Split Learning Based on Self-supervised Learning

Shaojie Yang[1](✉), Hao Chen[2], Jianping Huang[2], Yong Yan[2], Jiewei Chen[1], and Ao Xiong[1]

[1] Beijing University of Posts and Telecommunications, Beijing 100876, China
412121029@qq.com

[2] State Grid Zhejiang Electric Power Co., Ltd. Research Institute, Hangzhou, China

Abstract. Vertical federated learning is a type of federated learning which aims to achieve feature fusion of different participants, and the data possessed at distinct participants usually contain different features. In vertical federated learning, distributed deep learning schemes represented by Split learning have received extensive attention. However, Split learning has problems such as label leakage due to frequent gradient interactions. Aiming to solving the above problems, and considering that there are many unused non-overlapping data in the vertical federated learning participants, we propose Split learning based on Self-supervised learning (self-split learning). We split the model into presentation layers and inference layers. The participants first perform self-supervised learning based on the idea of autoencoder on non-overlapping data. After the training of the presentation layers is completed, the server aggregates the overlapping data encoded by each participant, and completes the training of the inference layers independently. By truncating the gradient propagation between the server and the participants, the scheme proposed in this paper effectively solves a series of problems caused by gradient leakage, enables privacy protection. At the same time, our method only requires participants to upload once, which reduces communication overhead and alleviates disconnection, etc. question. We conducted experiments on the financial risk control dataset, and the experiments proved that our algorithm is competitive in performance with current mainstream vertical federated learning algorithms.

Keywords: Vertical federated learning · Self-supervised learning · Privacy protection

1 Introduction

The training of high-quality machine learning models is inseparable from a large amount of high-quality data. However, in real scenarios, data often exists in the form of isolated islands. With the improvement of users' data privacy awareness and the improvement of data protection regulations, the traditional method of collecting data to the cloud for unified training is no longer applicable. Federated learning is based on the idea of "data does not move and model moves", which can realize data sharing under the premise of ensuring the data security of the participants, and has received more and more attention.

Generally speaking, federated learning is divided into three categories: horizontal federated learning, vertical federated learning, and federated transfer learning. In vertical federated learning, multiple participants work together to process data from the same individual set, but each client has only a unique set of features. There are many application scenarios of vertical federated learning. For example, financial institutions need to establish risk control models to conduct risk management or risk control of financial activities. Financial institutions have financial-related characteristics such as user loan history and repayment status, and financial institutions can establish risk control models based on all their data. At the same time, financial institutions can jointly conduct vertical federated learning with e-commerce, and e-commerce has data such as users' recent consumption, which may be helpful for the establishment of risk control models. The vertical federated learning of financial institutions and e-commerce can be regarded as the expansion of the feature dimension (Fig. 1).

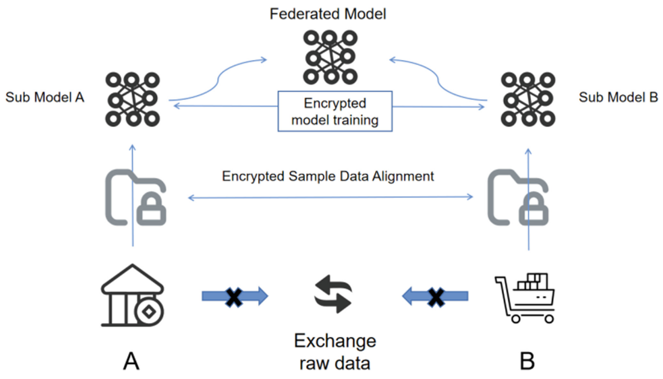

Fig. 1. A scenario of vertical federated learning.

As a branch of machine learning, deep learning has shown great potential. Therefore, the research on vertical federated learning based on neural network has received extensive attention. The Split learning architecture is proposed in [1]. The participants perform hierarchical segmentation on the deep learning model, and complete the joint deep learning model without sharing sensitive data. Split learning only interacts with the intermediate calculation results of the split layer between the server side and the participants side, and is considered to be a flexible and efficient distributed deep learning solution. However, like many existing federated learning schemes, this scheme requires multiple gradient transmissions. Many existing studies have shown that the gradient will leak the information of the original data. Therefore, this scheme has defects in security. Specific risks are discussed in Sect. 3.

Vertical federated learning requires encrypted entity alignment, that is, to find the intersection of the data of each participant, which we call overlapping data. The rest of the data we call non-overlapping data. For non-overlapping data, the corresponding samples are missing in some participants, and the incomplete data prevents them from participating in the training of traditional vertical federated learning schemes. In other words, in vertical federated learning, these data are wasted. Based on the above discussion, we have come up with a simple idea whether we can reasonably use non-overlapping data to improve the Split learning scheme (Fig. 2).

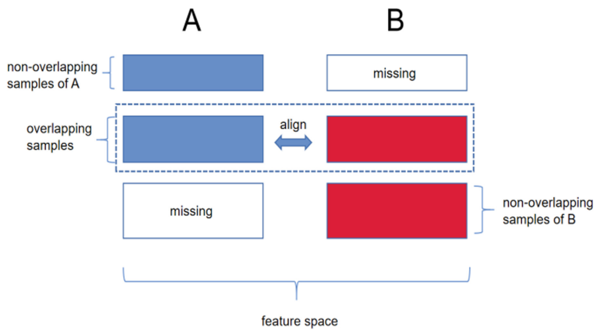

Fig. 2. Overlapping data and non-overlapping data.

Split learning needs to upload the calculation results of the cutting layer to the server. These intermediate calculation results can be regarded as a mapping of the original data of the participants. Therefore, it is necessary to find an effective mapping method that can retain the original data information. In light of the above considerations, we explored training data mapping methods on non-overlapping datasets. Non-overlapping datasets of various parties can be viewed as unlabeled data, and our method is based on self-supervised learning. Self-supervised learning uses artificially defined tasks on unlabeled datasets, hoping to learn a general feature expression that can be used for downstream tasks. Participants can flexibly define self-supervised training tasks. After each participant completes the self-supervised training task on non-overlapping data, the mapping method is determined. On overlapping datasets, each participant completes the mapping of the data they own, and aggregates the mapped data on the server side. We believe that these mapped data will not reveal the information of the original data. The server side performs inference layer training based on the aggregated data, and converts the feature expressions of each participant into the final output.

Our contributions are as follows: 1. Based on the idea of autoencoder, we propose a split learning scheme based on self-supervised learning, which truncates the back-propagation process between the server and the participant, effectively avoiding gradient leakage, is a safer distributed deep learning scheme. 2. We design a self-supervised learning task to complete the training of the representation layers, and train robust representation layers by restoring the original data with noise, and its output can retain most of the information of the original data. 3. We have verified our algorithm on the financial risk control dataset. The experiment proves that our algorithm is still comparable to the current mainstream vertical federated learning algorithm in performance while improving the security.

2 Related Work

2.1 Vertical Federated Learning

[3] proposed an efficient and secure SVM algorithm for vertically partitioned data. [4] designed a linear regression algorithm for vertically partitioned datasets. But the designs in [3, 4] are all linear algorithms. [5] proposed a boosted tree algorithm SecureBoost under the federated learning setting based on vertically partitioned datasets. SecureBoost provides the same level of accuracy as non-privacy-preserving methods, while not revealing the information of each private data provider. With the rise of deep learning, more research on neural network models in vertically federated learning has emerged. In [1], a distributed deep learning scheme, Split learning, is proposed, and several settings of Split learning are pointed out, which can allow healthy entities to collaboratively train deep learning models without sharing sensitive raw data. [6] pointed out that most of the non-overlapping data in the traditional vertical federated learning algorithms are not fully utilized, and proposed a semi-supervised learning algorithm to predict the pseudo-labels of unlabeled samples to expand the training set.

2.2 Self-supervised Learning

Self-supervised learning can avoid extensive labeling of datasets, use self-defined pseudo-labels as training signals, and then use the learned representations for downstream tasks. Autoencoder is a type of self-supervised learning that aims to learn to reconstruct input observations with the lowest possible error, creating a useful and meaningful latent representation. The concept of autoencoder was first proposed in [7]. After that, many improved autoencoder models have been proposed, including denoising autoencoder proposed in [8] and the variational autoencoder proposed in [9]. These improved autoencoder models are designed to make the hidden layer expression more meaningful. Nowadays, self-supervised learning has received extensive attention in computer vision, natural language processing and other fields. [10] used self-supervised learning for model pre-training, enabling the BERT model to achieve huge improvements on a range of NLP tasks. In the field of computer vision, the MoCo model in [11] and the SimCLR model in [12] both show strong competitiveness.

3 Split Learning Based on Self-Supervised Learning

Consider the participants set $M = \{1, 2...\mathrm{M}\}$, the participants' overlap dataset D_u, whose size is N_u, the data $\{x_{u,n}\}_{n=1}^{N_u} \in R^d$ is scattered among the participants, and each participant holds a part of the data. For example, the participant m holds the feature $x_{u,n,m}$, $n = 1, 2...N_u$, $x_{u,n,m}$ is a part of $x_{u,n} = [x_{u,n,1}, x_{u,n,2}...x_{u,n,M}]$ in the participant m. $\{y_{u,n}\}_{n=1}^{N_u} \in \{0, 1\}$ stored on the server side.

The data set of participant m is $D_m \in R^{d_m}$, D_m includes $D_{u,m}$ and $D_{l,m}$, $x_{u,n,m} \in D_{u,m}$, $D_{l,m}$ represents non-overlapping data belonging to participant m, for any pair of participants m_1, m_2 in M, the counts of D_{u,m_1} and D_{u,m_2} are equal, but the counts of D_{l,m_1} and D_{l,m_2} are not necessarily equal.

Without loss of generality, we consider that vertical federated learning has two participants A and B, which can be easily extended to multiple participants. In the traditional vertical federated learning methods, after completing the data alignment, vertical federated training is performed on the overlapping dataset D_u, the data in D_u is fully utilized, but it can be seen that $D_{l,A}$ and $D_{l,B}$ do not participate in the process of federated training, that is, non-overlapping data is wasted.

The Split learning architecture is proposed in [1], where each participant trains only a part of the "split layer" of the entire deep neural network. However, this architecture still suffers from the above-mentioned problem, that is, the non-overlapping data is not fully utilized. Drawing on the ideas in [1], we believe that the encoded data will not reveal the participants' private information, and we use non-overlapping data to complete the training of the representation layers to realize the utilization of non-overlapping data. The representation layers can implement the encoding of the participant's raw input. On overlapping datasets, participants can use the already trained representation layers to complete the encoding task, and send the encoded output to the server. Our scheme is very flexible. On the one hand, it is reflected in the fact that the form of the original data is not limited, which can be vectors, pictures, texts, etc. On the other hand, in the process of self-supervised learning, participants can flexibly set up self-supervised learning tasks.

3.1 Representation Layers Training

The representation layer implements the encoding of the original input of the participant. According to the original input data form of the participant, it can be a neural network, a convolutional neural network, a recurrent neural network, or other structures. Considering participant A, $\mathrm{x} \in D_A$, the representation layers model of participant A is expressed as $f_A(.)$, then the encoding result of x is

$$x_{enc} = f_A(x) \tag{1}$$

We use self-supervised learning for the training of representation layers based on the idea of autoencoder.

The autoencoder mainly includes an encoder and a decoder. The encoder compresses the original data, and the decoder decompresses the encoded data, and strives to make the decompressed data as close to the original input data as possible. The hidden layer usually has the least number of neurons. The encoder can extract data that has representative information in the original data. When self-supervised training is performed on the participant, the presentation layers act as the encoder part, and the decoder part is as symmetrical as possible to the encoder part. We use the data in $\mathrm{D}_{\mathrm{n,A}}$ to train the autoencoder. For $\mathrm{x}_{\mathrm{l,n,A}} \in \mathrm{D}_{\mathrm{l,A}}$, let the decoder model of participant A be expressed as $\mathrm{h}_{\mathrm{A}}(.)$, then the output of the original data of participant A through the autoencoder is

$$\mathrm{x}^{'} = h_A(f_A(\mathrm{x}_{l,n,A})) \tag{2}$$

During self-supervised training, the training objective to minimize is

$$|\mathrm{x}^{'} - \mathrm{x}_{l,n,A}|_2 \tag{3}$$

The representation layer learned by the self-supervised learning scheme based on the above autoencoder is sensitive to outliers, lacks robustness, and the extracted features are not effective for downstream tasks. Therefore, we consider optimizing the self-supervised learning task (Fig. 3).

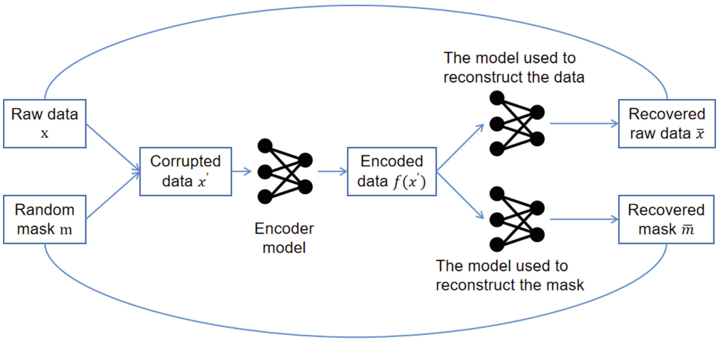
Raw data x
Random mask m
Corrupted data x'
Encoder model
Encoded data $f(x')$
The model used to reconstruct the data
The model used to reconstruct the mask
Recovered raw data $\bar{x}$
Recovered mask $\bar{m}$

Fig. 3. Self-supervised training process.

For a random sample $\mathrm{x_{l,n,A}} \in \mathrm{D_{l,A}}$, we generate a random mask $m_{n,A}$ for it, $m_{n,A} \in \{0, 1\}^{d_A}$. Based on $\mathrm{x_{l,n,A}}$ and $m_{n,A}$ we produce corrupted data $x'_{l,n,A}$, we use the encoder model to encode $x'_{l,n,A}$, get the encoded data $f(x'_{l,n,A})$, then take $f(x'_{l,n,A})$ as input, use the data recovery model and the mask recovery model to restore the original data $\overline{\mathrm{x_{l,n,A}}}$ and the mask $\overline{m_{n,A}}$. The encoder model, the data recovery model and the mask recovery model are all neural networks. The loss function includes two parts, the mean square error l_d of the restored data and the original data, and the relative entropy l_m of the restored mask and the random mask, the total loss can be expressed as

$$(1-\alpha) * \mathrm{l}_m(m_{n,A}, \overline{m_{n,A}}) + \alpha * \mathrm{l}_d(\mathrm{x_{l,n,A}}, \overline{\mathrm{x_{l,n,A}}}) \tag{4}$$

where α represents the weight parameter, which adjusts the weight of the data recovery error and the mask recovery error.

The training of the encoder model, data recovery model and mask recovery model is performed according to the back-propagation of the training error defined in (4). After training, the encoder model will be used in downstream tasks.

3.2 Inference Layers Training

After each participant has completed the training of the representation layers on the non-overlapping datasets. For $\mathrm{x}_{u,n,A} \in D_{u,A}$, $\mathrm{x}_{u,n,B} \in D_{u,B}$, we use the trained representation layers $f_A(.)$, $f_B(.)$ to encode. The encoded data connection is used as the encoded representation of a sample, and the representation incorporates the features of the corresponding samples at client A and client B.

$$\mathrm{x}_{u,n,A,enc} = f_A(\mathrm{x}_{u,n,A}) \tag{5}$$

$$\mathrm{x}_{u,n,B,enc} = f_B(\mathrm{x}_{u,n,B}) \tag{6}$$

$$\mathrm{x}_{u,n,enc} = concat(\mathrm{x}_{u,n,A,enc}, \mathrm{x}_{u,n,B,enc}) \tag{7}$$

The inference layers are trained using the fused encoded representation, and the inference layers implement the transformation of the fused encoding into the final output. Denote the inference layer model as g(.), then for $\mathrm{x}_{u,n,A}$, $\mathrm{x}_{u,n,B}$, the inference layers output result $\mathrm{y}'_{u,n}$ is

$$\mathrm{y}'_{u,n} = g(\mathrm{x}_{u,n,enc}) \tag{8}$$

We set l_{Infer} as the loss function of $\mathrm{y}'_{u,n}$ and $y_{u,n}$ which depends on our goals. The training objective of the inference layers to minimize is

$$l_{Infer}(\mathrm{y}'_{u,n}, y_{u,n}) \tag{9}$$

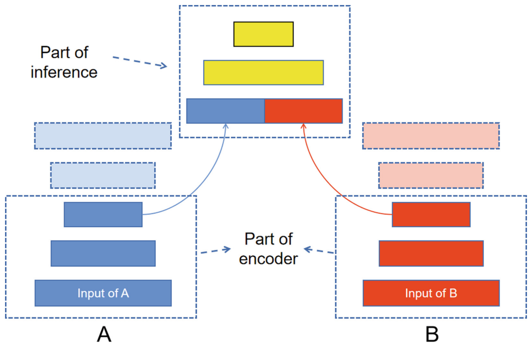

Fig. 4. Split learning based on self-supervised learning.

Without loss of generality, the loss in (9) uses the mean squared error. In [2], it is pointed out that the confidence of the model in judging a positive example as a positive example is often less than the confidence in judging a negative example as a negative example. That is, for a positive sample y_+, the model predicts y'_+ for it, and a negative example y_- and the model predicts y'_-, $\left|\mathrm{y}_+ - y'_+\right|$ is often greater than $|y'_- - y\_|$. Regardless of whether the sample is positive or negative, $||\nabla_a g(a)|_{a=f(x)}|_2$ is often very close, because $||\nabla_a g(a)|_{a=f(x)}|_2$ has nothing to do with y. According to the chain derivation rule, the output gradient of the model error to the representation layers is $||\nabla_a g(a)|_{a=f(x)}|_2 * \left|\mathrm{y}_+ - y'_+\right|$ or $||\nabla_a g(a)|_{a=f(x)}|_2 * |y'_- - y\_|$. Therefore, for positive and negative examples, if the gradient is sent back to the participant, the gradient values of the positive and negative samples are often quite different. The participant can deduce the sample label accordingly although the participant may not have permission to see it. At the same time, in the process of gradient backhaul, considering the insecurity of the link, the backhauled gradient may be at risk of being stolen or tampered with. In some real-world scenarios, such label leakage is unacceptable. Therefore, we consider truncating the gradient return at the participant side, and the server side independently completes the training of the inference layers to avoid label leakage and potential risk of gradient leakage. At the same time, traditional federated learning faces the challenge of device heterogeneity, which leads to different computing times on the participant side or even disconnection. The server-side independently completes the inference layers training to avoid frequent transmission of gradients, effectively avoiding the negative impact of device heterogeneity. Although the representation layers and inference layers are trained on different datasets in the scheme proposed in this paper, we improve the robustness of the representation layers by reasonably setting the task of self-supervised learning, so that the output encoding retains most information of original data which will not cause a decline in the overall model performance (Fig. 4 and Table 1).

Table 1. The flow of our proposed algorithm.

```
Split learning based on Self-supervised learning
Input:
Client A, client B and Server;
Datasets $D_{u,A}$,$D_{l,A}$ of A, Datasets $D_{u,B}$,$D_{l,B}$ of B, set of labels in Server $Y$ ;
Class probability threshold $t$ , the epoch number of Presentation layer training $K_P$ $and$ the
epoch number of Inference layer training $K_I$;
Output:
Model of Presentation layers $\mathrm{h_A}$,$\mathrm{h_B}$ and model of Inference layers $g$;
For e = 1,2,…, $K_P$ do
    Select a mini batch $\mathrm{x}_{l,e,A} \subseteq D_{l,A}$ and $\mathrm{x}_{l,e,B} \subseteq D_{l,B}$
    Learn $\mathrm{h_A}$ $and$ $\mathrm{h_B}$ through $\mathrm{x}_{l,e,A}$ and $\mathrm{x}_{l,e,B}$
End for
A send $\mathrm{h_A}(D_{u,A})$ to Server
B send $\mathrm{h_B}(D_{u,B})$ to Server
For $k$ = 1,2,…, $K_I$ do
    Select a mini batch $\mathrm{x}_{u,k,A} \subseteq \mathrm{h_A}(D_{u,A})$ ,$\mathrm{x}_{u,k,B} \subseteq \mathrm{h_B}(D_{u,B})$ and $y_{u,k} \subseteq \mathrm{Y}$
    Learn $g$ through $\mathrm{x}_{u,k,A}$,$\mathrm{x}_{u,k,B}$ and $y_{u,k}$
End for
```

4 Experiment

We verify our proposed algorithm on the financial risk control dataset. The financial risk control dataset includes more than 800,000 loan default data of financial risk control, loan records from a credit platform, each data contains 47 columns of variable information, including loan amount, loan term, loan interest rate, year income and other information, we design a model to predict whether the loan defaults. We first split the data set, with 70% of the data as the training set and 30% of the data as the validation set. Following the setting of vertical federated learning, we divided the dataset by variables, randomly selected half of the variables to be placed in participant A, and the other half of the variables were placed in participant B. Set the overlapping data of participant A and participant B to be 10% of the training set. In the remaining 90% of the training set data, half of the data is used as the non-overlapping data of participant A, and half of the non-overlapping data of participant B. A and B's non-overlapping data samples do not overlap. We contrast the algorithm with vertical logistic regression methods and split learning methods (Fig. 5).

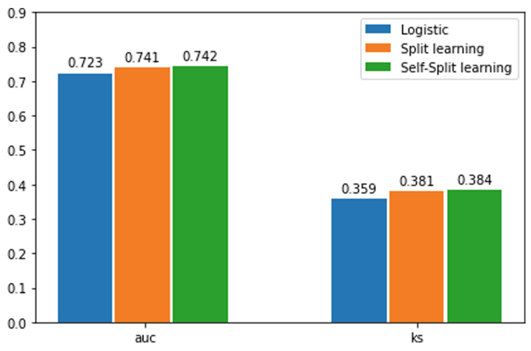

Fig. 5. Experimental results.

We verify the algorithms on the validation set, and use AUC and KS as the evaluation metrics of the algorithms. We find that Split learning and the proposed algorithm perform better on AUC and KS than the logistic regression algorithm. We believe this is because Split learning and the algorithm proposed in this paper are based on neural network models, and the nonlinearity of neural network models improves the performance of the models. Compared with split learning, the algorithm proposed in this paper has no obvious advantages in AUC and KS, but the algorithm proposed in this paper is a safer version of split learning. The presentation layers retain the important information of the original data while encoding, so the algorithm proposed in this paper has no obvious performance loss compared to Split learning.

5 Conclusion

In this paper, we propose split learning based on self-supervised learning. The client completes the training of the presentation layer on the non-overlapping data set through self-supervised learning, and the presentation layers can encode the non-overlapping data of the participants. In order to reduce the risk of privacy data leakage caused by gradient transmission between the participant and the server, the participant can encode the original data of the overlapping data part through the presentation layer, send the encoded data to the server, and the server collects all participant data. After the encoded data is sent, the data is aggregated and the presentation layers are trained independently. It should be emphasized that our algorithm is very flexible. Our framework does not limit the data type of the client. Different participants can even hold different data types to achieve a multi-modal vertical federated learning. At the same time, the participants can set self-supervised learning tasks flexibly. At the same time, we do not limit the model structure on the server side. We design a self-supervised learning task that restores the original data with added noise and trains a high-quality representation layer model to ensure that the encoded data retains the important information of the original data. We conducted experiments on the algorithm proposed in this paper on the financial risk control dataset. The experiments show that our algorithm can achieve performance comparable to the current mainstream vertical federated learning algorithms.

Acknowledgement. This work was supported by State Grid Corporation of China Science and Technology Project "Research and application of industry chain finance key technology based on blockchain" (5211DS21NOOU).

References

1. Praneeth Vepakomma, et al.: Split learning for health: Distributed deep learning without sharing raw patient data (2018). arXiv: Learning:n.pag
2. Li, O., et al.: Label leakage and protection in two-party split learning (2021). arXiv: Learning:n.pag
3. Yu, H., et al.: Privacy-preserving SVM classification on vertically partitioned data. In: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, vol. 3918, pp. 647–656 (2011)
4. Kikuchi, H., et al.: Privacy-preserving multiple linear regression of vertically partitioned real medical datasets. J. Inf. Process. **26**, 638–647 (2018)
5. Cheng, K., et al.: SecureBoost: a lossless federated learning framework (2019). abs/1901.08755
6. Kang, Y., Liu, Y., Chen, T.: Fedmvt: semi-supervised vertical federated learning with multiview training (2020). arXiv preprint arXiv:2008.10838
7. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
8. Vincent, P., Larochelle, H., Lajoie, I., et al.: Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. **11**(12) (2010)
9. Kingma, D.P., Welling, M.: Auto-encoding variational bayes (2013). arXiv preprint arXiv: 1312.6114
10. Devlin, J., et al.: Bert: Pre-training of deep bidirectional transformers for language understanding (2018). arXiv preprint arXiv:1810.04805
11. He, K., et al.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020)
12. Chen, T., et al.: A simple framework for contrastive learning of visual representations. In: Proceedings of the International Conference on Machine Learning, PMLR (2020)