mathematics

MDPI

Article

TabFedSL: A Self-Supervised Approach to Labeling Tabular Data in Federated Learning Environments

Ruixiao Wang [1,†], Yanxin Hu [1,†], Zhiyu Chen [1,2], Jianwei Guo [1] and Gang Liu [1,2,*]

[1] School of Computer Science and Engineering, Changchun University of Technology, Changchun 130102, China; 2202203076@stu.ccut.edu.cn (R.W.); 2202103079@stu.ccut.edu.cn (Y.H.); chenzhiyu@ccut.edu.cn (Z.C.); guojianwei@ccut.edu.cn (J.G.)
[2] Jilin Province Data Service Industry Public Technology Research Centre, Changchun 130102, China
* Correspondence: lg@ccut.edu.cn
† These authors contributed equally to this work.

Abstract: Currently, self-supervised learning has shown effectiveness in solving data labeling issues. Its success mainly depends on having access to large, high-quality datasets with diverse features. It also relies on utilizing the spatial, temporal, and semantic structures present in the data. However, domains such as finance, healthcare, and insurance primarily utilize tabular data formats. This presents challenges for traditional data augmentation methods aimed at improving data quality. Furthermore, the privacy-sensitive nature of these domains complicates the acquisition of the extensive, high-quality datasets necessary for training effective self-supervised models. To tackle these challenges, our proposal introduces a novel framework that combines self-supervised learning with Federated Learning (FL). This approach aims to solve the problem of data-distributed training while ensuring training quality. Our framework improves upon the conventional self-supervised learning data augmentation paradigm by incorporating data labeling through the segmentation of data into subsets. Our framework adds noise by splitting subsets of data and can achieve the same level of centralized learning in a distributed environment. Moreover, we conduct experiments on various public tabular datasets to evaluate our approach. The experimental results showcase the effectiveness and generalizability of our proposed method in scenarios involving unlabeled data and distributed settings.

Keywords: Federated Learning; self-supervised learning; tabular data; deep learning

MSC: 68T07

Citation: Wang, R.; Hu, Y.; Chen, Z.; Guo, G.; Liu, G. TabFedSL: A Self-Supervised Approach to Labeling Tabular Data in Federated Learning Environments. *Mathematics* **2024**, *12*, 1158. https://doi.org/10.3390/math12081158

Academic Editors: Kunpeng Liu, Pengfei Wang, Yifeng Gao and Yanjie Fu

Received: 4 March 2024
Revised: 4 April 2024
Accepted: 8 April 2024
Published: 12 April 2024

1. Introduction

Self-supervised learning has demonstrated its ability to yield meaningful results when applied to high-quality, extensive datasets characterized by distributed features [1–6]. Simultaneously, significant strides have been made in natural language processing, audio analysis, and image recognition through the employment of data augmentation techniques [7–16]. Notably, self-supervised learning exhibits enhanced robustness compared with supervised learning in addressing the class imbalance problem within data, both in in-domain and out-of-domain evaluation scenarios [5]. These advancements are facilitated by methods such as data augmentation [9] and predefined task generation strategies [11]. However, in several scenarios such as finance, healthcare, and insurance, tabular data predominate. This data type lacks spatial, temporal, and semantic structures, rendering efficient augmentation challenging and potentially compromising training efficacy. Moreover, given the heightened importance of preserving user confidentiality in these domains, acquiring high-quality data to enhance training outcomes becomes increasingly challenging. The limited presence of tabular data in self-supervised FL can be attributed to the complexity

of designing effective enhancement methods tailored for such data types and the inherent challenges in sourcing high-quality data in these domains.

The predominant approach to enhancing tabular data involves disrupting their structure and introducing noise to augment the dataset [17]. However, uniformly treating all tabular data with noise addition may yield disparate enhancement outcomes, thus sub-optimizing the efficacy of data augmentation. Moreover, addressing estimation and measurement errors poses challenges as determining the trade-off between bias and efficiency in managing various specifications of masking and measurement errors remains elusive [18]. Consequently, self-supervised learning offers a promising avenue by transforming the single-view problem into a multi-view framework, generating reconstructed data through feature subsets. This approach effectively ameliorates issues relating to data quality and quantity. Nonetheless, self-supervised learning relies on high-quality features to yield improved data outcomes, and co-aggregating data for training is often impractical in high-privacy contexts. FL emerges as a viable solution, facilitating distributed learning in privacy-constrained environments. FL effectively addresses data silos by securely aggregating data for distributed training, aligning well with stringent requirements across diverse environments [19]. By integrating FL with self-supervised learning, we harness the potential to leverage high-quality data for training superior models in data labeling tasks.

In this study, we extend the paradigm of training efficient models using a single data source to encompass multiple data sources for simultaneous model training within a FL framework, akin to secure training with an enhanced data volume across distributed environments. Concurrently, by leveraging self-supervised learning on clients' unlabeled data, we ensure the high quality of our models. Our experimental results demonstrate that TabFedSL achieves comparable performance to SubTab under the Mnist, UCI Adult Income, and UCI BlogFeedback datasets in the tabular setting. This validates the efficacy of our FL approach in scenarios characterized by non-independent and non-identically distributed (Non-IID) data, underscoring the robustness of our FL framework. Additionally, the robustness of TabFedSL's framework is further substantiated through federal hyperparameter adjustment experiments.

(1) Our proposition introduces self-supervised learning within a FL framework, offering a solution tailored to address the challenge of tabular data labeling in sensitive contexts.
(2) We enhance the conventional data augmentation scheme for self-supervised learning by incorporating a novel approach of partitioning the dataset and applying noise to augment the subset data.
(3) Our work also utilizes several public tabular datasets, and the experimental results demonstrate the effectiveness and generalizability of our proposed approach in unlabeled data and distributed scenarios.

2. Literature Review

In this section, we enumerate recent significant contributions in the fields of FL and self-supervised learning. We delve into the rationale behind selecting FL and self-supervised learning as the approach to tackle the challenge of intricate tabular data labeling, considering the current state-of-the-art advancements in these domains.

2.1. Federated Learning

Issues such as cloud computing capacity limitations, data security concerns, and data silos are increasingly recognized as barriers to fostering trust, gathering private user data, and facilitating federated training [20]. Consequently, to mitigate these challenges, there arises a critical need for effective solutions in privacy-sensitive scenarios. Such solutions aim to bolster user trust while enhancing the performance of training models. It is within this context that the concept of FL emerged.

The concept of FL was pioneered by Google's team in 2016 with the aim of enabling users to achieve improved local model performance in privacy-sensitive scenarios without compromising their privacy. Google has since developed FL frameworks suitable for

deployment in real production environments. These frameworks enable customers to engage in joint training utilizing algorithms such as FedAvg, all while safeguarding the confidentiality of their information [21].

As a secure distributed machine learning approach, FL facilitates collaborative model training across multiple nodes while upholding the privacy of local data [22]. By decentralizing data processing to individual local nodes, reliance on a central server is diminished. Within this framework, the central server's role is primarily confined to aggregating parameter updates, thereby avoiding direct data processing. Consequently, global models can be trained to exhibit robust performance even in the presence of decentralized data. This mechanism contrasts with traditional centralized training methods, which typically necessitate processing all information centrally on a server [23]. FL also enables cross-sectoral data sharing, thereby broadening the sample size and data dimensions to support the development of high-precision big data models and applications. Consequently, FL facilitates the provision of high-quality data services, thereby generating greater societal value.

FL technologies have witnessed significant advancements over time, leading to practical applications across various domains, including smart healthcare [24], recommendation systems, smart cities [25], banking, edge training [26], and cybersecurity, among others. To cater to the requirements of these diverse applications, efficient central aggregation algorithms are essential to strike an optimal balance between preserving data security and enhancing computational efficiency. Through connections facilitated by secure platforms, different organizations have embraced the implementation of FL, thereby fostering the development of FL-based data collaboration models aligned with legal compliance standards.

In academia, FL has seen remarkable advancements, with innovative approaches such as employing graph neural networks within FL frameworks to tackle the time series graph problems prevalent in various business environments [27]. Furthermore, in the realm of the Internet of Things (IoT), which targets clients with heterogeneous data, collaborations between clients are analyzed to derive the most effective model using rating-based feedback mechanisms [28]. Additionally, a self-supervised pre-training paradigm has been introduced in self-supervised FL, leveraging transformers to efficiently address image processing challenges in the medical field [29]. Moreover, the utilization of self-supervised FL has been instrumental in addressing standard acoustic event classification [30] and facilitating the targeted training of video data generated on edge devices [31].

In broad terms, self-supervised FL has demonstrated effectiveness in domains such as image, audio, and video processing, yet solutions in the realm of tabular data remain scarce. To address this gap, we propose a self-supervised FL approach tailored to the tabular domain. Our solution involves leveraging segmented subsets for feature enhancement to bolster the performance of self-supervised models. By doing so, we aim to mitigate the scarcity of high-quality data for efficient data annotation within privacy-sensitive environments.

2.2. Self-Supervised Learning

Self-supervised learning has garnered significant attention due to its ability to mitigate the labeling costs associated with large datasets. This learning paradigm leverages pseudo-labels as learning cues, and the feature representations learned through this approach are versatile and applicable across a diverse array of tasks. As a result, self-supervised learning has emerged as a mainstream method in fields like computer vision and natural language processing [1,32].

Since the advent of Generative Adversarial Networks (GANs) [33], there has been a surge in generative modeling, giving rise to various architectures such as CycleGAN [34], StyleGAN [35], PixelRNN [36], Text2Image [37], and DiscoGAN [38], among others. These advancements have encouraged researchers to leverage unlabeled data more effectively for model training in the realm of self-supervised learning. However, despite their achievements, GAN-based methods have encountered challenges in training. They often struggle to converge, with parameters oscillating and proving difficult to stabilize. Alternatively,

the discriminators in these models may perform too well, resulting in generative models that fail to produce plausible artifacts, thereby hindering the learning process.

To tackle these challenges, recent studies have identified effective strategies through innovations in both data augmentation and pre-training tasks. In the realm of data augmentation, novel approaches such as self-distillation with masked input views and transformer-based architectures have been proposed. These methods aim to predict the hidden representations of complete input data [7]. Additionally, stochastic data augmentation modules have been introduced, which generate multiple correlated views of the data and apply random cropping, resizing, color distortion, and Gaussian blurring [9]. Regarding pre-training task generation, advancements such as BERT (Bidirectional Encoder Representations from Transformers) have emerged. Unlike traditional one-way language models, BERT employs a masked language model to pre-train deep bidirectional representations, enabling adaptation to a broader range of scenarios without the need for specific modifications for each situation [11].

The aforementioned methods have proven effective in natural language, audio, and image domains under certain conditions. However, in the domain of tabular data, training outcomes may be suboptimal due to the absence of the spatial, temporal, and semantic structures inherent in tables. This limitation makes it challenging to efficiently augment the data and generate pre-training tasks. To address this challenge, recent approaches have proposed utilizing flow blending for data augmentation in tabular data. Additionally, schemes such as noise injection and averaging error loss have been employed to enhance the performance of self-supervised models [39]. VIME extends the success of self-supervised and semi-supervised learning to the domain of tables [40]. Furthermore, an alternative approach involves leveraging unimproved self-supervised representation learning techniques, which utilize stochastic regularization methods independent of negative pairs. This strategy aims to capture the highly heterogeneous and unstructured information present in tabular data [41]. However, it is important to note that such reconstruction strategies may overlook specific errors within individual datasets, thereby diminishing the effectiveness of self-supervision.

Henceforth, we propose a novel framework utilizing FL to address privacy concerns while enabling joint model training, thereby enhancing data security and the efficacy of model utilization. Through FL, customers' data privacy is safeguarded, allowing for collaborative model training, which shifts the focus toward improving the model's performance. This approach effectively reframes the privacy challenge into a data labeling opportunity, thereby ameliorating the issue. Departing from traditional data augmentation schemes in self-supervised learning, our framework involves segmenting data using noise to augment data subsets. Simultaneously, we leverage data with uniform feature distributions to generate high-quality data, thereby augmenting the dataset and enhancing the labeling effect. This approach aims to bolster the effectiveness of data labeling efforts.

3. Overall Framework

In our approach, we simulate joint training involving multiple clients, all of whom are presumed to be honest and trustworthy, and share common data features. Each client trains its own self-supervised model using its labeled data. During the model learning process, the local model of each client is uploaded to the server. The server aggregates and optimizes the local models uploaded by all clients to create a global model, which is then sent back to the clients for the further training of their local models. This iterative process continues until the best model is trained. Through this learning method, we effectively mitigate the learning bottleneck associated with small data volumes, ensuring that each client obtains an improved model.

Figure 1 gives the framework, in which we designed multiple clients (Client) and a server (Server) in a single client in the enterprise (Data Enterprise) according to the data processing, divided into labeled data (Labeled Data) and unlabeled data (Unlabeled Data). It also includes the Encoder and Decoder in the process of self-supervision learning. The

global model download and local model upload between the Client and Server are used to communicate the training process to achieve knowledge sharing, and the local model in Client can generate labels for Unlabeled Data to become Labeled Data.

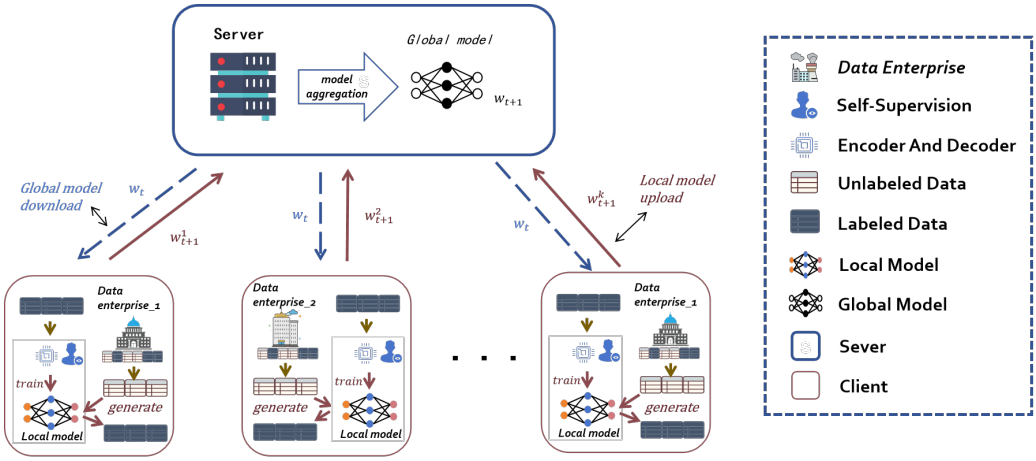

Figure 1. An overall introduction to the TabFedSL framework, including Data Enterprise, Self-Supervision, Encoder and Decoder, Unlabeled Data, Labeled Data, Local Model, Global Model, Server, Client.

In addition, we also simulate the case of non-IID data in real scenarios, where the bottleneck of the individual client training is greatly suppressed in the case of extremely unbalanced data, but in the case of only one type of data or two types of data in our design, our approach can fully validate the effectiveness of FL.

We use a self-supervised learning method for each client's unlabeled data, as shown in Figure 2, to divide the tabular data into multiple subsets, while using the same encoder for each subset to ensure that each subset receives the same parameter sharing and, at the same time, reconstructing all the features from the subset of the features to generate the complete data table. We selectively generate losses suitable for the target additions by combining the projection pairs in their own way, and we reduce the distance between projection pairs via the mean squared error (MSE).

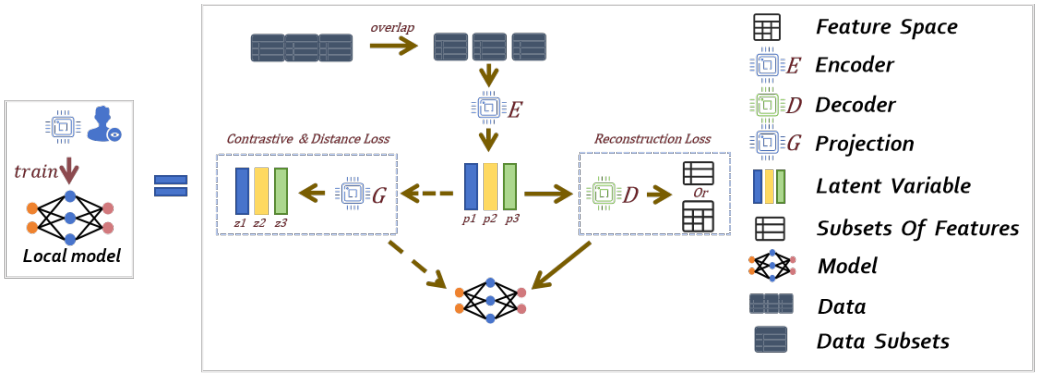

Figure 2. Self-supervised training process where p and z are the feature projections, including Feature Space, Encoder, Decoder, Projection, Latent Variable, Subsets of Features, Model, Data, Data Subsets.

4. Methods

A great deal of recent success in machine learning has been based on optimization by means of stochastic gradient descent (SGD). Many of these optimizations can be understood as adjustments to the structure of the model as well as the loss function to achieve optimization. At the same time, by dividing the features of the dataset into multiple subsets of features, the table data representation learning task is transformed into a multi-view representation learning task, and the features of the data are redefined. Subsetting the

features of tabular data and solving the problem of treating all features equally as well as solving the problem of tabular data not being applicable in high-latitude scenarios [42], we are inspired by this and naturally propose a method of self-supervised FL of tabular data in combination with high-latitude hidden scenarios such as banks, insurance, and hospitals.

4.1. Add Noise Strategy

Our approach is to perform data augmentation on tabular data, so we try to augment the data by adding one of three types of noise: (i) Gaussian noise, (ii) swap noise, with features randomly selected from the same column method swapped with features of other entries, and (iii) randomly selected entries with their feature size set to 0 to become zero-out.

We create a binomial mask m and a noise matrix s_c with the same shape as the subset s, where $\circ$ is the Hadamard product. After data augmentation, the subset s is corrupted and a new subset s_{1e} is generated as follows:

$$s_{1e} = (1 - m) \circ s_1 + m \circ s_c \,. \tag{1}$$

4.2. Training Strategies

Our overall objective function is

$$l_{f_i}(w) : \mathcal{L}_t = \mathcal{L}_r + \mathcal{L}_c + \mathcal{L}_d \,. \tag{2}$$

where L_t, L_r, L_c and L_d are the total loss, reconstruction loss, contrast loss, and distance loss, respectively. The total loss of FL is taken as the loss of each client and the average $l_{f_i}(w)$.

4.2.1. Reconstruction Loss

For each defined subset, we can reconstruct the new subset. By comparing the original subset and the new subset, the computational MSE of the reconstructed new subset can be calculated. The reconstruction loss formula is shown below:

$$\mathcal{L}_r = \frac{1}{K} \sum_{k=1}^{K} l_k, \text{ where } l_k = \frac{1}{N} \sum_{i=1}^{N} \left(S^{(i)} - \hat{S}_k^{(i)} \right)^2. \tag{3}$$

where K is the total number of subsets, N is the batch size, l_k is the reconstruction loss of the kth subset, L_r is the average of the reconstruction loss of all subsets, and S is the overall dataset.

4.2.2. Contrastive Loss

When the dataset has many categories, the chance of randomly selecting negative samples is high, and we can use the projection network to get the projection z. If the subsets of z_1 and z_2 samples are more effective for the training result, then the remaining samples can be negative for the training result. For the 3 feature subsets $\{s_1, s_2, s_3\}$, we can compute the loss between any 2 subsets to form the set $T = \{z_1, z_3\}, \{z_1, z_3\}, \{z_2, z_3\}$, for a total of 3 pairs of contrast losses. The overall contrastive loss is

$$\mathcal{L}_d = \frac{1}{J} \sum_{\{z_a, z_b\} \in T} p(z_a, z_b), \text{ where } p(z_a, z_b) = \frac{1}{2N} \sum_{i=1}^{N} \left[l(z_a^{(i)}, z_b^{(i)}) + l(z_b^{(i)}, z_a^{(i)}) \right]. \tag{4}$$

$$l(z_a^{(i)}, z_b^{(i)}) = -\log \frac{\exp(\text{sim}(z_a^{(i)}, z_b^{(i)})/\tau)}{\sum_{k=1}^{N} 1_{k \neq i} \exp(\text{sim}(z_a^{(i)}, z_b^{(k)})/\tau)}.$$

where J is the total number of pairs in the set T, $p(z_a, z_b)$ is the total contrast loss of a pair of projections, $l(z_a, z_b)$ are the loss functions of the positive pairs of examples $\{z_a(i), z_b(i)\}$ in the corresponding subset, and L_c is the average of the contrast losses of all pairs.

4.2.3. Distance Loss

In order to make the paired samples in the dataset more closely matched, we can introduce the MSE as part of the loss as a measure of the similarity between samples within a subset. Accordingly, we were able to estimate a composite MSE loss, which improves the fit of the model to the data:

$$\mathcal{L}_d = \frac{1}{J} \sum_{\{z_a, z_b\} \in T} p(z_a, z_b), \text{ where } p(z_a, z_b) = \frac{1}{N} \sum_{i=1}^{N} \left(z_a^{(i)} - z_b^{(i)} \right)^2. \tag{5}$$

4.2.4. FedAvg

Above is the formula representation of the relevant loss function that we use in the client side in FL, while in the server side we refer to FedAvg for the summation and averaging algorithm of the parameter list uploaded by all the clients, which is again sent down to the clients for training to complete the gradient convergence.

Client local update: In each iteration t, the server sends the current global model parameters $\mathbf{w}_t$ to a selected set of clients. Each client k updates the model using its local data S_k.

$$\mathbf{w}_{t+1}^i = \mathbf{w}_t - \eta \nabla L_{f_i}(\mathbf{w}_t).$$

where $\nabla L_{f_i}(\mathbf{w}_t)$ is the gradient of the model parameter $\mathbf{w}_t$ computed by client k on its data, and η is the learning rate.

Global model update: The server computes a weighted average of the updates sent back by all clients to update the global model.

$$\mathbf{w}_{t+1} = \frac{\sum_{k=1}^{K} n_k \mathbf{w}_{t+1}^k}{n}.$$

where n_k is the number of data points for client k, and n is the total number of data points for all clients.

4.3. Pseudocode

The algorithm orchestrates interactions between a central server and multiple client nodes to facilitate the training of a global model while preserving the privacy of raw data stored on the clients. The server-side procedure, delineated in Algorithm 1, entails selecting clients, transmitting the current model weights to these selected clients, receiving their updated weights after local training, and subsequently aggregating these weights to update the global model through weighted averaging. This iterative process persists until convergence of the global model is achieved. Conversely, the client-side algorithm, as depicted in Algorithm 2, is executed locally on each client node. Here, the client receives the current global model weights from the server, conducts local training using these weights, and forwards the updated weights back to the server. Within each local training epoch, the client partitions its data into small batches, computes the gradient based on the loss function, and iteratively updates the model weights accordingly. By employing this approach, Federated Learning enables collaborative model training utilizing distributed computational resources while safeguarding the confidentiality of individual client data.

Algorithm 1 Federated Learning Server-Side Algorithm

Input: Number of clients K
Initialize server model weights w_0
Set round $t = 0$
Server executes:
while not converged **do**
    $t \leftarrow t + 1$
    Initialize an array: $client\_updates \leftarrow []$
    for each client k from 1 to K **do**
        Send current model weights w_{t-1} to client k
        Receive updated weights w_{t_k} from client k
        $client\_updates[k] \leftarrow w_{t_k}$
    end for
    $w_t \leftarrow \text{average}(client\_updates)$
    for each client k from 1 to K **do**
        Send global model weights w_t to client k
    end for
end while
return w_t

Algorithm 2 Federated Averaging Client-Side Algorithm

Input: client_data, initial_server_weights, batch_size B, local_epochs E, learning_rate η
Initialization: Receive initial weights w_0 from server
Set weights w to w_0
Initialize total reconstruction loss L_{recon} to 0
Initialize total contrastive loss $L_{contrast}$ to 0
Initialize total distance loss $L_{distance}$ to 0
Client executes:
for each local epoch i from 1 to E **do**
    Receive updated weights w from the server
    Divide client_data into batches of size B
    for each batch X in client_data **do**
        Perform a forward pass to get latent representations z
        Compute reconstruction loss L_{recon} using S and its reconstruction
        $L_{batch} = L_{recon}$
        if Apply contrastive loss **then**
            Compute contrastive loss $L_{contrast}$ for all pairs in z
            $L_{batch} = L_{batch} + L_{contrast}$
        end if
        if Apply distance loss **then**
            Compute distance loss $L_{distance}$ for all pairs in z
            $L_{batch} = L_{batch} + L_{distance}$
        end if
        Calculate gradients of L_{batch} with respect to w
        Update weights $w = w - \eta \times$ gradients
    end for
    Send updated weights w to the server
end for
return w

5. Experiments

In our experimental evaluation, we focused on tabular data, utilizing datasets such as Mnist, UCI Adult Income, and UCI BlogFeedback in tabular format to assess the effectiveness of the TabFedSL framework. We conducted several comparative experiments within the TabFedSL framework: (i) We compared the efficacy of the TabFedSL framework in

training a self-supervised model against training a joint self-supervised model, maintaining consistency in the number of training rounds, dataset characteristics, noise conditions, etc.; (ii) We evaluated the robustness of our framework in handling real-world data imbalances, and we simulated scenarios of data imbalance by assigning different classes to each client individually. This simulation enabled us to validate the effectiveness of the TabFedSL framework in addressing data imbalances. (iii) We conducted a hyperparameter analysis focusing on the number of client aggregation rounds and the number of clients in the FL setup of TabFedSL. This analysis aimed to showcase the robustness of our approach across different hyperparameter settings. Through these experiments, we aimed to provide comprehensive insights into the performance and effectiveness of the TabFedSL framework in addressing key challenges in tabular data analysis and model training.

5.1. Experimental Platforms

The experimental environment for this study was as follows: an Intel Xeon Silver 4216 CPU @ 2.10 GHz with 16 cores and 32 threads, accompanied by an NVIDIA Tesla T4 GPU produced from H3C Group, Beijing, China. The entire setup was hosted on a server with the following specifications. CPU: Intel Xeon Silver 4216 @ 2.10 GHz produced from H3C Group, Beijing, China (16 cores, 32 threads); Memory: 64 GB RAM produced from H3C Group, Beijing, China; GPU: NVIDIA Tesla T4 with 16 GB of memory; Operating System: Linux x86_64 ubuntu 22.04 (specific distribution not mentioned); Deep Learning Framework: PyTorch 1.13.1; CUDA Version: 12.0; Python Version: 3.7.12; NVIDIA Driver Version: 525.125.06.

The machine used for the experiments was equipped with an extensive 64 GB of RAM and a total of 16 GB of GPU memory. The system was operated under a Linux environment, with PyTorch leveraging CUDA 12.0 for GPU-accelerated deep learning tasks.

5.2. Dataset Description

MNIST: In our experimental setup, we employed the flattening technique to convert each 28 $\times$ 28 image into a one-dimensional vector, followed by resizing the images to 255 pixels to normalize the data, as described in [43]. We utilized the entire training dataset for model training, while the test set was reserved for model evaluation.

UCI Adult Income: Derived from the 1994 U.S. Census database [44], the UCI Adult Income dataset is renowned for its utility in predicting whether an individual's annual income exceeds $50,000. Comprising 6 continuous and 8 categorical variables, the dataset undergoes one-hot encoding of categorical variables, resulting in an expansion of the feature space to include 101 distinct features.

UCI BlogFeedback: This data originally came from blog posts and was used to perform a regression task to predict the number of comments in the next 24 h [45]. Now, we convert this data into a binary classification task that determines whether an article has comments or not. These datasets contain 280 integer and real features, and separate training and test sets are provided.

5.3. Experimental Setup

In the experimental design of this study, we took 20 rounds of training cycles as a benchmark and set the learning rate (lr) to 0.001. To monitor the performance of the model during the training process, we computed the validation loss at the end of each epoch round. In addition, to simulate the regularization effect during the training process, we set the Dropout rate to 0.2 and set the tau value to 0.1 to control the strength of the regularization. As the base model for the classification task, we chose the logistic regression model. The number of clients set in the experiment was 10, while the number of servers was fixed at 1 to construct a distributed learning environment.

For the non-independent identical distribution (Non-IID) property of the data, we simulated the inhomogeneity of the data distribution in the real world through the skewness of the label distribution.The MNIST dataset was used for the decimation task, while the

Income and Blog datasets were used for the bicategorization task, whereby our dataset is partitioned into 10 and 2 subsets according to the skewness of the label distribution. To further model the label imbalance, we conducted independent experiments using each client's dataset with the inclusion of 2 and 10 clients, respectively.

For the independent identically distributed (IID) nature of the data, we completely disrupted the dataset to ensure that each client's training set contains multiple species in distributed training, as a way to explore the differences in model performance under different data distribution settings.

5.4. Evaluation Indicators

After training our federated self-supervised model, we proceeded to generate a training set comprising logistic regression models based on this model. Subsequently, we trained a logistic regression model to assess the quality of the generated data. We then evaluated the performance of the trained logistic regression model using the test set. In the case of TabFedSL, we specifically employed Accuracy as the evaluation metric in logistic regression to gauge the effectiveness of the model.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{6}$$

In the context of classification models, TP (True Positives) represents the number of instances correctly predicted as positive by the model. TN (True Negatives) indicates the number of instances correctly predicted as negative by the model. FP (False Positives) denotes the number of instances incorrectly predicted as positive by the model. FN (False Negatives) signifies the number of instances incorrectly predicted as negative by the model. These metrics are fundamental in evaluating the performance and accuracy of classification models.

6. Results

In the experimental chapters, we categorize our experiments into two main classes based on the type of data: one is the performance of IID data in our proposed framework; the other is the performance demonstration of non-IID data in our framework. For each type of data, we further conduct a comparative study between centralized and distributed self-supervised learning. In particular, in the distributed learning scenario with non-IID data, we compare, in detail, the performance difference between client-side training alone and server-side distributed training.

6.1. IID Environment Setting

As shown in Table 1, we have selected the centralized self-supervised learning and distributed self-supervised learning of VIME as a benchmark for comparison to demonstrate that our method can effectively improve the training of the model's performance in self-supervised learning and distributed self-supervised learning by introducing a noise strategy in a subset of the segmented data under the condition of IID.

Figure 3 illustrates the comparison of the accuracy of VIME and Ours under focused learning in the IID case.

Figure 4 illustrates the accuracy comparison between VIME and Ours under distributed learning in the IID case.

Experimental results show that in both centralized and distributed learning environments, our framework is able to significantly improve the data quality through the strategy of segmenting and adding noise to a subset of the data, thus effectively enhancing the model's performance.

In our framework, the difference between centralized self-supervised learning and distributed self-supervised learning under different noise conditions is compared, and the results are shown in the table below. Table 2 shows that in the IID case, our framework is able to achieve the performance level of centralized learning training under distributed training.

Table 1. Comparison of the accuracy of centralized and distributed learning in the IID case.

Dataset	Model	Training Type	Accuracy
MNIST	VIME	Centralized	0.9370
		Distributed	0.9246
	Ours	Centralized	**0.9749**
		Distributed	**0.9755**
Income	VIME	Centralized	0.7543
		Distributed	0.7542
	Ours	Centralized	**0.8509**
		Distributed	**0.8509**
Blog	VIME	Centralized	0.7019
		Distributed	0.7019
	Ours	Centralized	**0.8424**
		Distributed	**0.8452**

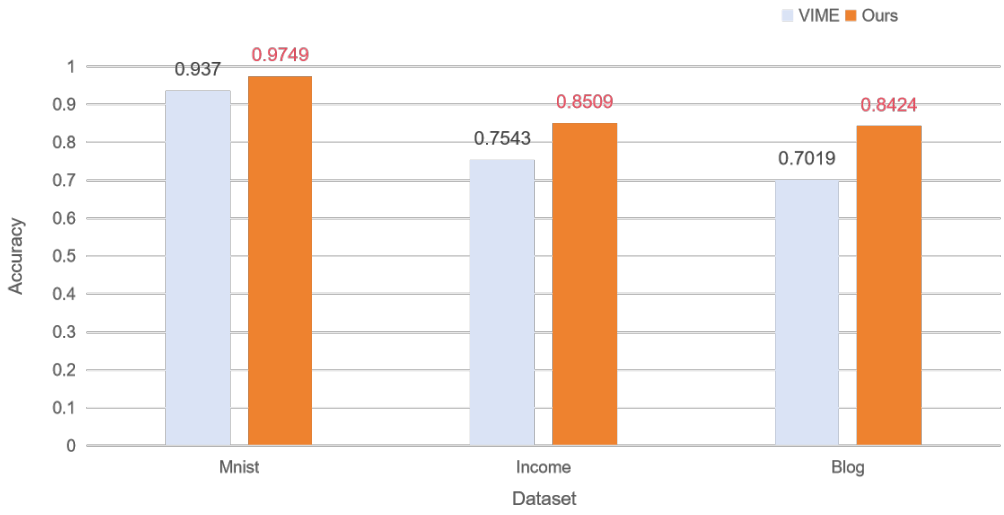

Figure 3. Comparison of the accuracy of VIME and Ours centralized learning in the IID case.

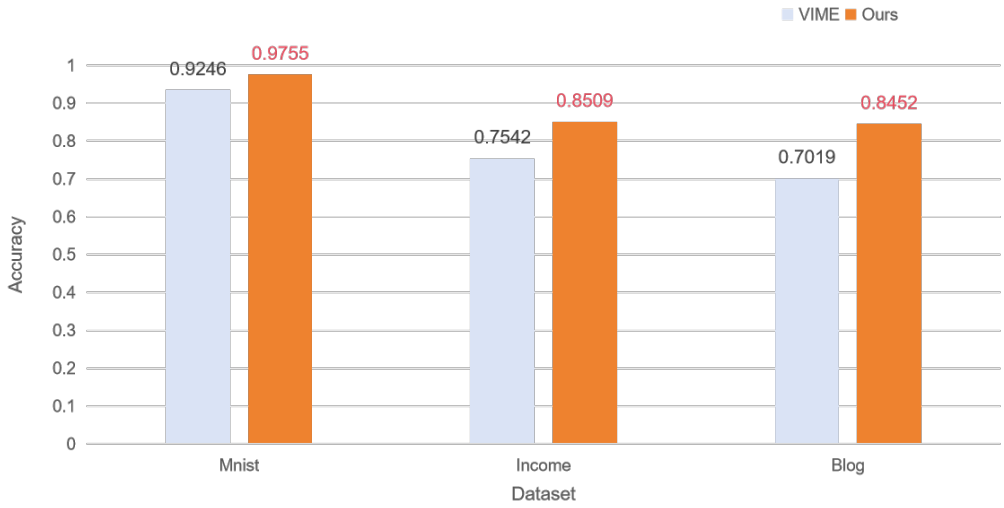

Figure 4. Comparison of the accuracy of VIME and Ours distributed learning in the IID case.

Table 2. Comparison of the accuracy of different noises in centralized and distributed learning in the IID case.

Dataset	Training Type	Noise	Accuracy
MNIST	Centralized	no_noise	0.9749
		gaussian_noise	0.9751
		swap_noise	0.9820
		zero_out	0.975
	Distributed	no_noise	**0.9755**
		gaussian_noise	**0.9804**
		swap_noise	**0.9826**
		zero_out	**0.9794**
Income	Centralized	no_noise	0.8509
		gaussian_noise	0.8531
		swap_noise	0.8513
		zero_out	0.8517
	Distributed	no_noise	**0.8509**
		gaussian_noise	**0.8534**
		swap_noise	**0.8533**
		zero_out	**0.8530**
Blog	Centralized	no_noise	0.8424
		gaussian_noise	0.8457
		swap_noise	0.8429
		zero_out	0.8425
	Distributed	no_noise	**0.8431**
		gaussian_noise	**0.8477**
		swap_noise	**0.8440**
		zero_out	**0.8461**

Figure 5 shows the performance of centralized and distributed training with no added noise.

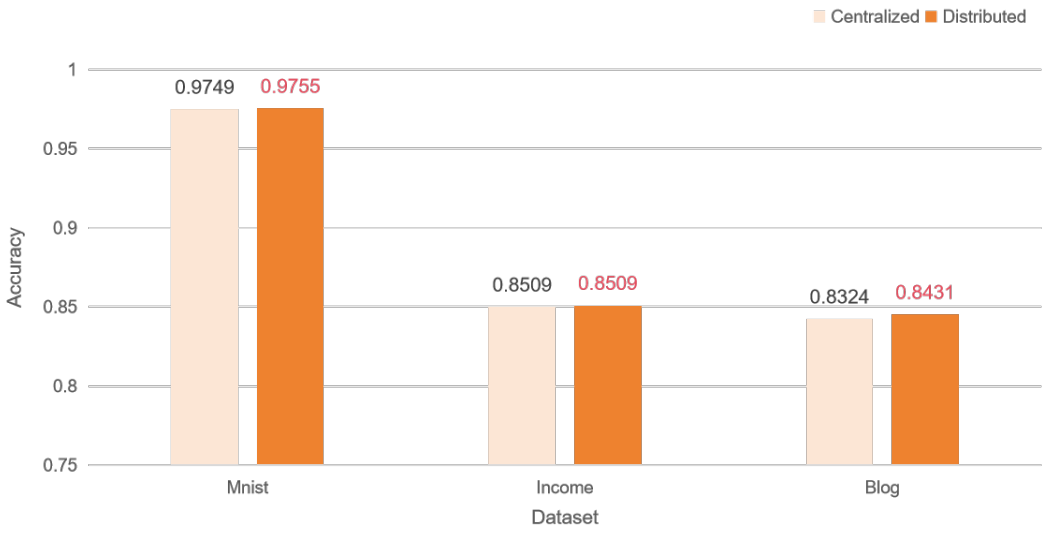

Figure 5. Comparison of the accuracy of no_noise in the IID case for centralized and distributed learning.

Figure 6 shows the performance of centralized and distributed training after adding gaussian_noise.

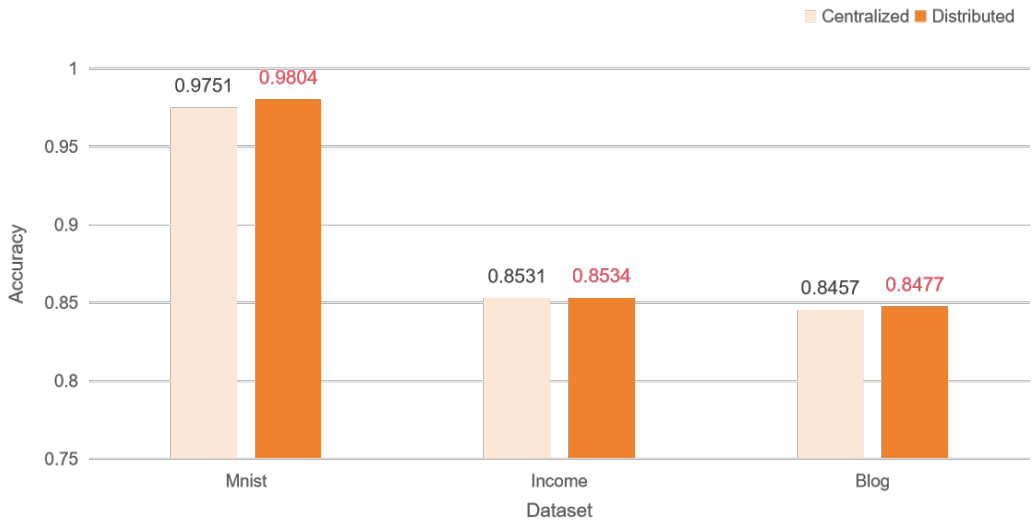

Figure 6. Comparison of the accuracy of gaussian_noise in the IID case for centralized and distributed learning.

Figure 7 shows the performance of centralized and distributed training after adding swap_noise.

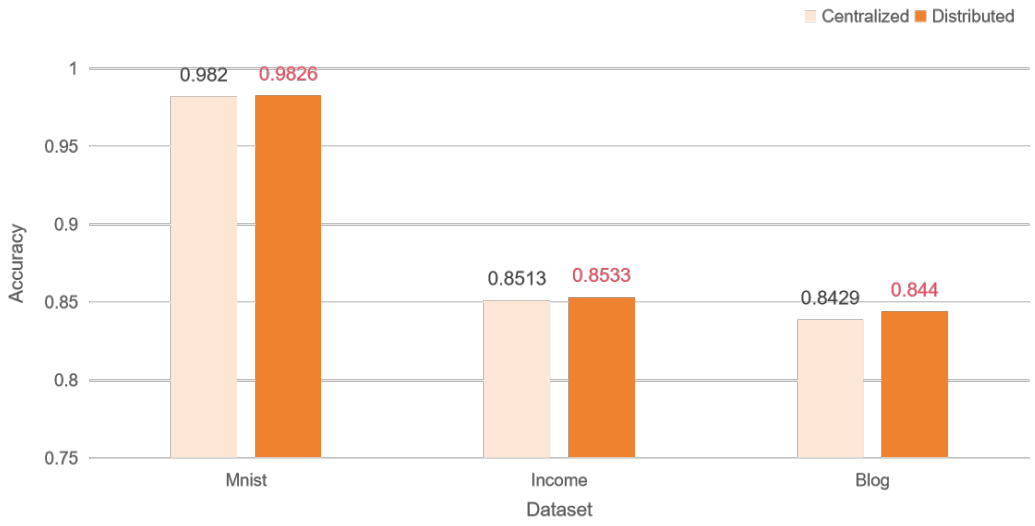

Figure 7. Comparison of the accuracy of swap_noise in the IID case for centralized and distributed learning.

Figure 8 shows the performance of centralized and distributed training after adding zero_out.

The implementation results show that in the IID case, both with and without added noise, our distributed learning results reach the performance level of models trained with centralized learning results.

The following Figures 9 and 10 clearly show the impact of adding noise and not adding noise on the performance of our framework in both distributed and centralized learning environments. The results in the figures demonstrate that the model trained by adding noise outperforms the model without added noise.

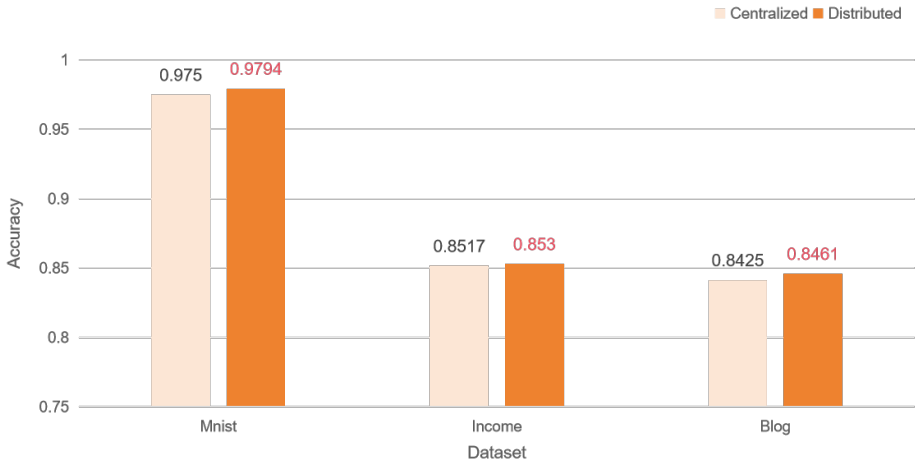

Figure 8. Comparison of the accuracy of zero_out in the IID case for centralized and distributed learning.

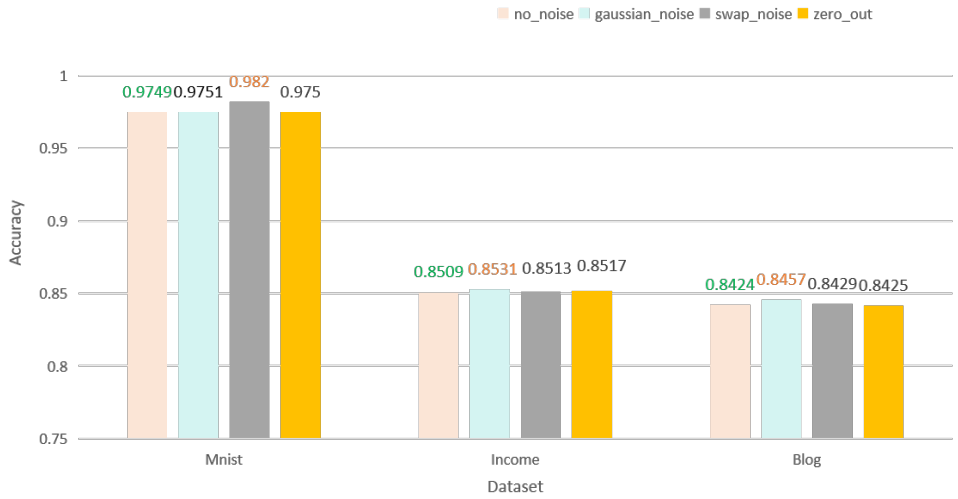

Figure 9. Comparison of the accuracy of centralized learning with additive noise versus no additive noise in the IID case.

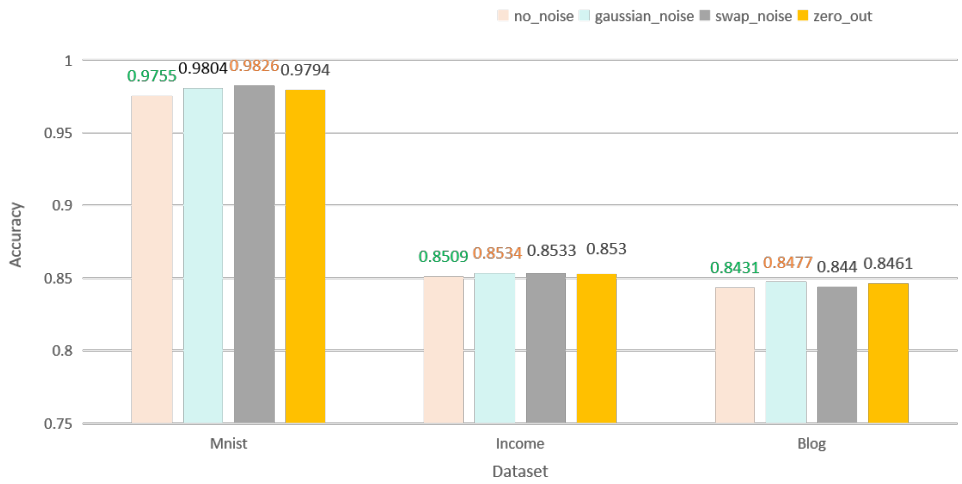

Figure 10. Comparison of the accuracy of distributed learning with additive noise versus no additive noise in the IID case.

6.2. Non-IID Environment Setting

As shown in Table 3, we have selected the centralized self-supervised learning and distributed self-supervised learning of VIME as a benchmark for comparison to demonstrate that our method can effectively improve the model training performance of self-supervised learning and distributed self-supervised learning under non-IID conditions by introducing a noise strategy in the subset of the segmented data.

Table 3. Comparison of the accuracy of different noises in centralized and distributed learning in the non-IID case.

Dataset	Model	Training Type	Accuracy
MNIST	VIME	Centralized	0.9224
		Distributed	0.9239
	Ours	Centralized	**0.9749**
		Distributed	**0.9767**
Income	VIME	Centralized	0.7543
		Distributed	0.7542
	Ours	Centralized	**0.8509**
		Distributed	**0.852**
Blog	VIME	Centralized	0.7019
		Distributed	0.7018
	Ours	Centralized	**0.8424**
		Distributed	**0.8443**

Figure 11 illustrates the accuracy comparison between VIME and Ours under centralized learning in the non-IID case.

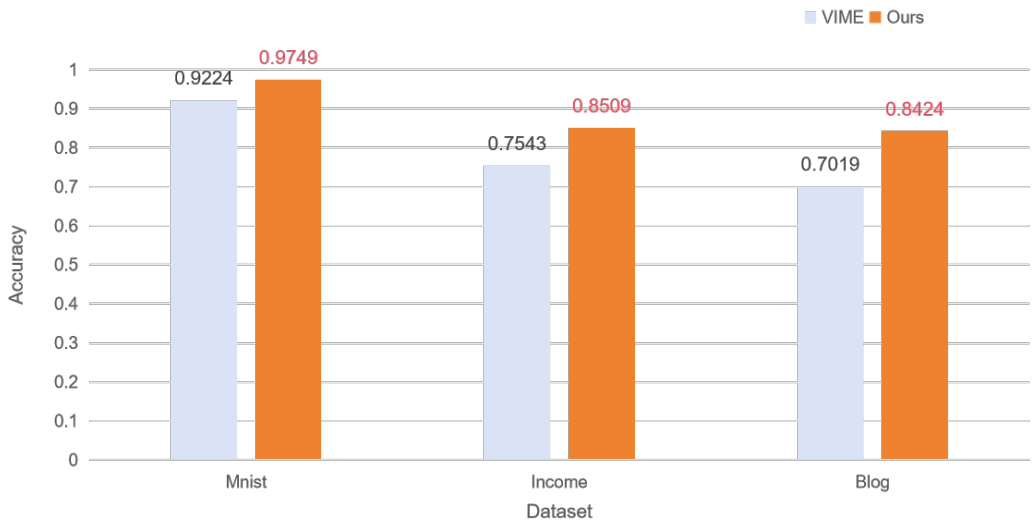

Figure 11. Comparison of the accuracy of VIME and Ours centralized learning in the non-IID case.

Figure 12 illustrates the accuracy comparison between VIME and Ours under distributed learning in the non-IID case.

The experimental results show that our proposed framework is able to significantly improve the quality of data through data subset partitioning and noise addition strategies in both centralized and distributed learning environments. This improvement in data quality in turn effectively enhances the performance of the model. This finding highlights the importance of well-designed data preprocessing strategies for optimizing the model training process and improving learning efficiency in self-supervised learning and its distributed variants.

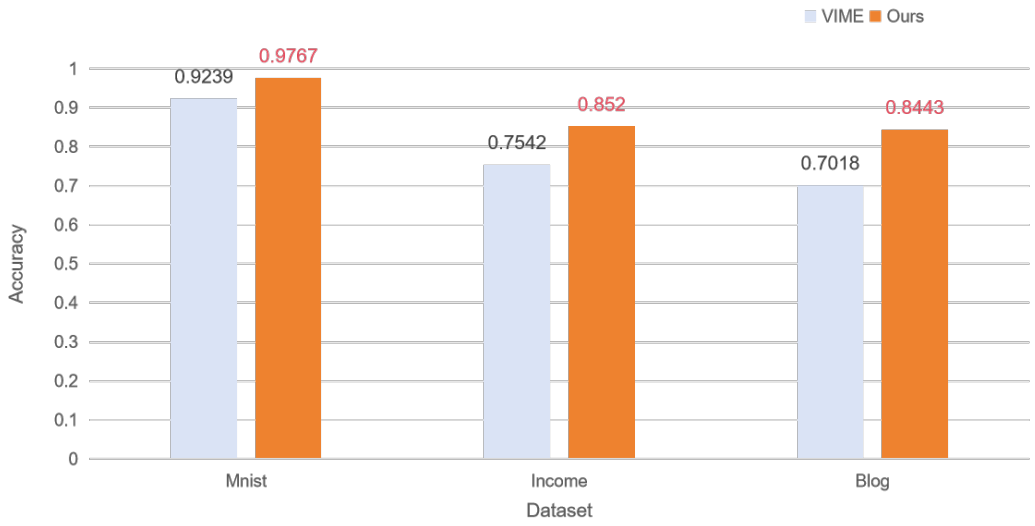

Figure 12. Comparison of the accuracy of VIME and Ours distributed learning in the non-IID case.

In this study, we selected three different datasets and trained the data used by each client individually to evaluate the accuracy of the model and compare these results with the results of our distributed experiments. The aim of this exercise is to simulate the data label skew problem, which is common in real-world scenarios, as a way to validate the applicability and effectiveness of our proposed framework in dealing with the specific context of data imbalance. Through this approach, we are able to deeply analyze the performance difference between independently trained and centrally trained models across clients in a distributed learning environment under label skew conditions, thus demonstrating the advantages and applicability of our framework in dealing with such unbalanced data distribution scenarios. The results on the MNIST, Income, and Blog datasets are shown below in Figures 13–15.

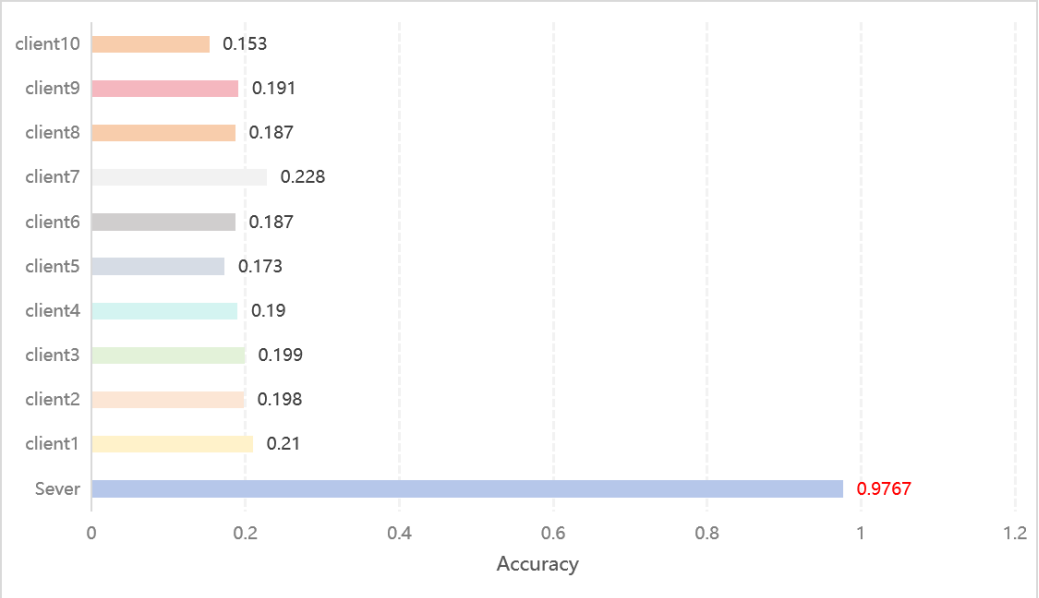

Figure 13. Comparison of the accuracy of client data trained alone and through server-side co-training on the MNIST dataset.

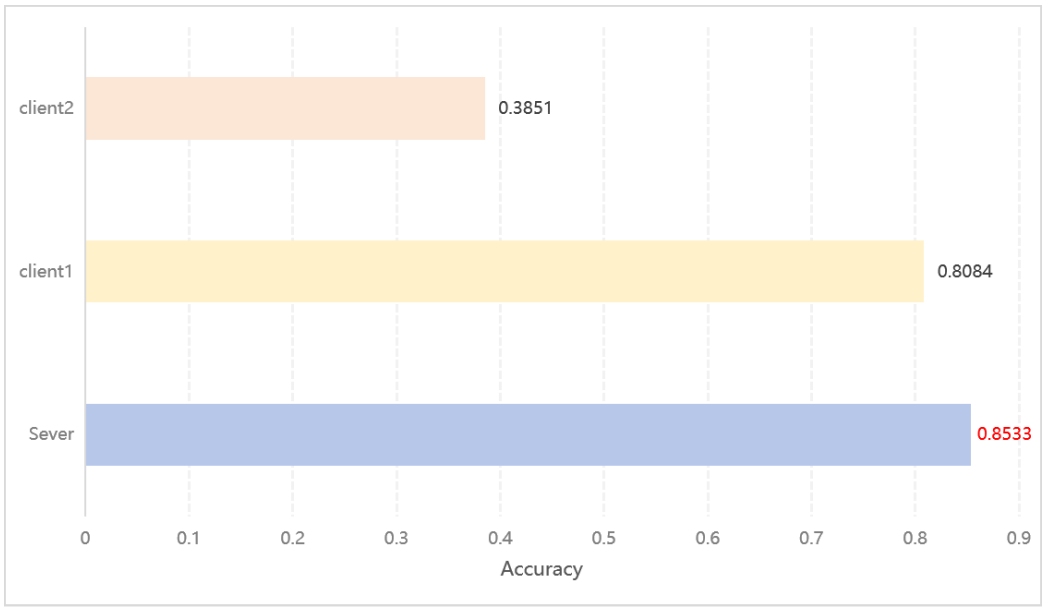

Figure 14. Comparison of the accuracy of client data trained alone and through server-side co-training on the Income dataset.

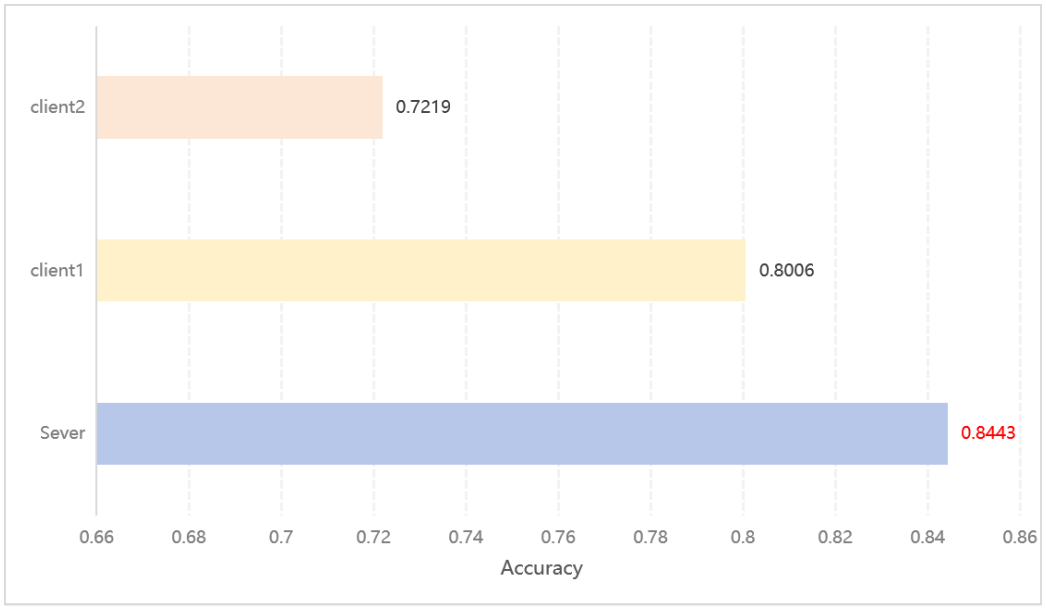

Figure 15. Comparison of the accuracy of client data trained alone and through server-side co-training on the Blog dataset.

Finally, we performed a comparative analysis, as shown in Figure 16, evaluating the performance of our proposed distributed training framework under two different data settings: non-IID and IID.

From the provided graphs of the results, we can clearly observe that the model training is better in the IID data setting than under the non-IID condition. In addition, the accuracy of the server-side model suffers a significant loss in the IID setting. It highlights the important impact of data distribution characteristics on model training performance, especially in distributed learning environments, where the independent, identically distributed nature of the data helps to improve the generalization ability and accuracy of the model.

The experimental results show that our framework achieves a level of performance that matches centralized learning in distributed learning environments, a result that validates the effectiveness of our approach in coping with limited data annotation and distributed

learning scenarios. These findings provide new perspectives in the field of distributed learning, emphasizing the critical role of data augmentation in improving the effectiveness of distributed learning.

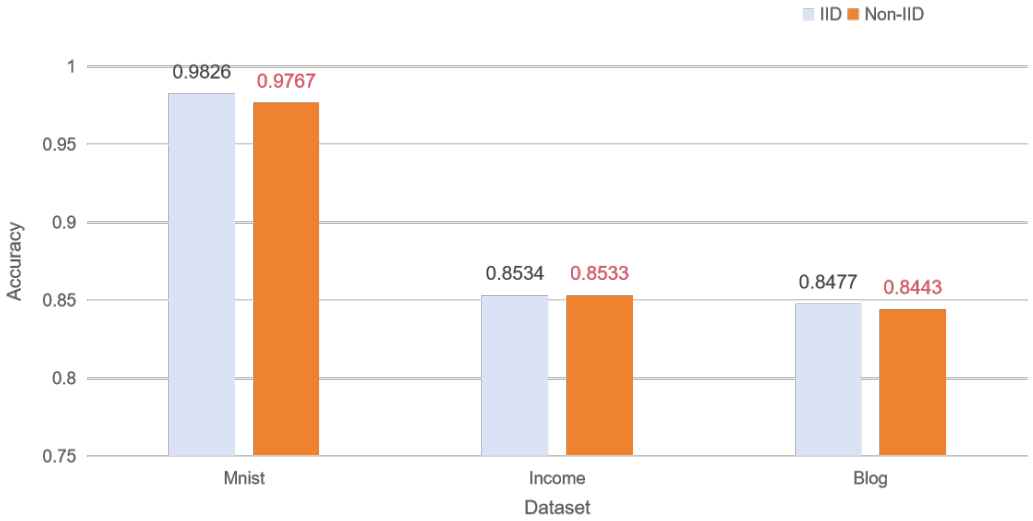

Figure 16. Comparison of the accuracy of non-IID and IID in a distributed environment.

7. Conclusions

In limited data annotation and distributed environments, although existing work can solve the problems of small data volume and difficult annotation to a certain extent, there is still a gap with the centralized learning method. Our framework performs data augmentation by adding noise on different subsets, resulting in high-quality data annotations. The experimental results show that in distributed scenarios, the performance level of our framework is consistent with that of centralized learning. We demonstrate its effectiveness across the MNIST, UCI Adult Income, and UCI BlogFeedback datasets, with validation conducted in a simulated non-IID scenario. Additionally, our current study will provide a kind of baseline for subsequent applications of self-supervised FL in the field of forms. Within our framework, model performance enhancement primarily stems from three components. (i) Self-Supervised Learning: Our model utilizes self-supervised learning to tackle challenges relating to data quality and quantity. It accomplishes this by converting the single-view problem into a multi-view problem. Through the generation of reconstructed data via feature subsets, self-supervised learning substantially improves both data quality and quantity. (ii) Refined Data Augmentation: We improve traditional self-supervised learning data augmentation by partitioning datasets and adding noise to augment subset data. This process further enhances the model's performance. (iii) Federated Learning: FL plays a pivotal role in enhancing model performance by facilitating the sharing of high-quality data across multiple nodes in real-world non-IID scenarios.

We provide a solution to the tabular data labeling challenge. By augmenting tabular data, we enhance the self-supervised model's performance, integrating self-supervised learning with FL. We intend to employ this framework in handling structured tasks in our forthcoming endeavors.

Author Contributions: Conceptualization, J.G., Y.H. and R.W.; methodology, Z.C., Y.H. and R.W.; software, Z.C., Y.H. and R.W.; validation, J.G., Y.H. and R.W.; formal analysis, G.L., Y.H. and R.W.; investigation, J.G. and G.L.; resources, R.W., Y.H. and G.L.; data curation, Y.H. and R.W.; writing—original draft preparation, R.W. and Y.H.; writing—review and editing, R.W., Z.C. and Y.H.; visualization, R.W. and Y.H.; supervision, J.G. and G.L.; project administration, J.G.; funding acquisition, J.G. All authors have read and agreed to the published version of the manuscript.

Funding: This study was funded by the Scientific Research Project of Jilin Provincial Education Department (JJKH20230764KJ).

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: We want to thank "Changchun Computing Center" and "Eco-Innovation Center" for providing the inclusive computing power and technical support of MindSpore during the completion of this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Jaiswal, A.; Babu, A.R.; Zadeh, M.Z.; Banerjee, D.; Makedon, F. A survey on contrastive self-supervised learning. *Technologies* **2020**, *9*, 2. [CrossRef]
2. Schiappa, M.C.; Rawat, Y.S.; Shah, M. Self-supervised learning for videos: A survey. *ACM Comput. Surv.* **2023**, *55*, 1–37. [CrossRef]
3. Misra, I.; Maaten, L.v.d. Self-supervised learning of pretext-invariant representations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 6707–6717.
4. Liu, X.; Zhang, F.; Hou, Z.; Mian, L.; Wang, Z.; Zhang, J.; Tang, J. Self-supervised learning: Generative or contrastive. *IEEE Trans. Knowl. Data Eng.* **2021**, *35*, 857–876. [CrossRef]
5. Hendrycks, D.; Mazeika, M.; Kadavath, S.; Song, D. Using self-supervised learning can improve model robustness and uncertainty. *Adv. Neural Inf. Process. Syst.* **2019**, *32*.
6. Zbontar, J.; Jing, L.; Misra, I.; LeCun, Y.; Deny, S. Barlow twins: Self-supervised learning via redundancy reduction. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 12310–12320.
7. Baevski, A.; Hsu, W.N.; Xu, Q.; Babu, A.; Gu, J.; Auli, M. Data2vec: A general framework for self-supervised learning in speech, vision and language. In Proceedings of the International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022; pp. 1298–1312.
8. Elnaggar, A.; Heinzinger, M.; Dallago, C.; Rehawi, G.; Wang, Y.; Jones, L.; Gibbs, T.; Feher, T.; Angerer, C.; Steinegger, M.; et al. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 7112–7127. [CrossRef] [PubMed]
9. Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G. A simple framework for contrastive learning of visual representations. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020; pp. 1597–1607.
10. Toering, M.; Gatopoulos, I.; Stol, M.; Hu, V.T. Self-supervised video representation learning with cross-stream prototypical contrasting. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 3–8 January 2022; pp. 108–118.
11. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
12. Oord, A.v.d.; Li, Y.; Vinyals, O. Representation learning with contrastive predictive coding. *arXiv* **2018**, arXiv:1807.03748.
13. Caron, M.; Misra, I.; Mairal, J.; Goyal, P.; Bojanowski, P.; Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 9912–9924.
14. He, K.; Fan, H.; Wu, Y.; Xie, S.; Girshick, R. Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 9729–9738.
15. Falcon, W.; Cho, K. A framework for contrastive self-supervised learning and designing a new approach. *arXiv* **2020**, arXiv:2009.00104.
16. Chen, X.; Fan, H.; Girshick, R.; He, K. Improved baselines with momentum contrastive learning. *arXiv* **2020**, arXiv:2003.04297.
17. Vincent, P.; Larochelle, H.; Bengio, Y.; Manzagol, P.A. Extracting and composing robust features with denoising autoencoders. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 June 2008; pp. 1096–1103.
18. Abdelhameed, S.A.; Moussa, S.M.; Khalifa, M.E. Enhanced additive noise approach for privacy-preserving tabular data publishing. In Proceedings of the 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, 5–7 December 2017; pp. 284–291.
19. Wen, J.; Zhang, Z.; Lan, Y.; Cui, Z.; Cai, J.; Zhang, W. A survey on federated learning: Challenges and applications. *Int. J. Mach. Learn. Cybern.* **2023**, *14*, 513–535. [CrossRef] [PubMed]
20. Kuze, N.; Ishikura, S.; Yagi, T.; Chiba, D.; Murata, M. Classification of diversified web crawler accesses inspired by biological adaptation. *Int. J. Bio-Inspired Comput.* **2021**, *17*, 165–173. [CrossRef]
21. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
22. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol. TIST* **2019**, *10*, 1–19. [CrossRef]
23. Wang, L.; Meng, Z.; Yang, L. A multi-layer two-dimensional convolutional neural network for sentiment analysis. *Int. J. Bio-Inspired Comput.* **2022**, *19*, 97–107. [CrossRef]

24. Liang, B.; Cai, J.; Yang, H. A new cell group clustering algorithm based on validation & correction mechanism. *Expert Syst. Appl.* **2022**, *193*, 116410.
25. Long, T.; Jia, Q.S. Matching uncertain renewable supply with electric vehicle charging demand—A bi-level event-based optimization method. *Complex Syst. Model. Simul.* **2021**, *1*, 33–44. [CrossRef]
26. Zhou, H.; Yang, G.; Dai, H.; Liu, G. PFLF: Privacy-preserving federated learning framework for edge computing. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 1905–1918. [CrossRef]
27. Xie, Z.; Huang, Y.; Yu, D.; Parizi, R.M.; Zheng, Y.; Pang, J. FedEE: A federated graph learning solution for extended enterprise collaboration. *IEEE Trans. Ind. Inform.* **2022**, *19*, 8061–8071. [CrossRef]
28. Pang, J.; Huang, Y.; Xie, Z.; Han, Q.; Cai, Z. Realizing the heterogeneity: A self-organized federated learning framework for IoT. *IEEE Internet Things J.* **2020**, *8*, 3088–3098. [CrossRef]
29. Yan, R.; Qu, L.; Wei, Q.; Huang, S.; Shen, L.; Rubin, D.; Xing, L.; Zhou, Y. Label-Efficient Self-Supervised Federated Learning for Tackling Data Heterogeneity in Medical Imaging. *arXiv* **2022**, arXiv:2205.08576.
30. Feng, M.; Kao, C.C.; Tang, Q.; Sun, M.; Rozgic, V.; Matsoukas, S.; Wang, C. Federated self-supervised learning for acoustic event classification. In Proceedings of the ICASSP 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, 23–27 May 2022; pp. 481–485.
31. Rehman, Y.A.U.; Gao, Y.; Shen, J.; de Gusmao, P.P.B.; Lane, N. Federated self-supervised learning for video understanding. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 506–522.
32. Hojjati, H.; Ho, T.K.K.; Armanfard, N. Self-supervised anomaly detection: A survey and outlook. *arXiv* **2022**, arXiv:2205.05173.
33. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. *Commun. ACM* **2020**, *63*, 139–144. [CrossRef]
34. Zhu, J.Y.; Park, T.; Isola, P.; Efros, A.A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2223–2232.
35. Karras, T.; Laine, S.; Aila, T. A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4401–4410.
36. Van Den Oord, A.; Kalchbrenner, N.; Kavukcuoglu, K. Pixel recurrent neural networks. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016; pp. 1747–1756.
37. Reed, S.; Akata, Z.; Yan, X.; Logeswaran, L.; Schiele, B.; Lee, H. Generative adversarial text to image synthesis. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016; pp. 1060–1069.
38. Kim, T.; Cha, M.; Kim, H.; Lee, J.K.; Kim, J. Learning to discover cross-domain relations with generative adversarial networks. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 1857–1865.
39. Hajiramezanali, E.; Diamant, N.L.; Scalia, G.; Shen, M.W. Stab: Self-supervised learning for tabular data. In Proceedings of the NeurIPS 2022 First Table Representation Workshop, New Orleans, LA, USA, 1 August–26 September 2022.
40. Yoon, J.; Zhang, Y.; Jordon, J.; van der Schaar, M. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 11033–11043.
41. Chitlangia, S.; Muralidhar, A.; Agarwal, R. Self supervised pre-training for large scale tabular data. In Proceedings of the NeurIPS 2022 First Table Representation Workshop, New Orleans, LA, USA, 1 August–26 September 2022.
42. Ucar, T.; Hajiramezanali, E.; Edwards, L. Subtab: Subsetting features of tabular data for self-supervised representation learning. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 18853–18865.
43. LeCun, Y. The MNIST Database of Handwritten Digits. 1998. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 2 February 2024).
44. Kohavi, R. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In Proceedings of the KDD, Portland, OR, USA, 2–4 August 1996; Volume 96, pp. 202–207.
45. Búza, K. Feedback Prediction for Blogs. In Proceedings of the Annual Conference of the Gesellschaft für Klassifikation, Hildesheim, Germany, 1–3 August 2012.