often subject to some correlations, which naturally results in local interactions [11]. That motivates us to explore local pattern learning for tabular data. Here local learning is referring to that only a few number of features are involved in the pattern learning via the convolutional neural network (CNN) kernels. CNN networks have known as effective network design with parameter sharing to reduce model complexity to capture spatial connections between neighboring features with contiguous values. To address this problem in tabular data, we will create a niche of feature correlation by exploring a meaningful reordering of input features to apply convolutional kernels.

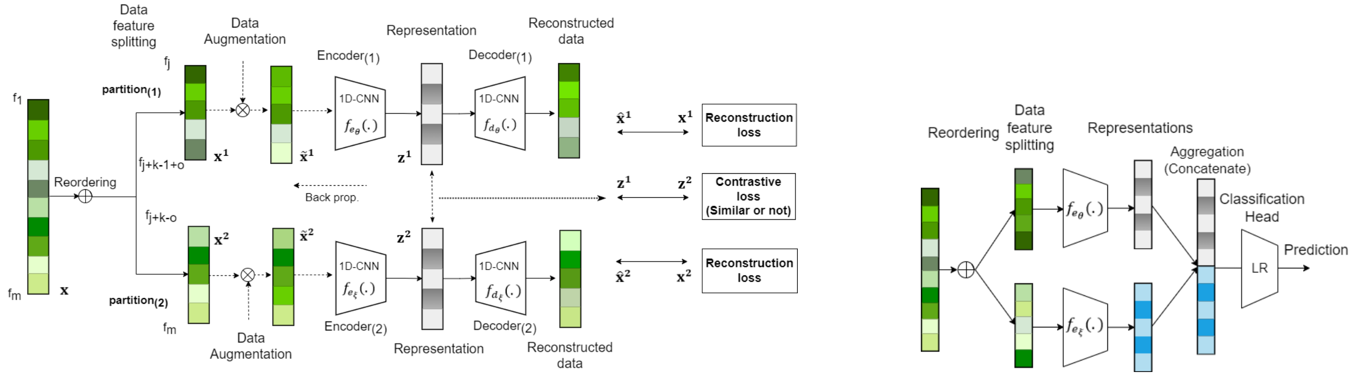
To leverage convolutional feature learning, we develop a novel algorithm to use pairwise Pearson correlation coefficients between features as the metric to create a maximum spanning tree to connect all features followed by a depth-first-search traverse. It generates the new order of features being spatially correlated. In addition, we convert the definition of feature learning from a holistic format containing the whole feature values to multiple subsets of features created by feature splitting. We propose a self-supervised learning framework which leverages a 1-D convolutional denoising autoencoder [13] as a building block to capture correlations within a subset of the reordered input features and to maximize mutual information using contrastive comparisons between pair of subset embedding vectors. We hypothesize that a deep neural network with convolutional operation on a local set of features, combined with contrastive and reconstruction optimizations in a self-supervised manner, can provide effective performance in classification downstream tasks.

2 PROBLEM DEFINITION

A tabular dataset $\mathcal{D} = \{\mathbf{x}_i\}_1^N$ consists of N instances and m features as m-dimensional vector $\mathbf{x}_i \subset \mathcal{X} \in \mathbb{R}^m$ among which a small subset of data samples is labeled, i.e. $\mathcal{D}_L = \{\mathbf{x}_i, y_i\}_1^N$ where $\mathcal{D}_L \subset \mathcal{D}$, $|\mathcal{D}| \gg |\mathcal{D}_L|$ and $y_i \subset \mathcal{Y} \in \mathbb{R}$ is a discrete label set containing two or more categorical values. In a supervised setting, learning a predictive model $f : \mathcal{X} \to \mathcal{Y}$ is optimized by using a supervised loss function (e.g. cross-entropy loss function). But when a small set

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557630



(a) **(b)**

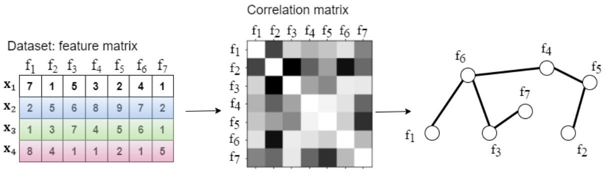
Figure 1: a)(Unsupervised pre-training): workflow of the proposed method for self-supervised local contrastive learning: From left to right, the m features of a tabular detest are partitioned into two subsets (with or without overlapping). A convolutional auto-encoder is trained from each subset, respectively. A contrastive loss is added to ensure latent features learned from feature partitions of the same instance are close to each other and to be distant if they are from two different instances; b)(Supervised learning) Representation at the time for a prediction task consisting of training a linear classifier on top of frozen representations.

Algorithm 1 LoCL: Local Contrastive Feature Learning

```
Inputs: Augmentation $\mathcal{T}$, encoder $f_{e_\theta}$, encoder $f_{e_\xi}$, decoder $f_{d_\theta}$, decoder $f_{d_\xi}$,
Batch size: $n$, encoder layer size: $d$, input feature indices: $\mathbb{F}$=[$f_1, f_2, ..., f_m$]
                                                          ▷ Reordering input features
$\mathcal{M} \in \mathbb{R}^{m \times m} \longleftarrow$ Pearson($\mathbf{X}, \mathbb{F}$) ▷ Calculate a pairwise feature correlation matrix
MST $\longleftarrow$ Maximum Spanning Tree($\mathcal{M}$)   ▷ Create a feature correlation
    maximum spanning tree considering input features $f_1, f_2, ..., f_m$ as nodes and top
    $m - 1$ pearson correlations as the edges.
$\bar{\mathbb{F}} \longleftarrow$ DFS($MST$)   ▷ Generate feature orders using the order of visited nodes by
    using a DFS traverse starting from a feature with the highest pairwise correlation.
[$\mathbb{F}^1, \mathbb{F}^2$] $\longleftarrow$ split($\bar{\mathbb{F}}$)   ▷ Divide the features into 2 subsets split uniformly from $\bar{\mathbb{F}}$
for sampled batch $\mathcal{B} : \{\mathbf{X} | \{\mathbf{x}\}_{k=1}^{n}\}$ do
    for all $k$=1 to n do
                                        ▷ Apply augmentations and get network outputs
        $\mathbf{x}^1, \mathbf{x}^2 = \mathbf{x}[k, \mathbb{F}^1], \mathbf{x}[k, \mathbb{F}^2]$
        $\mathbf{z}^1, \mathbf{z}^2 = f_{e_1}(\mathcal{T}_1(\mathbf{x}^1)), f_{e_2}(\mathcal{T}_2(\mathbf{x}^2))$
        $\hat{\mathbf{x}}^1, \hat{\mathbf{x}}^2 = f_{d_1}(\mathbf{z}^1), f_{d_2}(\mathbf{z}^2)$
                                        ▷ reconstruction and contrastive loss
        $\mathcal{L}_{r_1}[k], \mathcal{L}_{r_2}[k] = \|\hat{\mathbf{x}}^2 - \mathbf{x}^2\|_2^2, \|\hat{\mathbf{x}}^1 - \mathbf{x}^1\|_2^2$
        $\mathcal{L}_r[k] = \frac{1}{2} \sum_i^2 (\mathcal{L}_{r_i}[k])$
        $\mathcal{L}_c[k] = l_c(\mathbf{z}^j, \mathbf{z}^l)[k]$
        $\mathcal{L}[k] = \mathcal{L}_c[k] + \alpha \ \mathcal{L}_r[k]$
    end for
    $\nabla_{\theta,\xi} \mathcal{L}$           ▷ Calculate gradients and update all trainable parameters
end for
Output: encoder $f_{e_\theta}$,$f_{e_\xi}$
```

of labeled data is available it may lead to overfitting. Therefore, we develop an unsupervised representation learning to use unlabeled data to handle this problem to learn a feature mapping function $f(.) : \mathcal{X} \rightarrow \mathcal{Z}$ where $\mathbf{z} = f(\mathbf{x})$ is a feature representation of input sample. In self-supervised learning, in absence of the label information, the representation $\mathbf{z}$ is optimized using a self-supervised loss function according to pre-defined pseudo labels. It is based on a pre-defined notion of similarity (positive labels) between embedding vectors of two pairs of data points versus the pre-defined dis-similarity(negative labels) between other pairs of data points.

For a given dataset, we use a self-supervised learning framework (as shown in Figure 1a) to learn latent feature representations. During this state, no label information is available for model learning (*i.e.* a pure self-supervised learning fashion). In order to validate the quality of the latent features for a classification task, during the fine-tuning stage, as shown in Figure 1b, we use a small number of labeled instances to train a classification model and validate the performance of the classifier trained using representations made from the concatenation of embedding vectors of feature subsets.



	f_1	f_2	f_3	f_4	f_5	f_6	f_7
$\mathbf{x}_1$	7	1	5	3	2	4	1
$\mathbf{x}_2$	2	5	6	8	9	7	2
$\mathbf{x}_3$	1	3	7	4	5	6	1
$\mathbf{x}_4$	8	4	1	1	2	1	5

Figure 2: A conceptual view of feature ordering using maximum spanning tree. Left: A toy tabular dataset containing four instances and seven features; Middle: Correlation between features (Lighter color denotes stronger correlation); Right: maximum spanning tree constructed using feature correlation.

3 PROPOSED FRAMEWORK

Self-supervised learning is typically defined through pretext tasks after applying data augmentation on data. Rather than doing regular contrastive learning on the entire feature set, in the proposed method, we consider a localized version of representation learning when the input features are split into different subsets. This method is developed based on the idea that in regular holistic unsupervised feature learning methods like Denoising Autoencoders[20], treating all input features equally may not be effective since it assumes that either they are independent or they contribute with almost similar levels to represent data. Pixel values in images have spatial or sequential correlations, making image cropping as one of the common operations for in data augmentation tasks [9, 18]. This motivates us to propose a method for tabular domain to divide the single neural network encoder into multiple modules being responsible to learn representation for the subsets of input features in the representation learning process.

Figure 1a illustrates the model using two subsets of features. The model contains two main components: (a) feature reordering and partitioning, which reorganizes features into subsets with reordered adjacency relationships. It is assumed that the model typically has non-overlapping feature subsets but some level of overlaps between feature subsets are also allowed; and (b) local contrastive learning which learns representations by combining convolutional autoencoder and contrastive optimization.

3.1 Tabular Feature Reordering

Since the order of features in tabular data does not necessarily follow spatial correlation like neighboring pixels in images or subsequential frames of videos, it is preceded by our own feature reordering step. This step leverages mutual correlation between input features to reorder them and let a specific deep learning structure like a convolutional neural layer in the encoder and decoder components be applied to learn effective local feature representations. Without this, although applying convolutional deep learning models is doable, but they do not come with any intuition. We propose to use mutual feature correlations to determine the new order of input features. In Fig. 2, we use Pearson correlation coefficients as the metric and create a feature-feature correlation matrix $\mathcal{M} \in \mathbb{R}^{m \times m}$ from all mutual Pearson correlation values. Assuming all features as nodes of a fully-connected graph in which the weights of the edges between nodes are presented by the absolute value of correlations, we create a maximum spanning tree where the sum of all weights of connecting edges is the maximum value as possible. The order of features is determined by a depth-first search starting from nodes that the edge with the highest value connects in the tree.

3.2 Local Contrastive Learning

After calculating the order of input features, we convert each instance into a data sample with the new order of features where feature values in the adjacent one-dimensional window of features contain Pearson correlations being similar to spatial correlation in images. So we use 1-D CNN in the deep learning network to learn new feature representations. Each sample in input data is divided by two feature subsets. For self-supervised learning, we apply a stochastic data augmentation on sample subsets through masking which randomly generates a binary matrix with a batch of data related to input feature subsets. Each vector in the binary mask is randomly sampled from a Bernoulli distribution with a pre-defined probability parameter. The corrupted version of each sample in either batch of feature subsets is calculated as follows:

$$\widetilde{\mathbf{x}}_i = \mathbf{x}_i \odot (\mathbf{1} - \mathbf{m}) + \bar{\mathbf{x}}_\mathbf{i} \odot \mathbf{m} \tag{1}$$

It is then fed to a designated encoder to transform the input data into a representation with respect to the selected subset, then the corresponding decoder is responsible to recover the original input data. Given the introduced data augmentation procedure, the optimization is done by using the linear combination of two loss functions with respect to the input feature value reconstruction task and the contrastive representation estimation task as follows:

$$\underset{\theta,\xi}{\operatorname{argmin}} \; \mathcal{L}_c + \alpha \; \mathcal{L}_r \tag{2}$$

where θ and ξ refer to trainable weights in encoder and decoder networks according to a pair of feature subsets. $\mathcal{L}_c$ is our contrastive loss function and $\mathcal{L}_r$ is the reconstruction loss function. We use the following loss function for a de-noising task to predict the original feature values from corrupted data vectors:

$$\mathcal{L}_r(\hat{\mathbf{X}}, \mathbf{X}) = \frac{1}{2} \sum_{j}^{2} \frac{1}{n} \sum_{i}^{n} (f_{d_j}(\mathbf{Z}^j) - \mathbf{X}^j)^2 \tag{3}$$

The reconstruction loss value is calculated as the average of minimum squared error between the reconstructed vectors and the original input over two feature subsets. The contrastive loss function between pair of representations of feature subsets are calculated via Barlow-twins contrastive loss function [22] by calculating a cross-multiplication matrix $\mathbf{C} = \mathbf{Z}^{1^T} \cdot \mathbf{Z}^{2^T} \in \mathbb{R}^{d \times d}$ where d is the the dimension of hidden layer in the model. It is computed through the dot-product of the batch of normalized embedding vectors. The matrix is going to be optimized to be equal to the identity matrix I_p in the following contrastive loss function:

$$\mathcal{L}_c(\mathbf{Z}^1, \mathbf{Z}^2) = |\mathbf{C} - \mathbf{I}_p|^2 = \sum_i (1 - \mathbf{C}[i,i])^2 + \lambda \sum_i \sum_{j \neq i} (\mathbf{C}[i,j])^2 \tag{4}$$

The contrastive loss function encourages the similarity of the pair of input feature subsets that are split from the same batch of data. The model learns from the level of inconsistency between a pair of corrupted data and also from the correlation between the non-mask area in pair of feature subsets. We expect that the aggregated representations learned by the encoder components can be employed in the fine-tuning step to be used in classification tasks. Algorithm 1 shows the pseudo-code of the proposed method.

4 EXPERIMENTS

4.1 Experimental Settings

Datasets. We evaluate the performance of the proposed method on six benchmark datasets publicly available on the UCI repository [6]. Table 1 describes the statistics of the datasets.

Table 1: Basic statistics of benchmark datasets used in the experiments

Dataset	# of features	# of Samples	# of Classes
MNIST	784	70,000	10
Income	14	48,842	2
BlogFeedback	280	60,021	2
Diabetic Retinopathy	20	1151	2
Wall-following	55	5456	4
Gas sensor array drift	128	13,910	4

Implementation details. In all experiments, we first use data pre-processing techniques to transform raw data into well-formed data formats. For the image dataset like MNIST, the pixel values follow approximately a Gaussian distribution. To normalize data before running experiments, we apply a simple min-max normalization to put input values in the range of [0,1]. For the other datasets, we apply standardization to get z-normalized data. The categorical features in the datasets like the adult income dataset are one-hot-encoded. Furthermore, we also assume that all features have non-zero standard deviation. Otherwise, we discard them for training procedures. We evaluate the performance of all studied models through 5-fold stratified cross-validation in which 90% of samples in the training data are randomly used as un-labeled data for the pre-training step. In the experiments setup, we use three 1-D CNN layers followed by max-pooling and up-sampling layers

Table 2: Target prediction results; Comparison between LoCL and the baseline methods. The evaluation metrics are mean ± std. of accuracy scores over 5-fold cross validation for the classification task. The number of latent dimension is shown within parentheses

Model/Dataset		MNIST(256)	INCOME(512)	BLOG(1024)	Diabetes(64)	Wall-follow.(64)	Gas sensor(512)	Average
Supervised Learning	LR	0.9221 ± 0.001	0.8243 ± 0.003	0.7728 ± 0.003	0.7280 ± 0.025	0.7008 ± 0.020	0.9902 ± 0.002	0.823 ± 0.01
	MLP	0.9743 ± 0.001	0.8501 ± 0.003	0.7885 ± 0.004	0.6977 ± 0.019	0.9129 ± 0.007	0.9891 ± 0.006	0.869 ± 0.01
	RF	0.9664 ± 0.002	0.8571 ± 0.003	0.8272 ± 0.003	0.6681 ± 0.035	0.9940 ± 0.003	0.9942 ± 0.002	0.885 ± 0.01
	XGBoost	0.9041 ± 0.002	0.8555 ± 0.004	0.8249 ± 0.003	0.6951 ± 0.042	0.9962 ± 0.002	0.9729 ± 0.002	0.875 ± 0.01
Self-supervised Learning	DAE [20]	0.8982 ± 0.006	0.8222 ± 0.003	0.7201 ± 0.002	0.6273 ± 0.031	0.6642 ± 0.025	0.9448 ± 0.009	0.779 ± 0.01
	Conv-DAE	0.9518 ± 0.003	0.8324 ± 0.005	0.7406 ± 0.006	0.5777 ± 0.061	0.6557 ± 0.021	0.9692 ± 0.003	0.788 ± 0.02
	Barlow-twins [22]	0.9431 ± 0.001	0.8378 ± 0.004	0.7507 ± 0.003	0.6386 ± 0.032	0.7269 ± 0.027	0.9807 ± 0.003	0.813 ± 0.01
	SimCLR [4]	0.9432 ± 0.003	0.8434 ± 0.005	0.7569 ± 0.003	0.6238 ± 0.055	0.6946 ± 0.022	0.9724 ± 0.006	0.806 ± 0.02
	VIME [21]	0.9377 ± 0.002	0.8458 ± 0.004	0.7406 ± 0.004	0.6186 ± 0.057	0.7382 ± 0.029	0.9628 ± 0.006	0.807 ± 0.02
	LoCL	**0.9540 ± 0.002**	**0.8461 ± 0.005**	**0.7783 ± 0.004**	**0.6438 ± 0.037**	**0.7479 ± 0.011**	**0.9825 ± 0.004**	**0.825 ± 0.01**

Table 3: Ablation studies to compare to the impact different encoder and the ordering of input features; average scores and standard deviation over all datasets are reported based on 5-fold cross-validation

Model Variants	Accuracy	Std
LoCL	**0.8254**	0.01
LoCL - Dense layer	0.8123	0.02
LoCL - Random ordering	0.8013	0.01
LoCL - Original order	0.8251	0.01
LoCL - Interleaved order	0.8070	0.01

and LeakyReLU activations in the body of the encoder and the decoder components respectively. We use RMSProp optimizer with a learning rate of 0.001. We set Bernoulli probability parameter to 0.3 in the data augmentation step. The optimal value of hyperparameters in the model (like trading parameter α, the kernel size in convolutional networks, *etc.*) are selected via cross-validation. All models are trained for a maximum of 200 epochs with an early stopping mechanism. After training the model, the trained model are used to transfer the remaining 10% of training labeled data to the new feature space, and train classifier.

Baselines. We compare LoCL with the following baseline:

- **DAE [20]:** a denoising autoencoder augmented with multiplicative mask-out noise.
- **Conv-DAE:** a 1-D denoising convolutional autoencoder
- **Barlow-Twins [22]:** a contrastive learning model with MLP as the encoder to do invariance optimization.
- **SimCLR [4]:** a contrastive learning model with MLP as the encoder to maximize mutual info using InfoNCE optimization. The projector in the model is skipped.
- **VIME [21]:** a model, which attachs a mask estimating decoder and a feature estimating decoder on top of the encoder.

4.2 Experimental Results

Table 2 demonstrates the performance of different supervised and self-supervised methods. We assess the performance of different baseline methods using accuracy metric on the separate testing data. For comparison purposes, we include supervised learning results which use label information of all training data to train classifiers (LR, MLP, RF, & XGBoost). This demonstrates the upper bound of self-supervised learning (which only uses 10% of label information to train the classifier).

According to the results, we can see in holistic autoencoder models, convolutional-based DAE models could obtain better performance than simple DAEs. Comparing with previous contrastive learning models like SimCLR and Barlow-Twins VIME models, we see improvements in accuracy scores in some datasets. It shows the importance of the self-supervised paradigm to learn informative representations for the downstream task. As for LoCL, it combines reconstruction and contrastive representation learning in one paradigm through local feature learning which leads to a superior performance against the state-of-the-art self-supervised methods.

Ablation Study. We have conducted additional ablation studies to measure separately the impacts of two main components of the proposed method LoCL including the feature order and local feature learning through convolutional encoding and decoding on the classification performance. We create variants of the proposed method when we vary the structure of the model from convolutional neural network to a network with dense layers. We also investigate the effect of different feature orders in the other variants of the model when we use random ordering, original feature order, and interleaved (every other) feature order. Table 3 confirms that the proposed feature ordering approach, along with local self-supervised learning empowered with convolutional networks, makes a great improvement in the performance.

5 CONCLUSION

In this paper, we introduced a new self-supervised method for learning feature representations for tabular data. We argued that existing methods largely rely on dense networks to learn feature representation, where dense networks aim to learn global patterns from all features. Since not all features are useful for learning tasks, and features often impose interactions, it is, therefore, more effective to learn local features. Alternatively, our model proposes a new feature reordering using feature-feature correlations and applies local feature learning to reordered feature subsets by using convolutional neural network modeling, combined with contrastive self-supervised learning. Experiments confirm the performance gain of the proposed method versus the state-of-the-art methods.

ACKNOWLEDGMENTS

This research is supported by the US National Science Foundation (NSF) through Grants IIS-1763452 and IIS-2027339.

REFERENCES

[1] Sercan O. Arik and Tomas Pfister. 2019. TabNet: Attentive Interpretable Tabular Learning. *arXiv e-prints*, Article arXiv:1908.07442 (Aug. 2019), arXiv:1908.07442 pages. arXiv:1908.07442 [cs.LG]

[2] Dara Bahri and et al. 2021. SCARF: Self-Supervised Contrastive Learning using Random Feature Corruption. *arXiv e-prints*, Article arXiv:2106.15147 (2021).

[3] Chun-Hao Chang, Jinsung Yoon, Sercan Arik, Madeleine Udell, and Tomas Pfister. 2022. Data-Efficient and Interpretable Tabular Anomaly Detection. *arXiv e-prints*, Article arXiv:2203.02034 (March 2022), arXiv:2203.02034 pages. arXiv:2203.02034 [cs.LG]

[4] Ting Chen and et al. 2020. A Simple Framework for Contrastive Learning of Visual Representations. *arXiv e-prints*, Article arXiv:2002.05709 (Feb. 2020).

[5] Sajad Darabi and et al. 2021. Contrastive Mixup: Self- and Semi-Supervised learning for Tabular Domain. *arXiv e-prints*, Article arXiv:2108.12296 (2021).

[6] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[7] Zhabiz Gharibshah, Xingquan Zhu, Arthur Hainline, and Michael Conway. 2020. Deep Learning for User Interest and Response Prediction in Online Display Advertising. *Data Science and Engineering* 5 (2020), 12–26.

[8] Yury Gorishniy, Ivan Rubachev, and et al. 2021. Revisiting Deep Learning Models for Tabular Data. *arXiv e-prints*, Article arXiv:2106.11959 (2021).

[9] Jean-Bastien Grill, Florian Strub, Florent Altché, and et al. 2020. Bootstrap your own latent: A new approach to self-supervised Learning. *arXiv e-prints*, Article arXiv:2006.07733 (June 2020), arXiv:2006.07733 pages. arXiv:2006.07733 [cs.LG]

[10] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. 2022. Why do tree-based models still outperform deep learning on tabular data? *arXiv preprint arXiv:2207.08815* (2022).

[11] Huimei Han, Xingquan Zhu, and Ying Li. 2019. Convolutional neural network learning for generic data classification. *Information Sciences, vol.477, pp.448-465* (2019).

[12] Arlind Kadra, Marius Lindauer, and et al. 2021. Well-tuned Simple Nets Excel on Tabular Datasets. *arXiv e-prints*, Article arXiv:2106.11189 (2021).

[13] Xingchen Liu, Qicai Zhou, and et al. 2019. Fault Diagnosis of Rotating Machinery under Noisy Environment Conditions Based on a 1-D Convolutional Autoencoder and 1-D Convolutional Neural Network. *Sensors* 19, 4 (2019).

[14] Tom Shenkar and Lior Wolf. 2022. Anomaly Detection for Tabular Data with Internal Contrastive Learning. In *ICLR*. https://openreview.net/forum?id=_hszZbt46bT

[15] Ravid Shwartz-Ziv and Amitai Armon. 2021. Tabular Data: Deep Learning is Not All You Need. *arXiv e-prints*, Article arXiv:2106.03253 (June 2021), arXiv:2106.03253 pages. arXiv:2106.03253 [cs.LG]

[16] Gowthami Somepalli, Micah Goldblum, and et al. 2021. SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training. , Article arXiv:2106.01342 (June 2021), arXiv:2106.01342 pages. arXiv:2106.01342 [cs.LG]

[17] Weiping Song, Chence Shi, and et al. 2018. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. , Article arXiv:1810.11921 (Oct. 2018), arXiv:1810.11921 pages. arXiv:1810.11921 [cs.IR]

[18] Talip Ucar, Ehsan Hajiramezanali, and et. al. 2021. SubTab: Subsetting Features of Tabular Data for Self-Supervised Representation Learning. *NIPS* 34 (2021).

[19] Aaron van den Oord, Yazhe Li, and et al. 2018. Representation Learning with Contrastive Predictive Coding. *arXiv e-prints*, Article arXiv:1807.03748 (July 2018), arXiv:1807.03748 pages. arXiv:1807.03748 [cs.LG]

[20] Pascal Vincent, Hugo Larochelle, and et al. 2008. Extracting and Composing Robust Features with Denoising Autoencoders *(ICML '08)*. 1096–1103.

[21] J. et al Yoon. 2020. VIME: Extending the Success of Self- and Semi-supervised Learning to Tabular Domain. In *NIPS*, Vol. 33. 11033–11043.

[22] Jure Zbontar, Li Jing, and et al. 2021. Barlow Twins: Self-Supervised Learning via Redundancy Reduction. , Article arXiv:2103.03230 (2021).

[23] Yutao Zhu, Jian-Yun Nie, Zhicheng Dou, Zhengyi Ma, Xinyu Zhang, Pan Du, Xiaochen Zuo, and Hao Jiang. 2021. Contrastive Learning of User Behavior Sequence for Context-Aware Document Ranking. *arXiv e-prints*, Article arXiv:2108.10510 (Aug. 2021), arXiv:2108.10510 pages. arXiv:2108.10510 [cs.IR]

[24] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Graph Contrastive Learning with Adaptive Augmentation. *arXiv e-prints*, Article arXiv:2010.14945 (Oct. 2020), arXiv:2010.14945 pages. arXiv:2010.14945 [cs.LG]