applied sciences

MDPI

Article

Task-Driven Transferred Vertical Federated Deep Learning for Multivariate Internet of Things Time-Series Analysis

Soyeon Oh and Minsoo Lee *

Department of Computer Science and Engineering, Ewha Womans University, Seoul 03760, Republic of Korea; s.oh@ewhain.net

* Correspondence: mlee@ewha.ac.kr; Tel.: +82-2-3277-3401

Abstract: As big data technologies for IoT services develop, cross-service distributed learning techniques of multivariate deep learning models on IoT time-series data collected from various sources are becoming important. Vertical federated deep learning (VFDL) is used for cross-service distributed learning for multivariate IoT time-series deep learning models. Existing VFDL methods with reasonable performance require a large communication amount. On the other hand, existing communication-efficient VFDL methods have relatively low performance. We propose TT-VFDL-SIM, which can achieve improved performance over centralized training or existing VFDL methods in a communication-efficient manner. TT-VFDL-SIM derives partial tasks from the target task and applies transfer learning to them. In our task-driven transfer approach for the design of TT-VFDL-SIM, the SIM Partial Training mechanism contributes to performance improvement by introducing similar feature spaces in various ways. TT-VFDL-SIM was more communication-efficient than existing VFDL methods and achieved an average of 0.00153 improved MSE and 7.98% improved accuracy than centralized training or existing VFDL methods.

Keywords: vertical federated learning; transfer learning; deep learning; time-series; sensor stream; IoT

check for updates

Citation: Oh, S.; Lee, M. Task-Driven Transferred Vertical Federated Deep Learning for Multivariate Internet of Things Time-Series Analysis. *Appl. Sci.* **2024**, *14*, 4606. https://doi.org/10.3390/app14114606

Academic Editor: Christos Bouras

Received: 22 April 2024
Revised: 22 May 2024
Accepted: 24 May 2024
Published: 27 May 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

As various IoT services based on big data and deep learning technologies emerge, the importance of cross-service distributed learning techniques for multivariate deep learning models on IoT time-series data collected from diverse sources is increasing. This is because extended IoT services can be developed through training of a multivariate deep learning model that analyzes the combined IoT time-series dataset from existing services. However, sharing raw data of existing services to distributedly train this multivariate deep learning model is vulnerable in terms of security. Therefore, it is important to develop techniques for distributed learning of multivariate deep learning models for existing services without sharing the raw data of them.

Using vertical federated deep learning (VFDL) methods, we can train a deep learning model for multivariate IoT time-series data distributed across services. VFDL methods are techniques for distributed learning of deep learning models based on vertically partitioned data architectures, where features are distributed [1]. We classified various existing VFDL methods into two categories from the design approach perspective. Existing VFDL methods are based on training loop parallelization approaches, typically Split Learning [2], or data-driven transfer approaches, typically [3], that utilize a transfer learning mechanism which transfers data-driven features trained by unsupervised representation learning. The mechanism of the latest VFDL methods [4,5], which is generally applicable to vertically partitioned multivariate IoT time-series data distributed across services, rather than targeting only security vulnerabilities [6] or specific neural network structures [7], is based on one of the two design approaches we classified.

However, existing VFDL methods still have limitations when applied to multivariate IoT time-series data distributed by service. The design strategy of the training loop

parallelization approach is to train task-driven features by parallelizing forward and back propagation. Because existing VFDL methods of the training loop parallelization approach parallelize forward and back propagation, a large communication amount is required, but because they train task-driven features, it shows similar performance, that is accuracy or MSE, to centralized training. On the other hand, the design strategies of the data-driven transfer approach are to train data-driven features using unsupervised representation learning and design a distributed learning mechanism based on the transfer learning mechanism to transfer the data-driven features. The existing VFDL methods of the data-driven transfer approach are very communication-efficient. This is because they use a transfer learning mechanism to design their distributed learning mechanism where each parallelization unit has a separate training loop. But because they train data-driven features, it shows relatively low performance, that is accuracy or MSE, in centralized training, especially in the IoT domain.

The purpose of this study is to propose a new design approach for vertical federated deep learning mechanisms suitable to the characteristics of the IoT domain. The limitations of existing VFDL methods are because the design strategies for the mechanism of each VFDL method do not sufficiently reflect the characteristics of IoT time-series data. Vertically partitioned IoT time-series data are typically big data distributed across services or data silos. In other words, because the design of a communication-efficient vertical federated deep learning mechanism is necessary, we design a distributed learning mechanism based on transfer learning, similar to the data-driven transfer approach. Meanwhile, even if it is the same type of sensor data, the characteristics that need to be focused on are different depending on the target task. That is, because the target task is more important than data characteristics in the IoT domain, we utilize the task-driven transfer approach rather than the data-driven approach.

We propose a new design approach of vertical federated deep learning suitable for the characteristics of the IoT domain, that is, the task-driven transfer approach. The task-driven transfer approach can overcome the limitations of the existing VFDL design approach in the IoT domain by selecting design strategies suitable for the characteristics of the IoT domain. These design strategies are implemented with four stages of a task-driven transfer approach considering the characteristics of multivariate IoT time-series data. First, considering the fact that each variable in multivariate IoT time-series data is dependent on each other, we fine-tune the multivariate target task based on task-driven features optimized for the univariate partial task. In addition, motivated by the importance of perturbation in IoT time-series data, unlike existing studies that remove perturbation, we prevent overfitting of task-driven features by injecting perturbation through similar feature spaces.

In addition, we designed TT-VFDL-SIM as a VFDL method applying the proposed task-driven transfer approach and evaluated its performance with comparison methods that implemented the existing representative VFDL mechanism. TT-VFDL-SIM achieved an average of 0.00153 improved MSE and 7.98% improved accuracy in a communication-efficient manner compared to centralized training or existing VFDL methods. Focusing on the various characteristics of the IoT domain mentioned above, we devised this task-driven transfer approach and designed TT-VFDL-SIM based on it. Unlike existing VFDL methods designed by the training loop parallelization approach, the task-driven transfer approach is communication-efficient by leveraging the transfer learning mechanism. Also, unlike existing VFDL methods designed by the data-driven transfer approach, the task-driven transfer approach shows enhanced performance by leveraging task-driven features. Especially, the SIM Partial Training mechanism contributes to the performance improvement of TT-VFDL-SIM by preventing overfitting by introducing similar feature spaces in various ways to leverage perturbation.

Through the proposed task-driven transfer approach and TT-VFDL-SIM designed based on it, a deep learning model defined on vertically partitioned multivariate IoT time-series data distributed among existing IoT services or data silos could be distributedly trained. The task-driven transfer approach can be applied to a deep learning model

with separate multivariate inputs with a generalized structure from existing studies for multivariate IoT time-series data analysis. Additionally, the task-driven transfer approach can be applied to an architecture where each input and output is distributed to a separate existing IoT service, as in a real-world scenario. Therefore, TT-VFDL-SIM, implemented based on a task-driven transfer approach, can be used to construct high-quality healthcare services that combine data from each hospital and IoT companies. Or, it can be used to logically combine existing IoT services, such as constructing an integrated analysis service for new economic and business models that combine real economy data and virtual currency data.

The contributions of our study are as follows:

- The task-driven transfer approach contributed to research on transfer learning-based vertical federated deep learning for multivariate IoT time-series data by effectively utilizing the advantages of the transfer learning mechanism and characteristics of the IoT domain, unlike existing VFDL mechanisms.
- TT-VFDL-SIM designed based on the task-driven transfer approach achieved better performance than centralized training and existing VFDL methods in a communication-efficient manner, and its applicability to cross-service distributed learning for multivariate IoT time-series data was validated. Therefore, new IoT services can be cost-effectively constructed through convergence between various domains.
- The task-driven transfer approach and TT-VFDL-SIM designed based on it can be used as a transfer learning-based analysis technique for not only VFDL but also multivariate IoT time-series data from various domains.

Meanwhile, the limitations of our study are as follows:

- Several options designed to apply similar feature spaces introduced in TT-VFDL-SIM for perturbation injection may still be affected by data or task characteristics. Although we were able to experimentally derive the option that could achieve good performance in most cases, the options of TT-VFDL-SIM are not organized into a single mechanism. Therefore, future work to address this problem is required.
- Like representative existing VFDL mechanisms, TT-VFDL-SIM also ensures a basic level of security because it does not expose sensitive raw IoT time-series data. However, encoded label data are exposed for classification target tasks. Although encoded label data are less sensitive than raw IoT time-series data, future work is required on more advanced security techniques to solve the label leakage problem caused by the exposure of encoded label data.

Section 2 summarizes existing studies and introduces the target system architecture and model structure of the target task for applying TT-VFDL-SIM. In addition, we summarize representative existing VFDL methods and introduce comparison methods used in the experiment. Section 3 introduces the task-driven transfer approach to design TT-VFDL-SIM. Section 4 introduces the TT-VFDL-SIM algorithms designed based on the task-driven transfer approach. We have designed two TT-VFDL-SIM algorithms, one for regression target tasks and the other for classification target tasks. In Section 5, we analyze the experimental results of applying TT-VFDL-SIM, centralized training, and two representative existing VFDL methods to three types of target task models and five multivariate IoT time-series datasets. Also, acronyms and terms used in this paper are summarized in Table 1.

Table 1. Acronyms and terms used in this paper.

Type	Acronyms	Description
Generalized Deep Learning Model used in Existing Studies or Target Task Model	FE Part FE Structure	Submodel structure of the target task model to extract features for each separate input stream.
	FM Part FM Structure	Submodel structure to perform "Feature Merging" where all layers are dense layers. That is, a submodel structure of the target task model which concatenates each input's features and trains the target task. It is also used as a merging task model of the host party.
	SLFM	Acronym of "Single Layer Feature Merging". The simplest FM part structure. It consists of only a single dense layer.
	MLFM	Acronym of "Multi-Layer Feature Merging". MLFM consists of multiple dense layers. The features of each input are concatenated at a single level in the first dense layer of MLFM.
	MFCMLFM	Acronym of "Multi-Layer Feature Merging with Multi-layer Feature Concatenation". MFCMLFM consists of multiple dense layers like MLFM, but it concatenates features through multiple depths.
	Split FM	Submodel structure of each univariate partial task model of each guest party. It consists of all or part of layers of the FM part of the target task model.
Proposed Method	TT-VFDL	Task-driven transferred vertical federated deep learning. Vertical federated deep learning based on the proposed task-driven transfer approach.
	TT-VFDL-SIM	Task-driven transferred vertical federated deep learning with the introduction of similar feature spaces. The proposed vertical federated deep learning method designed based on our task-driven transfer approach.
	DFS Model Split Technique	Model split technique of the proposed approach using DFS (Depth-First Search) traversal.
	LSTM-AE-based Data Augmentation Technique	Data augmentation technique of the proposed approach using LSTM autoencoder.
	Target Task	Multivariate IoT time-series analysis task which could be trained distributedly by several VFDL methods including the proposed method.
	Partial Task	Univariate IoT time-series analysis task based on deep learning of each guest party.
	Merging Task	Multivariate task based on deep learning of the host party.
Existing Mechanism or Comparison Methods	DT-VFDL	Vertical federated deep learning designed based on data-driven transfer approach. Existing studies of [3,8–10] correspond to this.
	DT-VFDL-AE	Vertical federated deep learning method based on transfer learning mechanism using data-driven features trained by autoencoder. Existing studies of [3,8–10] correspond to this.
	P-VFDL	Vertical federated deep learning designed based on training loop parallelization approach. Existing studies of [2,11–16] correspond to this.
	P-VFDL-SL	Vertical federated deep learning method of representative mechanism of training loop parallelization approach, which is Split Learning [2]. Existing studies of [11–15] are based on this mechanism.
	P-VFDL-SLG	Vertical federated deep learning method based on the advanced Split Learning mechanism with local gradient update. Method applying existing study [16] to vertical federated deep learning of the target task model.

2. Backgrounds

2.1. Deep Learning-Based Multivariate IoT Time-Series Analysis

A multivariate deep learning model is constructed to analyze IoT time-series data collected from multiple sources with various domains [17,18]. From existing studies, we can examine the general structure of deep learning models used to analyze multivariate IoT time-series data. The multi-sensor stream analysis task for a smart factory extracts features for each sensor and then combines the extracted features to analyze the target task [19]. Human activity recognition (HAR) tasks also extract features from each axis stream of the three-axis accelerometer and the gyro sensor and then combine them for analysis [20]. Even when analyzing a multivariate stream generated from a single sensor, features for each stream are extracted and then combined for analysis [21].

Figure 1 generalizes the structure of the deep learning model [17–21] used in various multivariate IoT time-series analysis. First, features are extracted for each input stream that

constitutes multivariate IoT time-series data. We call the submodel structure that extracts features for each input stream the "FE part". Generally, CNN, LSTM, and CNN-LSTM are used in the FE part [22]. After concatenating the features extracted through the FE part from each input stream, the target task is trained to reflect the combination information between each input. We call this process "feature merging", and the submodel structure for feature merging is called the "FM part". The FM part consists only of dense layers and can have a concatenation structure of various depths.

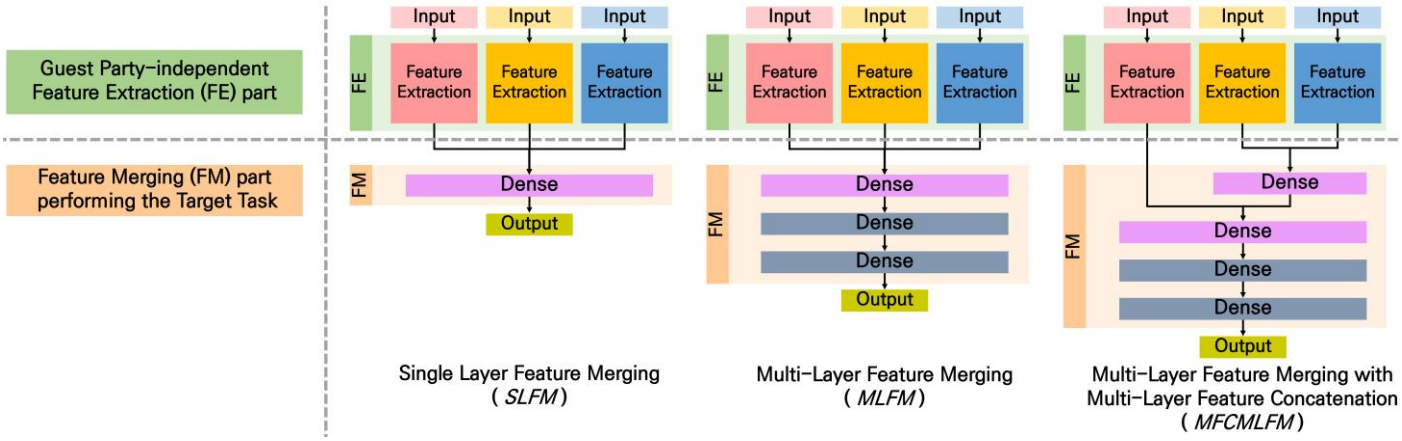


Figure 1. General structure of deep learning models used in multivariate IoT time-series analysis: multivariate deep learning models can be distinguished according to the type of the FM part. The three types of FM parts (SLFM, MLFM, and MFCMLFM) are determined by the level of the concatenation structure.

In Figure 1, we generalize various FM part structures into three types: SLFM, MLFM, and MFCMLFM. SLFM, the simplest FM part structure, consists of only a single dense layer. Through this single dense layer, the features of each input stream are concatenated at a single level, and the target task is trained based on this concatenated feature. MLFM consists of multiple dense layers. The features of each input are concatenated at a single level in the first dense layer of MLFM. MFCMLFM consists of multiple dense layers like MLFM, but it concatenates features through multiple depths. In other words, MFCMLFM assigns priority to feature concatenation for each input and performs multi-level feature concatenation based on this.

For the experiments in Section 5, we constructed a multivariate IoT time series dataset with a similar domain to [1–5]. In addition, for each dataset, three target task models according to the type of FM part were designed as in Figure 1 and used in the experiment.

2.2. Vertical Federated Learning (VFL)

Vertical federated learning is a distributed learning technique based on vertically partitioned data where features are distributed in the sample direction [1]. Each party, which is the distributed learning unit of vertical federated learning, is generally a service or a silo, a large-scale storage unit. Therefore, vertical federated learning is used for distributed learning between services or silos.

Meanwhile, [23] is a representative horizontal federated learning algorithm, a distributed learning method based on horizontally partitioned data. Unlike vertically partitioned data where features are distributed, horizontally partitioned data is a data architecture where samples are distributed. Therefore, horizontal federated learning is mainly used in edge computing.

There are distributed learning methods for high-performance computing based on parallel processing units such as GPUs or TPUs [24,25]. Vertical federated learning and [24,25] have different parallelization units and target system architecture. Ideas from distributed

learning methods for high-performance computing or horizontal federated learning methods can also be applied to the design of vertical federated learning. However, if the characteristics of the parallelization unit or target architecture are not sufficiently considered, it may be inefficient in terms of communication.

2.3. System Architecture for Vertical Federated Learning

The distributed system architecture commonly assumed in existing vertical federated learning studies is an active/passive party architecture [3,8,13]. A passive party stores only inputs, and an active party stores both inputs and outputs. Therefore, in this architecture, raw output data are not exposed due to the active party. In multivariate IoT time-series analysis, the output data of the classification target task is an encoded label, so it is suitable for an active/passive party architecture. However, the output data of the regression target task are IoT time-series data. Additionally, in real-world scenarios, the output data of the regression target task are also often stored in a separate existing IoT service. Therefore, an active/passive party architecture is not suitable for the regression target task scenario for multivariate IoT time-series data.

A completely distributed architecture is more suitable for various real-world multivariate IoT time-series analysis scenarios, including regression. In a completely distributed architecture, not only each input but also the output is distributed to each party. Among the parties in a completely distributed architecture, the host party stores the output, and the guest parties store each input. Split Learning [2] can be applied to vertical federated learning based on this completely distributed architecture. There is [11] which applied [2] to an integrated healthcare service construction scenario based on completely distributed architecture. There is [16], a horizontal federated learning method based on Split Learning, but because it has a local gradient update mechanism, each client and server play a role similar to that of a guest party and host party of the completely distributed architecture.

In a completely distributed architecture, whether the raw output data of the host party can be copied to each guest party differs for each study. Some methods assume that the raw output data of the host party cannot be copied to each guest party [2,11]. On the other hand, if [16] is applied to vertical federated learning, it must be assumed that each client holds the server's raw output data. However, in a real-world regression scenario, copying the host party's raw output data to each guest party is very vulnerable from a security perspective. Therefore, we prohibit copying the host party's raw output data to each guest party in TT-VFDL-SIM for the regression target task. However, in TT-VFDL-SIM for the classification target task, copying the host party's encoded label to each guest party is allowed.

2.4. Existing Vertical Federated Deep Learning (VFDL) Methods

Vertical federated deep learning (VFDL) methods are vertical federated learning methods for deep learning models. Existing VFDL methods can be divided into two types depending on the approach for mechanism design. First, VFDL methods of the training loop parallelization approach [2,11–15] parallelize the forward propagation and back propagation that constitute the training loop for the target task. Therefore, since communication for distributed learning occurs at every step of the training loop, a very large communication amount is required. On the other hand, VFDL methods of the data-driven transfer approach [3,8–10] train the target task by transfer learning based on features trained by unsupervised representation learning at each party. It is very communication-efficient because only one step of communication is required to transfer the trained features from each party to another party.

Figure 2 summarizes the mechanism of applying the proposed TT-VFDL-SIM and representative existing VFDL methods to a deep learning model for multivariate IoT time-series analysis. CNTR is a centralized training mechanism that is not parallelized, and the rest are VFDL mechanisms for a deep learning model with the same structure as CNTR. P-VFDL-SL is the Split Learning [2] mechanism most commonly used in existing VFDL

methods of the training loop parallelization approach. P-VFDL-SLG is an implementation of [16], which added a local gradient update mechanism to improve the performance of Split Learning, tailored to the VFDL scenario for multivariate IoT time-series data. DT-VFDL-AE represents existing VFDL methods of the data-driven transfer approach and utilizes an autoencoder for its unsupervised representation learning. TT-VFDL-SIM was designed based on our task-driven transfer approach.

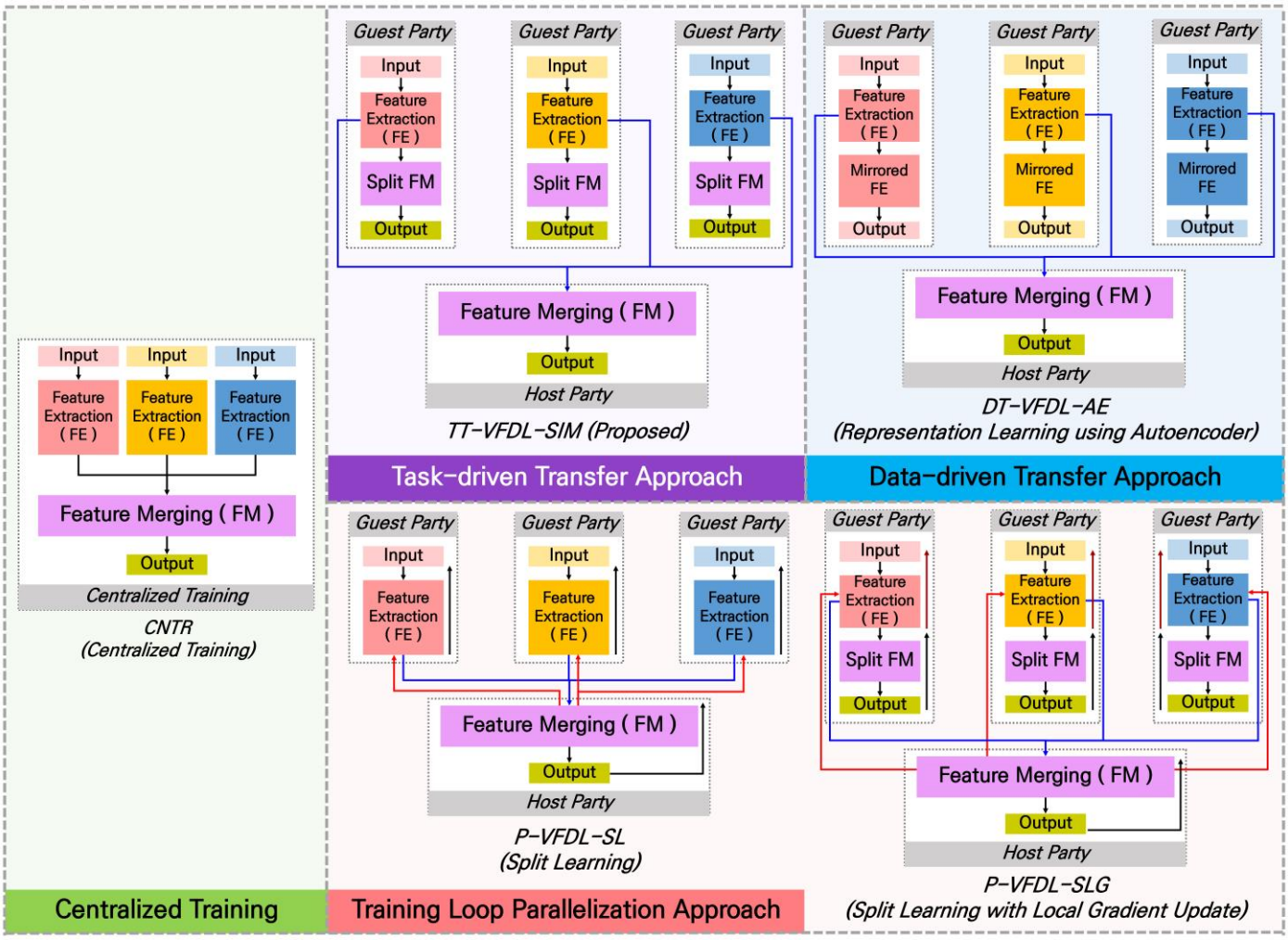


Figure 2. Training mechanism of proposed TT-VFDL-SIM and other comparison methods: CNTR is a centralized training mechanism. TT-VFDL-SIM and other comparison methods perform VFDL on a deep learning model with the same structure as CNTR. P-VFDL-SL is a Split Learning mechanism, one of the representative VFDL methods using the training loop parallelization approach. P-VFDL-SLG is Split Learning with a local gradient update mechanism added. DT-VFDL-AE is based on a data-driven transfer approach using unsupervised representation learning. TT-VFDL-SIM is based on our task-driven transfer approach. The blue arrows are the communication from each guest party to the host party, and the red arrows are the communication from the host party to each guest party.

2.4.1. Training Loop Parallelization Approach

Most of the existing VFDL methods based on the training loop parallelization approach split the target task model and parallelize forward propagation and back propagation based on the split submodels [2,11–15]. On the other hand, [14,15] defines a final loss that aggregates the training results of each party and parallelizes forward propagation and back propagation based on this. However, in all VFDL methods based on the training loop parallelization approach, communication occurs at every step of the training loop. As a

result, the amount of communication required for distributed learning is very large, and research is also being conducted to reduce the amount of communication, such as in [26].

Most VFDL methods based on the training loop parallelization approach leverage Split Learning [2]. In Split Learning, the target task model is split based on the set cut layer. Then, forward propagation and back propagation are parallelized based on this cut layer. Communication required for distributed learning also occurs at the cut layer. Several existing VFDL methods are based on Split Learning [11–13]. Some methods applied Split Learning to a federated healthcare service scenario based on a completely distributed architecture [11]. There are some methods that applied Split Learning to a VFDL scenario based on an active/passive party architecture [12,13]. In addition to vertical federated learning scenarios, Split Learning can also be applied to horizontal federated learning scenarios [16,27] or hybrid architectures [28,29].

P-VFDL-SL in Figure 2 applies Split Learning [2] to the vertical federated learning scenario for multivariate IoT time-series analysis. First, each guest party forward propagates its own submodel (FE part) consisting of all layers from the input layer to the cut layer. Then, the features of each cut layer are sent to the host party. The host party completes forward propagation for its submodel (FM part) based on the received features and calculates the loss at the output layer. The host party starts back propagation based on this loss. To complete parallelized back propagation, the host party transmits the gradient calculated at the cut layer to each guest party. P-VFDL-SLG is an implementation of [16] that added a local gradient update mechanism to improve the performance of Split Learning. It was originally a technique for horizontal federated learning scenarios but was implemented as P-VFDL-SLG to fit the VFDL scenario for multivariate IoT time-series data.

2.4.2. Data-Driven Transfer Approach

Transfer learning [30] is a mechanism that pre-trains a task in a source domain similar to the target domain and then trains the task in the target domain using the knowledge obtained from the source domain. Features or weights pre-trained in the source domain can be transferred to the target domain. In the target domain, additional fine-tuning can be performed based on features or weights transferred from the source domain. Transfer learning is generally applied in situations where there are sufficient data.

Existing VFDL methods of the data-driven transfer approach train the target task by transferring features trained by each party through unsupervised representation learning to other parties [3,8–10]. In [3], passive parties transfer features trained through unsupervised representation learning to the active party, and the active party trains the target task based on this. There is a VFDL method for vertically partitioned image data [8]. The active party transfers the feature extractor trained with the labeled partial image to each passive party, and each passive party trains an autoencoder based on it. Features obtained from the autoencoder of each passive party are transferred back to the active party and used for training the target task. In [9], each guest party trains an overcomplete autoencoder on its own data. The host party trains the target task based on expanded features extracted from the overcomplete autoencoder of each guest party. Each guest party in [10] uses PCA and an autoencoder for representation learning.

As in [3,8–10], existing VFDL methods of the data-driven transfer approach generally utilize an autoencoder for unsupervised representation learning. Since DT-VFDL-AE in Figure 2 is a representative mechanism of existing VFDL methods that utilize a data-driven transfer approach, an autoencoder is used for unsupervised representation learning at each guest party. In existing VFDL methods of data-driven transfer approach, only one step of communication occurs to transmit the trained features from each guest party to the host party. Due to this one-shot communication mechanism, existing VFDL methods of data-driven transfer approach are communication-efficient. However, they generally show relatively lower performance than centralized training.

2.5. Autoencoder for the Time-Series Analysis

In the analysis of various types of IoT time-series data, including sensor streams, autoencoders are used for various purposes [31–35]. Typically, this is when it is necessary to reconstruct sensor stream data for analysis. It is effective to reconstruct the sensor stream data using the LSTM-based autoencoder (LSTM-AE) [31]. Additionally, an autoencoder is also used as part of the transfer learning technique [32,33]. First, pre-train each layer of the target task model using an autoencoder. Then, the weight of each pre-trained layer is initialized into the target task model to train the target task. Existing VFDL methods of the data-driven transfer approach [3,8–10] utilize an autoencoder for unsupervised representation learning of each guest party. Some studies use LSTM-AE-based reconstructed data to augment insufficient training data [34]. That is, [34] introduces LSTM-AE-based reconstructed data to increase the number of insufficient samples of classes with relatively small distributions in the imbalanced training dataset.

The proposed TT-VFDL-SIM also introduces LSTM-AE-based reconstructed data in the partial task construction stage of each guest party. However, this has a clearly different purpose and meaning from [34], which uses LSTM-AE-based reconstructed data for data augmentation. In TT-VFDL-SIM, LSTM-AE-based reconstructed data are used for the purpose of creating similar feature spaces and are not used for the purpose of increasing the number of insufficient samples. In TT-VFDL-SIM, each guest party trains an LSTM-AE that can reconstruct its own raw data. Then, a reconstructed dataset is created based on this LSTM-AE. These LSTM-AE-based reconstructed data are used for input and output of the deep learning model along with the raw data of each guest party. That is, in TT-VFDL-SIM, each guest party trains a model based on similar feature spaces generated from its own raw data and LSTM-AE-based reconstructed data.

Meanwhile, in [35], the target task was analyzed based on convolutional autoencoder-based reconstructed data instead of raw data. As a result, performance was further improved when the target task was analyzed using convolutional autoencoder-based reconstructed data than when the target task was analyzed using raw data. These experimental results in [35] can support the justification for introducing LSTM-based reconstructed data in the design of TT-VFDL-SIM. However, unlike [35], which uses only reconstructed data for analysis, TT-VFDL-SIM uses two similar feature spaces created from reconstructed data and raw data for analysis. Therefore, there is a clear difference between TT-VFDL-SIM and [35] in the purpose of utilizing reconstructed data.

3. Task-Driven Transfer Approach

The target task of TT-VFDL-SIM is a multivariate task that accepts all raw data from each guest party as input and can be analyzed using a deep learning model with the structure shown in Figure 1. Each guest party in the completely distributed architecture stores IoT time-series data such as sensor streams. If the host party's data are IoT time-series data, the target task is a regression task. And if the host party's data are encoded label data, the target task is a classification task. The partial tasks derived from the target task are univariate tasks that have only data from each guest party as input. The host party's merging task is also derived from the target task and is a multivariate task that trains the target task using each guest party's task-driven features as input.

TT-VFDL-SIM derives partial tasks from the target task, then reorganizes these partial tasks into pre-training and fine-tuning tasks for the target task and performs transfer learning on them to distributedly train the target task. We call this approach the "task-driven transfer approach". Since each variable in multivariate IoT time-series data is generally dependent on each other, the task-driven transfer approach leverages univariate partial tasks for the pre-training of multivariate target tasks. Each variable in multivariate IoT time-series data is created simultaneously in the process of monitoring a specific environment or condition. Therefore, we assumed that task-driven features optimized for univariate partial tasks would also be suitable for the multivariate target task. For similar

reasons, it was assumed that task-driven features would be more suitable than data-driven features in multivariate IoT time-series analysis.

In this section, we describe each stage of the task-driven transfer approach to design TT-VFDL-SIM. The task-driven transfer approach consists of the following four stages:

1. **Partial Task Construction (Preparation)**
   - Derive partial tasks and merging task model structure (*DFS Model Split technique*)
   - Generate similar feature spaces (*LSTM-AE-based Data Augmentation technique*)
2. **Partial Task Training (Pre-training for the target task)**
   - Introduce similar feature spaces (*SIM Partial Training mechanism*)
3. **One-shot Communication**
   - Transfer task-driven features
4. **Merging Task Training (Fine-tuning for the target task)**
   - Train the target task with concatenated task-driven features

The first partial task construction is a necessary preparation stage prior to distributed learning. In partial task construction stage, partial tasks are derived from the target task. The host party's partial task is specially called a merging task. The model structure of each partial task and merging task is derived, and similar feature spaces necessary for training are created. After this preparation stage is completed, each guest party's partial task and the host party's merging task are reorganized into a pre-training and fine-tuning task for the target task according to the transfer learning mechanism. In TT-VFDL-SIM, distributed learning for the target task occurs through three stages following the partial task construction stage, and these three stages are summarized in Figure 3.

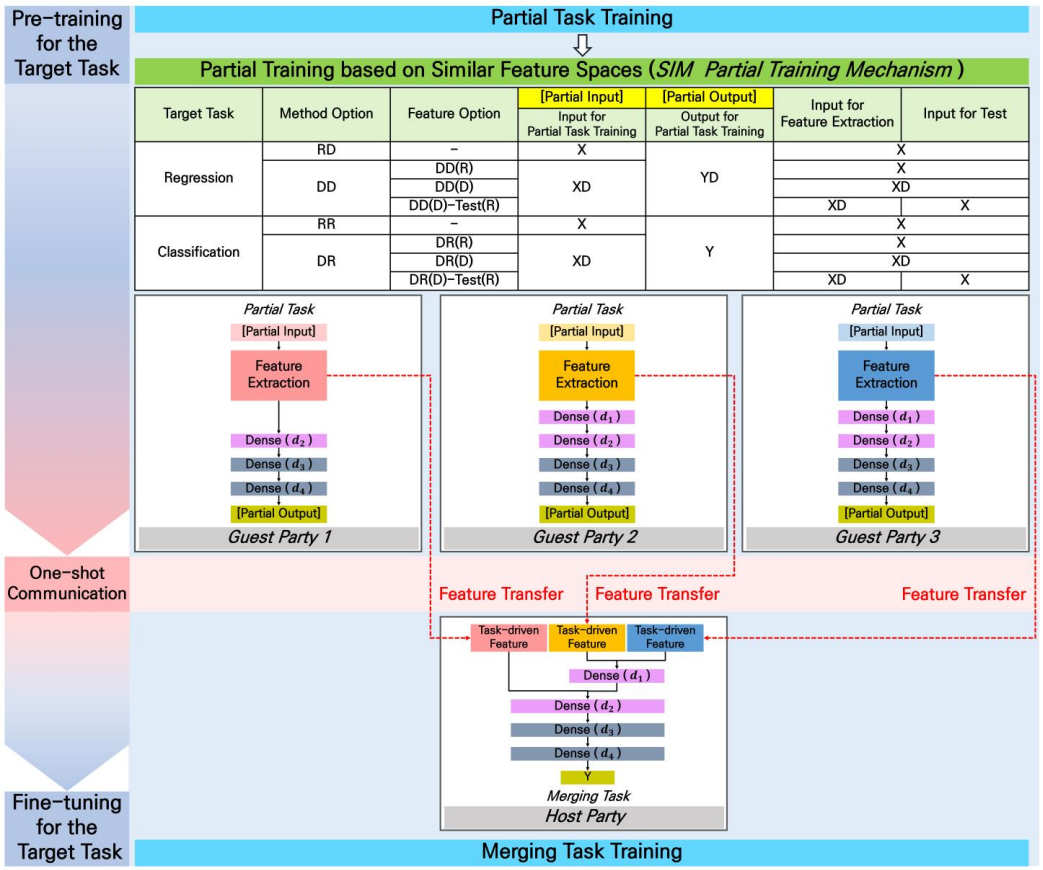


Figure 3. Distributed learning mechanism of TT-VFDL-SIM. The partial task model of each guest party and the merging task model of the host party have already been derived through partial task construction before the pre-training for the target task.

Through Figure 3, we can see how the stages after the partial task construction in the task-driven transfer approach are applied to the design of the distributed learning mechanism of TT-VFDL-SIM. Each guest party trains its own partial task to pre-train the target task. The SIM Partial Training mechanism presents various methods of introducing similar feature spaces previously created into the training of each partial task. After partial task training, the task-driven features of each guest party are transmitted to the host party through a one-shot communication mechanism. Finally, the host party trains the merging task by concatenating the task-driven features of guest parties. The target task is fine-tuned through merging task training.

Section 3.1 describes two techniques used in partial task construction. First, we describe the DFS (Depth-First Search) Model Split technique to derive the univariate model structure of each partial task from the multivariate target task. Second, we describe an LSTM-AE-based Data Augmentation technique that creates similar feature spaces to configure the input and output of each partial task model. Section 3.2 describes partial task training to pre-train the target task. In this paper, each partial task is trained by introducing similar feature spaces through the SIM Partial Training mechanism. Section 3.3 describes one-shot communication and merging task training to fine-tune the target task.

3.1. Partial Task Construction

Partial task construction leverages the DFS (Depth-First Search) Model Split technique and LSTM-AE-based Data Augmentation technique to derive each guest party's partial task and the host party's merging task from the target task. The partial task model structure of each guest party and the merging task model structure of the host party are derived by the DFS Model Split technique. Additionally, similar feature spaces to configure the input and output of each partial task model are created by the LSTM-AE-based Data Augmentation technique. Each guest party or host party can generate reconstructed data for its own raw data based on the LSTM-AE-based Data Augmentation technique. We call this reconstructed data "decoded data". As shown in Figure 3, a total of four similar feature spaces can be created by combining raw data and decoded data.

3.1.1. DFS (Depth-First Search) Model Split Technique

As shown in Figure 4, the DFS Model Split technique parses the layer-level model structure graph of the target task to derive the partial task model structure of each guest party and the merging task model structure of the host party. The partial task model structure and merging task model structure of TT-VFDL-SIM are determined only by the DFS Model Split technique.

Each guest party uses the input layer for its own data as the starting node and parses the layer-level model structure graph of the target task based on Depth-First Search (DFS). When parsing the FM part of the target task model, the number of units d_l in each dense layer l is kept the same. The FM part parsed from the target task model in this way is called "Split FM". As a result, each partial task model structure of TT-VFDL-SIM consists of an FE part for each input and a split FM, as shown in Figure 2.

The host party's merging task model structure is also derived through the DFS Model Split technique. The input to the merging task model is task-driven features sent from each guest party. Therefore, the starting node of DFS-based graph parsing in the host party is the layer that concatenates the last outputs of each FE part of the target task model. As a result, the merging task model structure is identical to the FM part structure of the target task model.

Table 2 summarizes all formulations used in this paper. Table 3 summarizes the partial task model structure of each guest party and the merging task model structure of the host party for TT-VFDL-SIM using the formulations in Table 2. The model structures trained by each party in P-VFDL-SL and DT-VFDL-AE, the comparison methods summarized in Section 2.4, are also summarized in Table 3.

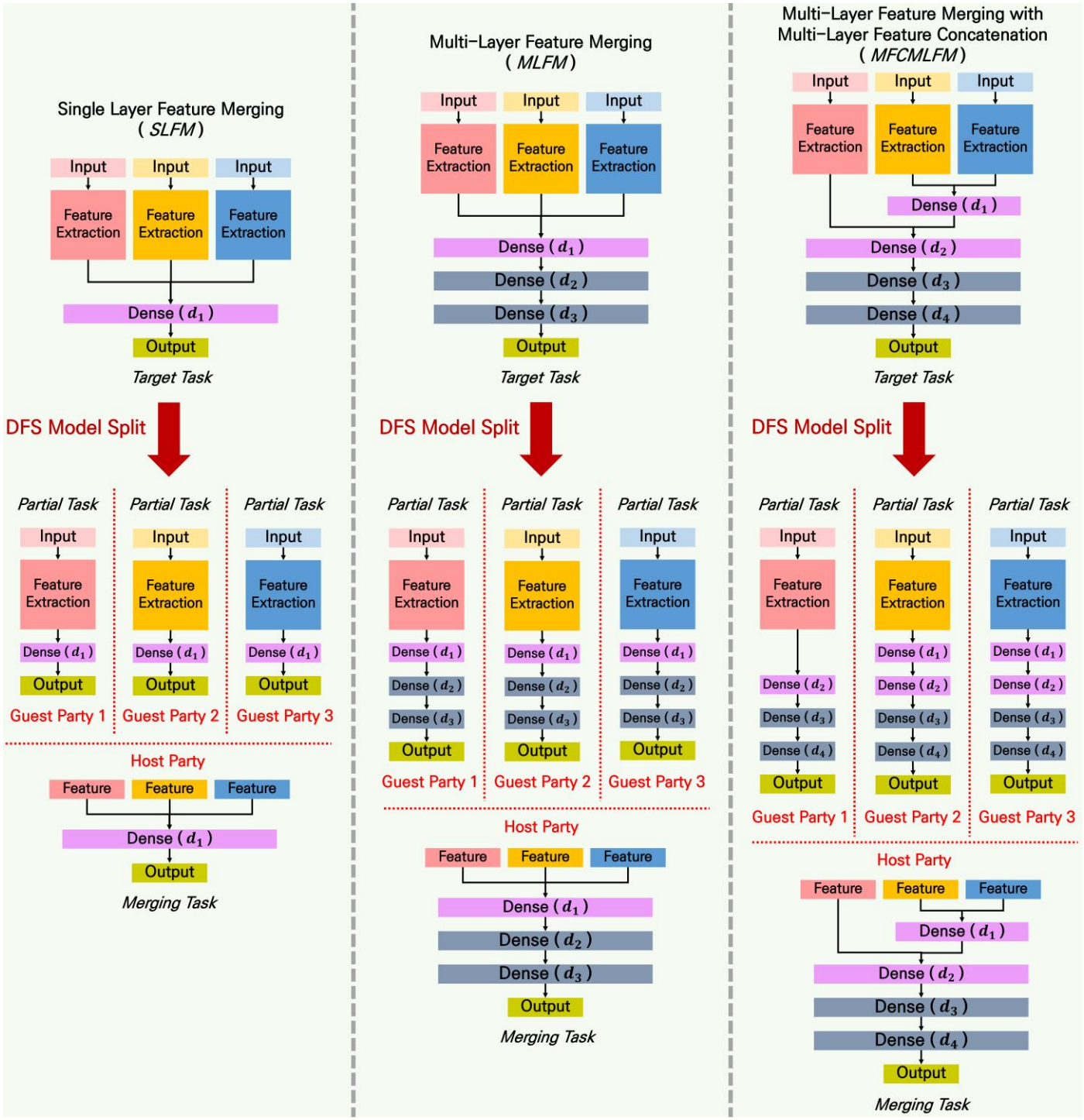


Figure 4. DFS (Depth-First Search) Model Split technique to derive a partial task model structure of each guest party and merging task model structure of the host party: this is the result of applying the DFS Model Split technique to three target tasks for each type of FM part. According to Depth-First Search graph parsing, starting from the input layer for each guest party's data, it reaches the output layer through the corresponding separate FE part and FM part layers.

Table 2. Formulation used in this paper.

Formulation	Description
f^k	The partial task model (FE part of the target task and Split FM part derived by DFS Model Split) of each Guest Party k when $k = 1, \ldots, K$ The merging task model of the Host Party (FM part of the target task) when $k = 0$
e_{FE}^k	The feature extraction part of the partial task model or the encoder of the FE-based autoencoder (FE part of the target task) of each Guest Party k when $k = 1, \ldots, K$
d_{FE}^k	The decoder of the FE-based autoencoder (mirror image of FE part of the target task) of each Guest Party k when $k = 1, \ldots, K$
e_{DA}^k	The encoder of the LSTM-AE of each Guest Party k for the LSTM-AE-based Data Augmentation
d_{DA}^k	The decoder of the LSTM-AE of each Guest Party k for the LSTM-AE-based Data Augmentation
ω^k	The parameter of the partial task model ($k = 1, \ldots, K$) or the merging task model ($k = 0$) f^k
θ^k	The parameter of the encoder e_{FE}^k of the FE-based autoencoder
ϕ^k	The parameter of the decoder d_{FE}^k of the FE-based autoencoder
δ_e^k	The parameter of the encoder e_{DA}^k of the autoencoder for the LSTM-AE-based Data Augmentation
δ_d^k	The parameter of the decoder d_{DA}^k of the autoencoder for the LSTM-AE-based Data Augmentation
L	Loss Function
X_i^k	i-th input in the dataset of each Guest Party k ($k = 1, \ldots, K$)
Y_i	i-th output in the dataset of the Host Party
XC_i^k	i-th encoded input of each Guest Party k ($k = 1, \ldots, K$) by LSTM-AE-based Data Augmentation $XC_i^k \leftarrow e_{DA}^k\left(X_i^k; \delta_e^k\right)$
YC_i	i-th encoded window of Y sent to each Guest Party k ($k = 1, \ldots, K$) from the Host Party ($k = 0$) $YC_i \leftarrow e_{DA}^0\left(Y_i; \delta_e^0\right)$
XD_i^k	i-th decoded input of each Guest Party k ($k = 1, \ldots, K$) by LSTM-AE-based Data Augmentation $XD_i^k \leftarrow d_{DA}^k\left(XC_i^k; \delta_d^k\right)$
YD_i	i-th decoded Y, which is used as the true output of the partial task model of each Guest Party k ($k = 1, \ldots, K$) $YD_i \leftarrow d_{DA}^0\left(YC_i; \delta_d^0\right)$
F_i^k	i-th extracted feature of each Guest Party k ($k = 1, \ldots, K$) $F_i^k \leftarrow e_{FE}^k\left(X_i^k; \theta^k\right)$

Table 3. Overview of the proposed TT-VFDL-SIM and other comparison methods.

Methods	Design Approach	Guest Party		Host Party		Communication
[Proposed] **TT-VFDL-SIM (RD) *** **TT-VFDL-SIM (DD) *** for Regression **TT-VFDL-SIM (RR) *** **TT-VFDL-SIM (DR) *** for Classification		**Model Structure**		**Model Structure**		One-shot Communication
		f^k $k = 1, \ldots, K$	Partial Task (FE + Split FM)	f^0	Merging Task (FM)	
		e_{FE}^k $k = 1, \ldots, K$	FE of Partial Task			
		Pre-training for the Target Task	SIM Partial Training (Supervised)	**Fine-tuning for the Target Task**	Merging Task Training (Supervised)	
[Existing Mechanism] **DT-VFDL-AE** [3,8–10]	Data-driven Transfer	**Model Structure (DT-VFDL-AE)** FE-based Autoencoder		**Model Structure**		One-shot Communication
		e_{FE}^k	FE of Partial Task (Encoder)	f^0	Merging Task (FM)	
		d_{FE}^k	FE-based Decoder			
		Pre-training for the Target Task	Representation Learning (Unsupervised)	**Fine-tuning for the Target Task**	Merging Task Training (Supervised)	
[Existing Mechanism] **P-VFDL-SL** [2,11–15] **P-VFDL-SLG** [16]	Training Loop Parallelization	**Forward Propagate** (**F.P.**) FE e_{FE}^k **Send Feature** F_i^k **[P-VFDL-SLG] F.P.** Partial Task f^k		**Receive** Feature F_i^k **Forward Propagate** FM f^0		Communication Per Every Step
		Receive Gradient **Back Propagate** (**B.P.**) FE e_{FE}^k **[P-VFDL-SLG] B.P.** Partial Task f^k		**Back Propagate** FM f^0 **Send** Gradient		
CNTR	Centralized Training for the Target Task Model					

* The proposed TT-VFDL-SIM method with various method options.

The structures of the merging task model of TT-VFDL-SIM and DT-VFDL-AE are the same, but their inputs are different. The merging task model of TT-VFDL-SIM takes the task-driven features of each guest party as input. For this purpose, each guest party in TT-VFDL-SIM trains a partial task model consisting of the FE part and Split FM. On the other hand, the merging task model of DT-VFDL-AE takes the data-driven features of each guest party as input. For this purpose, each guest party trains an autoencoder for unsupervised representation learning, and the encoder structure of the autoencoder is the same as the FE part. Meanwhile, in P-VFDL-SL, each guest party trains the FE part distributedly through the first half of forward propagation and the second half of back propagation. The host party trains the FM part distributedly through the second half of forward propagation and the first half of back propagation.

3.1.2. LSTM-AE-Based Data Augmentation Technique

In order to train each partial task model structure derived through the DFS Model Split technique, input and output must be assigned. For this purpose, we propose the LSTM-AE-based Data Augmentation technique. An overview of the LSTM-AE-based Data Augmentation technique is shown in Figure 5. The raw data of each guest party and host party are reconstructed based on LSTM-AE, and these reconstructed data (decoded data) are used for the input and output of each partial task model. The purposes of introducing decoded data into the input and output of the partial task model are as follows:

1. When training a regression target task based on a completely distributed architecture, output data to train the partial task of each guest party are prepared without exposing the host party's raw data.
2. By introducing similar feature spaces generated from raw data and decoded data into partial task training, overfitting of the univariate partial tasks that pre-train the multivariate target task is prevented.

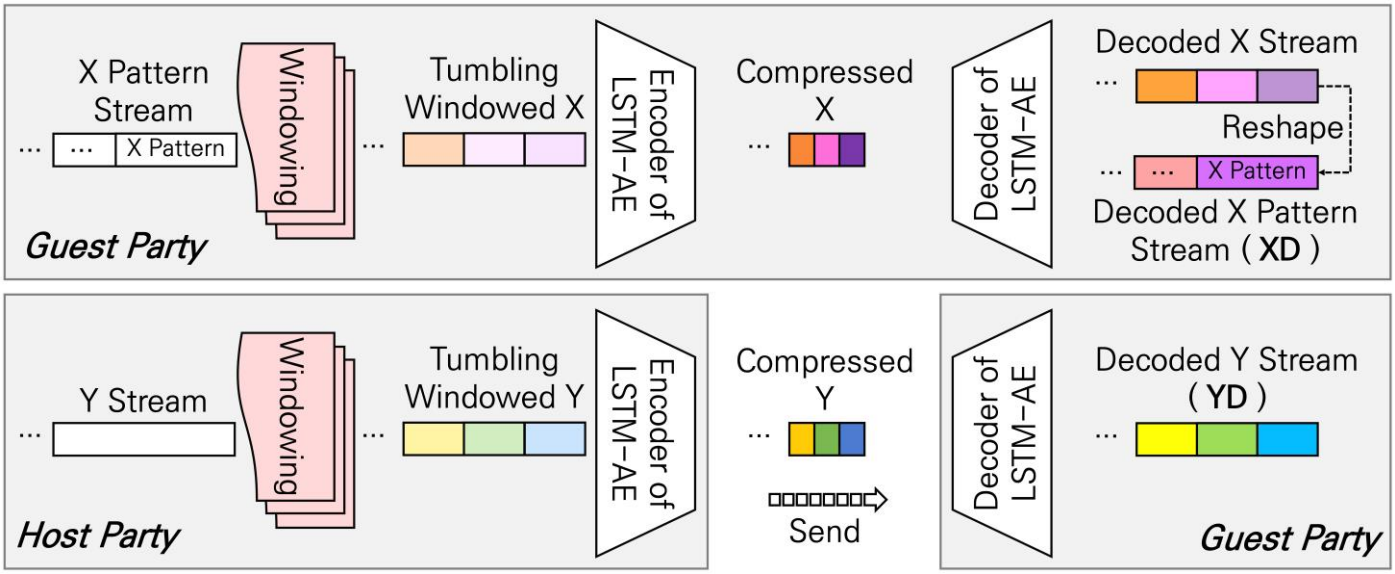


Figure 5. LSTM-AE-based Data Augmentation technique to generate similar feature spaces. It is assumed that each guest party and host party configure both the encoder and decoder of LSTM-AE as a single LSTM layer.

TT-VFDL-SIM requires output data to train the partial task of each guest party. In the classification target task, the host party's raw output data, that is, encoded label data, can be transmitted to each guest party. However, due to security issues, the raw data of the host party cannot be used for the partial task training of each guest party in the regression target task. In the regression target task, the partial task of each guest party can be trained without

exposing the host party's raw data by leveraging the LSTM-AE-based Data Augmentation technique.

The host party trains an LSTM-AE that can reconstruct its raw output data, as shown in Figure 5. Then, based on the trained LSTM-AE, an encoded latent vector dataset for the raw output data is generated and sent to each guest party along with the decoder parameters of the LSTM-AE. Each guest party constructs a decoder based on the parameters received from the host party and applies it to the encoded latent vector dataset. As a result, each guest party can generate decoded data for the host party's raw output data and use this as output for partial task training.

Meanwhile, each guest party can use decoded data generated from its own raw data as input for partial task training. This is unrelated to the exposure issue of raw data for each guest party but rather intended to create similar feature spaces. The output of each partial task model is determined as either the host party's encoded label data or decoded data depending on the type of target task. Therefore, in one partial task model, two similar feature spaces can be defined depending on the type of input data, that is, raw data or decoded data.

In this paper, the decoded data for the host party's raw data Y in the regression target task are denoted as YD. Meanwhile, in the classification target task, there is no need to generate LSTM-AE-based decoded data for Y, so the host party's YD is not defined. Additionally, the decoded data for each guest party's raw data X are denoted as XD. These notations are also summarized in Table 2 and used throughout this paper, including Tables 3 and 4 and Figures 3 and 5.

Table 4. Objective function of the proposed TT-VFDL methods and feature transfer-based DT-VFDL methods.

Methods	Guest Party		Host Party	
	Pre-training for the Target Task	SIM Partial Training (Supervised)	**Fine-tuning for the Target Task**	Merging Task Training (Supervised)
[Proposed] **TT-VFDL-SIM (RD) *** **TT-VFDL-SIM (DD) *** for Regression **TT-VFDL-SIM (RR) *** **TT-VFDL-SIM (DR) *** for Classification	**Objective Function (Regression RD)**	$\underset{\omega^k}{\operatorname{argmin}} \sum_i L\left(f^k\left(\omega^k; X_i^k\right), YD_i\right)$ $k = 1, \ldots, K$	**Objective Function**	$\underset{\omega^0}{\operatorname{argmin}} \sum_i L\left(f^0\left(\omega^0; F_i^1, \ldots, F_i^K\right), Y_i\right)$
	Objective Function (Regression DD)	$\underset{\omega^k}{\operatorname{argmin}} \sum_i L\left(f^k\left(\omega^k; XD_i^k\right), YD_i\right)$ $k = 1, \ldots, K$		
	Objective Function (Classification RR)	$\underset{\omega^k}{\operatorname{argmin}} \sum_i L\left(f^k\left(\omega^k; X_i^k\right), Y_i\right)$ $k = 1, \ldots, K$		
	Objective Function (Classification DR)	$\underset{\omega^k}{\operatorname{argmin}} \sum_i L\left(f^k\left(\omega^k; XD_i^k\right), Y_i\right)$ $k = 1, \ldots, K$		
[Existing Mechanism] **DT-VFDL-AE** [3,8–10]	**Pre-training for the Target Task**	Representation Learning (Unsupervised)	**Fine-tuning for the Target Task**	Merging Task Training (Supervised)
	Objective Function	$\underset{\phi^k, \theta^k}{\operatorname{argmin}} \sum_i L\left(d_{FE}^k\left(\phi^k; e_{FE}^k\left(\theta^k; X_i^k\right)\right), X_i^k\right)$	**Objective Function**	$\underset{\omega^0}{\operatorname{argmin}} \sum_i L\left(f^0\left(\omega^0; F_i^1, \ldots, F_i^K\right), Y_i\right)$

* The proposed TT-VFDL-SIM method with various method options.

In addition, to indicate the type of input/output data for training a partial task model, that is, the type of feature space, two uppercase letters are sometimes concatenated as shown in Table 4. The first capital letter indicates the type of input data, and the second capital letter indicates the type of output data. Additionally, R stands for raw data and D stands for decoded data. To easily distinguish between the types of feature spaces, R and D are marked in blue and red colors, respectively.

3.2. Partial Task Training (SIM Partial Training Mechanism)

TT-VFDL-SIM pre-trains target tasks based on partial tasks. The target task is a multivariate task that takes raw data from all guest parties as input. On the other hand, a partial task is a univariate task that takes only the raw data of each guest party as input. Therefore, in order to improve the performance of fine-tuning for the multivariate target task, it is necessary to prevent overfitting of the univariate partial task, which is a pre-training task of the target task.

To achieve this, we introduce the previously generated similar feature spaces and train partial tasks. Two similar feature spaces can be defined for one partial task model, and there are various ways to create task-driven features for each feature space. We organized various methods of introducing similar feature spaces through the SIM Partial Training mechanism.

SIM Partial Training is a one-phased training mechanism like ordinary training, but one of similar feature spaces can be selected depending on the method option. Additionally, it has several feature options for extracting task-driven features depending on the feature space. The feature options are summarized in the algorithm of Section 4. Furthermore, the performance based on each method option and feature option of SIM Partial Training was analyzed through experiments in Section 5.

In Table 4, the objective functions for each guest party and host party in TT-VFDL-SIM are summarized. Table 4 shows the method options of the SIM Partial Training mechanism depending on the type of target task. Similar feature spaces that can be assigned to partial tasks are RD, DD, RR, and DR. Among these, the two method options for the classification target task are RR and DR, and the two method options for the regression target task are RD and DD. A more detailed description related to this is as follows.

Similar feature spaces that can be selected in SIM Partial Training are prioritized to ensure basic security depending on the type of the target task. In the regression target task, the host party's sensitive raw output (Y) cannot be used for privacy reasons, so decoded data (YD) are used as output. On the other hand, in the classification target task, the relatively less sensitive encoded label of the host party (Y) is received and used as output. After the type of output for each partial task is determined according to the type of target task, two similar feature spaces can be defined depending on the type of available input. In other words, two similar feature spaces are defined for each target task depending on whether raw data (X) or decoded data (XD) of each guest party are used as input.

As a result, one of the two similar feature spaces RD and DD can be selected in the regression target task, and one of the two similar feature spaces RR and DR can be selected in the classification target task. Which of the two similar feature spaces is selected is implemented as a method option. In the regression target task, when the method option is RD, SIM Partial Training can be performed using each guest party's raw data as input and the host party's decoded data as output. On the other hand, when method option DD is used, the decoded data of each guest party are used as input and the decoded data of the host party are used as output. Similarly, in the classification target task, when the method option is RR, the raw data of each guest party are used as input and the encoded label of the host party is used as output. On the other hand, when method option DR is used, the decoded data of each guest party are used as input and the encoded label of the host party is used as output.

Meanwhile, additional feature options exist in method options DD and DR that use each guest party's decoded data as input for SIM Partial Training. This is because, even if each partial task is trained based on the decoded data of each guest party, the raw data of each guest party can be used when extracting features to be transmitted to the host party. Taking the method option DD of the regression target task as an example, there are three feature options: DD(R), DD(D), and DD(D)-Test(R). DD(R) extracts features based on the raw data of each guest party for the partial task trained based on the decoded data of each guest party and transmits them to the host party. DD(D) extracts features based on the decoded data of each guest party for a partial task trained based on the decoded data of

each guest party and transmits them to the host party. A final test or model operation can be performed on the results of the entire TT-VFDL-SIM which consists of the trained merging task utilizing each guest party's features and each previously trained partial task. In this final test, the raw data of each guest party can be applied to the partial task model trained according to the feature option DD(D) of method option DD. This is implemented with feature option DD(D)-Test(R).

Which method option or feature option is optimal for each target task basically depends on the data or task characteristics. Therefore, we derived the optimal method option or feature option through experiments in Section 5. In our experimental results, in the regression target task, the feature option DD(R) or DD(D)-Test(R) for the method option DD was generally effective in improving performance. In classification target tasks, feature option DR(R) or DR(D)-Test(R) for method option DR was generally effective in improving performance. In other words, training each partial task based on the decoded data of each guest party and applying the raw data of each guest party when using the trained partial task model was effective in improving performance. This means that transfer learning through a similar feature space is effective in improving performance.

3.3. Merging Task Training

Task-driven features can be extracted based on the FE part of the partial task model trained by the SIM Partial Training mechanism in each guest party. As shown in Figure 3, the task-driven features of each guest party are transmitted to the host party through one-shot communication. The merging task of the host party is trained to output the raw data of the host party based on the concatenated feature constructed by combining task-driven features from each guest party. Combination information between each guest party that was not considered when training the univariate partial task is trained through merging task training.

In general, horizontal federated learning implements federated learning by aggregating the weights trained by each client on the server. This is because each client generally trains a submodel with the same structure in common. However, in representative existing VFDL mechanisms, there is no submodel of the same structure that is commonly trained by each party. Therefore, weight aggregation is generally not performed in existing VFDL mechanisms.

In the proposed TT-VFDL mechanism based on the task-driven transfer approach, submodels with the same structure that are commonly trained between each guest party do not always exist. Therefore, basically, weight aggregation is not performed in merging task training. Instead, the merging task fine-tunes the target task based on task-driven features by training a merging task model with randomly initialized weights. This is because in the task-driven transfer approach, the target task is distributedly trained based on transfer learning between partial tasks derived from the target task and the merging task. However, unlike existing VFDL mechanisms, the TT-VFDL mechanism can sometimes define submodels with the same structure that are commonly trained between each guest party depending on the structure of the target task model. We are conducting future research on weight aggregation techniques for this case.

Meanwhile, merging task training is fine-tuning for the target task. Generally, in transfer learning, the learning rate of fine-tuning is set to a lower value than the learning rate in pre-training. Then what should be the learning rate for merging task training? Instead of uniformly setting the learning rate for merging task training to a small, fixed value, we derived an appropriate learning rate for each dataset and target task model through experimentation.

The reason why the learning rate of merging task training is not fixed to a specific value is because the feature space dimensions in pre-training and fine-tuning are different. In general transfer learning, pre-training and fine-tuning are performed based on feature spaces of the same dimension. On the other hand, in our merging task training, we fine-tune the target task in a multivariate feature space with more expanded dimensions than the

univariate feature space of each partial task. Since the impact of feature space dimension expansion on the learning rate of the fine-tuning task is not yet explained, this paper derived the appropriate learning rate of merging task training for each dataset and target task model through experiments.

4. TT-VFDL-SIM

TT-VFDL-SIM is a VFDL method designed based on the task-driven transfer approach in Section 3. Prior to distributed learning, the univariate partial task model structure of each guest party and the merging task model structure of the host party are derived from the multivariate target task based on the DFS Model Split technique. Furthermore, similar feature spaces are generated for training each partial task based on the LSTM-AE-based Data Augmentation technique.

The subsequent distributed learning mechanism is summarized in Figure 3. First, following the SIM Partial Training mechanism, the target task is pre-trained by training each partial task model using one of the selected similar feature spaces. The task-driven features of each guest party are transmitted to the host party through a one-shot communication mechanism. Finally, the target task is fine-tuned by training the host party's merging task based on the task-driven features of each guest party.

On the other hand, the composition of similar feature spaces for partial task training varies depending on the type of target task. As summarized in Table 4, similar feature spaces that can be introduced to the regression target task in SIM Partial Training are either DD or RD. Additionally, similar feature spaces that can be introduced in the classification target task are either DR or RR. Therefore, TT-VFDL-SIM has method options for each target task. Algorithms for the regression target task and classification target task were designed. Section 4.1 describes TT-VFDL-SIM for the regression target task, and Section 4.2 describes TT-VFDL-SIM for the classification target task.

4.1. TT-VFDL-SIM for Regression

Algorithm 1 is the pseudo code of TT-VFDL-SIM for the regression target task. In the preparation stage, decoded datasets to be used for input and output of each partial task model are created based on the LSTM-AE-based Data Augmentation technique (lines 1~9). The host party trains LSTM-AE to reconstruct its raw data. Based on this, a compressed dataset consisting of latent vectors for the host party's raw data is constructed and transmitted to each guest party. At this point, the weight of the decoder that can reconstruct this compressed dataset is also transmitted. Each guest party constructs a decoded dataset (YD) by applying a decoder to the host party's compressed dataset. In addition, each guest party trains LSTM-AE to reconstruct its own raw data (X). And based on this, a decoded dataset (XD), which consists of the decoded data for the raw data, is constructed. As a result, a group of input/output candidates, denoted as DD and RD, can be formed for the partial task model of each guest party.

Afterwards, each guest party trains its own partial task model based on the SIM Partial Training mechanism (lines 10~28). First, a partial task model structure is derived through the DFS Model Split technique. Then, one of the input/output candidates prepared in the preparation stage is assigned. In other words, one of the two method options can be chosen: DD and RD. Each guest party trains a partial task based on the selected method option and then transmits the task-driven feature dataset extracted from the FE part of the trained partial task model to the host party.

In the method option RD, task-driven features are extracted from the raw data (X) for each guest party based on the partial task model trained on their respective raw data (X). Meanwhile, in method option DD, one of two feature options is selected to extract task-driven features of each guest party (lines 26~27). Feature option DD(D) extracts features for the decoded data (XD) based on the partial task model trained on the decoded data (XD) of each guest party. On the other hand, feature option DD(R) extracts features for raw data (X) based on a partial task model trained on the decoded data (XD) of each guest party. Raw

data (X) never appeared in the training dataset of the partial task model. In other words, feature option DD(R) of method option DD is a type of transfer learning.

The host party fine-tunes the target task by training the merging task based on the task-driven features of guest parties (lines 29~37). Through the DFS Model Split technique, we derive the structure of the merging task model and allocate its inputs and outputs. The input is the task-driven features of each guest party, and the output is the raw data (Y) of the host party. If the SIM Partial Training mechanism's method option is DD and the feature option is DD(D) when training the partial task model at each guest party, then the host party has an opportunity to apply feature option DD(D)-Test(R) to the trained merging task model (lines 36~37).

Algorithm 1. TT-VFDL-SIMfor Regression

Preparation (for LSTM-AE based Data Augmentation)

Host Party

1 Construct a LSTM-AE with the encoder structure as e^0_{DA} and the decoder structure as d^0_{DA}

2 Train the LSTM-AE with objective function as $\underset{\delta^0_e,\delta^0_d}{\operatorname{argmin}} \sum_i L\left(d^0_{DA}\left(\delta^0_d; e^0_{DA}\left(\delta^0_e; Y_i\right)\right), Y_i\right)$

3 Get compressed dataset consist of $YC_i \leftarrow e^0_{DA}\left(Y_i; \delta^0_e\right)$

4 Send YC_i and d^0_{DA} to each Guest Party

Guest Party

5 Construct LSTM-AE with the Encoder structure as e^k_{DA} and the Decoder structure as d^k_{DA}

6 Train the LSTM-AE with objective function as $\underset{\delta^k_e,\delta^k_d}{\operatorname{argmin}} \sum_i L\left(d^k_{DA}\left(\delta^k_d; e^k_{DA}\left(\delta^k_e; X^k_i\right)\right), X^k_i\right)$

7 Receive YC_i and d^0_{DA} from the Host Party

8 Get decoded dataset consist of $YD_i \leftarrow d^0_{DA}\left(YC_i; \delta^0_d\right)$

9 Get decoded dataset consist of $XD^k_i \leftarrow d^k_{DA}\left(e^k_{DA}\left(X^k_i; \delta^k_e\right); \delta^k_d\right)$

TT-VFDL-SIM: Pre-training for the Target Task (SIMPartial Training)

Guest Party

10 **Partial Task Construction**

11 By **DFS Model Split**,

12 get the **model structure** of the Partial Task f^k consist of FE part e^k_{FE} and Split FM of the target task

13 By **LSTM-AE based Data Augmentation**, set the **input and output** of the Partial Task.

14 **IF Method Option == RD:**

15 Partial Task Input : X^k_i

16 Partial Task Output : $YD_i \leftarrow d^0_{DA}\left(YC_i; \delta^0_d\right)$ // prepared in the previous ***Preparation*** step

17 **ELIF Method Option == DD:**

18 Partial Task Input : $XD^k_i \leftarrow d^k_{DA}\left(e^k_{DA}\left(X^k_i; \delta^k_e\right); \delta^k_d\right)$

19 Partial Task Output : $YD_i \leftarrow d^0_{DA}\left(YC_i; \delta^0_d\right)$ // prepared in the previous ***Preparation*** step

20 **SIM Partial Training**

21 **IF Method Option == RD:**

22 Train the Partial Task with objective function as $\underset{\omega^k}{\operatorname{argmin}} \sum_i L\left(f^k\left(\omega^k; X^k_i\right), YD_i\right)$

23 Get Feature Data $F^k_i \leftarrow e^k_{FE}\left(X^k_i; \theta^k\right)$

24 **ELIF Method Option == DD:**

25 Train the Partial Task with objective function as $\underset{\omega^k}{\operatorname{argmin}} \sum_i L\left(f^k\left(\omega^k; XD^k_i\right), YD_i\right)$

26 **IF Feature Option** == **DD**(**R**) : Get Feature Data $F^k_i \leftarrow e^k_{FE}\left(X^k_i; \theta^k\right)$

27 **ELIF Feature Option** == **DD**(**D**) : Get Feature Data $F^k_i \leftarrow e^k_{FE}\left(XD^k_i; \theta^k\right)$

28 Send Feature Data F^k_i to the Host Party

TT-VFDL-SIM: Fine-tuning for the Target Task (Merging Task Training)

Host Party

29 Receive Feature Data F^k_i from each Guest Party

30 **Merging Task Construction**

31 By **DFS Model Split**, get the **model structure** of the Merging Task f^0 consist of FM part of the target task

32 Set **input** of Merging Task as the received each Guest Party's Feature Data $F^1_i, \ldots, F^K_i$

33 Set **output** of Merging Task as Y_i

34 **Merging Task Training**

35 Train the Merging Task with objective function as $\underset{\omega^0}{\operatorname{argmin}} \sum_i L\left(f^0\left(\omega^0; F^1_i, \ldots, F^K_i\right), Y_i\right)$

36 **IF** Guest Parties use **Method Option** as **DD** with **Feature Option** as **DD(D):**

37 **IF Feature Option** == **DD**(**D**)-**Test**(**R**) : $Y_i \leftarrow f^0\left(e^1_{FE}\left(X^1_i; \theta^1\right), \ldots, e^K_{FE}\left(X^K_i; \theta^K\right); \omega^0\right)$

In other words, task-driven features for X are extracted based on the partial task model trained on XD. And this can be applied to a merging task model trained based on task-driven features extracted from XD. The raw data (X) of each guest party never appear in the training process of the partial task model and merging task model. In other words, feature option DD(D)-Test(R) is a type of transfer learning. Additionally, feature option DD(D)-Test(R) is suitable for real-world scenarios. This is because by using X as input to a model trained and deployed based on XD, the overhead of creating XD for model input can be reduced.

4.2. TT-VFDL-SIM for Classification

Algorithm 2 is the pseudo code of TT-VFDL-SIM for the classification target task. In the preparation stage, decoded datasets to be used for input and output of each partial task model are created based on the LSTM-AE-based Data Augmentation technique (lines 1~5). The host party transmits its raw data, that is, encoded label data, to each guest party. Additionally, each guest party trains LSTM-AE to reconstruct its own raw data (X). And based on this, a decoded dataset (XD) for each raw dataset is constructed. As a result, a group of input/output candidates, denoted as DR and RR, can be formed for the partial task model of each guest party.

Algorithm 2. TT-VFDL-SIMfor Classification

Preparation (for LSTM-AE based Data Augmentation)
Host Party
1 Send encoded label Y_i to each Guest Party
Guest Party
2 Construct LSTM-AE with the encoder structure as e^k_{DA} and the decoder structure as d^k_{DA}
3 Train the LSTM-AE with objective function as $\underset{\delta^k_e,\delta^k_d}{\operatorname{argmin}} \sum_i L\left(d^k_{DA}\left(\delta^k_d; e^k_{DA}\left(\delta^k_e; X^k_i\right)\right), X^k_i\right)$
4 Receive Y_i from the Host Party
5 Get decoded dataset consist of $XD^k_i \leftarrow d^k_{DA}\left(e^k_{DA}\left(X^k_i; \delta^k_e\right); \delta^k_d\right)$

TT-VFDL-SIM: Pre-training for the Target Task (SIMPartial Training)
Guest Party
6 **Partial Task Construction**
7 By **DFS Model Split**,
8 get the **model structure** of the Partial Task f^k consist of FE part e^k_{FE} and Split FM the target task
9 By **LSTM-AE based Data Augmentation**, get the **input and output data** of the Partial Task.
10 **IF Method Option == RR:**
11 Partial Task Input : X^k_i
12 Partial Task Output : Y_i // prepared in the previous ***Preparation*** step
13 **ELIF Method Option == DR:**
14 Partial Task Input : $XD^k_i \leftarrow d^k_{DA}\left(e^k_{DA}\left(X^k_i; \delta^k_e\right); \delta^k_d\right)$
15 Partial Task Output : Y_i // prepared in the previous ***Preparation*** step
16 **SIM Partial Training**
17 **IF Method Option == RR:**
18 Train the Partial Task with objective function as $\underset{\omega^k}{\operatorname{argmin}} \sum_i L\left(f^k\left(\omega^k; X^k_i\right), Y_i\right)$
19 Get Feature Data $F^k_i \leftarrow e^k_{FE}\left(X^k_i; \theta^k\right)$
20 **ELIF Method Option == DR:**
21 Train the Partial Task with objective function as $\underset{\omega^k}{\operatorname{argmin}} \sum_i L\left(f^k\left(\omega^k; XD^k_i\right), Y_i\right)$
22 **IF Feature Option == DR(R):**
23 Get Feature Data $F^k_i \leftarrow e^k_{FE}\left(X^k_i; \theta^k\right)$
24 **ELIF Feature Option == DR(D):**
25 Get Feature Data $F^k_i \leftarrow e^k_{FE}\left(XD^k_i; \theta^k\right)$
26 Send Feature Data F^k_i to the Host Party

TT-VFDL-SIM: Fine-tuning for the Target Task (Merging Task Training)
Host Party
27 Receive Feature Data F^k_i from each Guest Party
28 **Merging Task Construction**
29 By **DFS Model Split**, get the **model structure** of the Merging Task f^0 consist of FM part of the target task
30 Set **input** of Merging Task as the received each Guest Party's Feature Data $F^1_i, \ldots, F^K_i$
31 Set **output** of Merging Task as Y_i
32 **Merging Task Training**
33 Train the Merging Task with objective function as $\underset{\omega^0}{\operatorname{argmin}} \sum_i L\left(f^0\left(\omega^0; F^1_i, \ldots, F^K_i\right), Y_i\right)$
34 **IF** Guest Parties use **Method Option** as **DR** with **Feature Option** as **DR(D):**
35 **IF Feature Option == DR(D)-Test(R):**
36 $Y_i \leftarrow f^0\left(e^1_{FE}\left(X^1_i; \theta^1\right), \ldots, e^K_{FE}\left(X^K_i; \theta^K\right); \omega^0\right)$

Afterwards, each guest party trains its own partial task model according to the SIM Partial Training mechanism (lines 6~26). First, a partial task model structure is derived through the DFS Model Split technique. Then, one of the input/output candidates prepared in the preparation stage is assigned. In other words, one of the two method options can be chosen: DR and RR. Each guest party trains a partial task based on the selected method option and then transmits the task-driven feature dataset extracted from the FE part of the trained partial task model to the host party.

In the method option RR, task-driven features are extracted from the raw data (X) for each guest party based on the partial task model trained on their respective raw data (X). Meanwhile, in method option DR, one of two feature options is selected to extract task-driven features of each guest party (lines 22~25). Feature option DR(D) extracts features for the decoded data (XD) based on the partial task model trained on the decoded data (XD) of each guest party. On the other hand, feature option DR(R) extracts features for raw data (X) based on a partial task model trained on the decoded data (XD) of each guest party. Raw data (X) never appeared in the training dataset of the partial task model. In other words, feature option DR(R) of method option DR is a type of transfer learning.

The host party fine-tunes the target task by training the merging task based on the task-driven features of guest parties (lines 27~36). Through the DFS Model Split technique, we derive the structure of the merging task model and allocate its inputs and outputs. The input is the task-driven features of each guest party, and the output is the raw data (Y) of the host party, that is, encoded label data. If the SIM Partial Training mechanism's method option is DR and the feature option is DR(D) when training the partial task model at each guest party, then the host party has an opportunity to apply feature option DR(D)-Test(R) to the trained merging task model (lines 35~36).

In other words, task-driven features for X are extracted based on the partial task model trained on XD. And this can be applied to a merging task model trained based on task-driven features extracted from XD. The raw data (X) of each guest party never appear in the training process of the partial task model and merging task model. In other words, feature option DR(D)-Test(R) is a type of transfer learning. Additionally, feature option DR(D)-Test(R) is suitable for real-world scenarios. This is because by using X as input to a model trained and deployed based on XD, the overhead of creating XD for model input can be reduced.

5. Experiments and Discussion

In this section, we analyze the performance of TT-VFDL-SIM and validate the suitability of the task-driven transfer approach through experiments. In Section 5.2, prior to analyzing the performance of TT-VFDL-SIM, we summarize the results of applying the LSTM-AE-based Data Augmentation technique to each dataset to create similar feature spaces required for TT-VFDL-SIM. In other words, we summarize similar feature spaces created in the process of implementing TT-VFDL-SIM for the experiments in this section. Section 5.3 compares the representative performance metrics and communication amount of TT-VFDL-SIM and other comparison methods for the regression target task and classification target task, respectively. In Section 5.4, we validate the suitability of each stage that constitutes the task-driven transfer approach based on detailed experimental results for TT-VFDL-SIM.

5.1. Overview of the Experiments

5.1.1. Dataset Preparation

Table 5 is an overview of the five IoT time-series datasets used in the experiment. All five datasets are multivariate IoT time-series datasets consisting of four streams. For each dataset, a multivariate target task with three streams as input and one stream as output was constructed and used in the experiment. AIRQUAL, HOMENV, and GASVOLT are datasets for the regression target task, and ACCHAR and TRAFFIC are datasets for the classification target task.

Table 5. Datasets used in the experiments.

Dataset	Task	FE Structure	Pattern Length			Train Size	Validation Size	Test Size
			Input	Feature	Output			
AIRQUAL	Regression	CNN-LSTM	24	10	1	504	72	144
HOMENV	Regression	LSTM	72		1	504	72	144
GASVOLT	Regression	CNN-LSTM	70		1	490	70	140
ACCHAR	Classification	CNN	204		4	168	24	83
TRAFFIC	Classification	CNN-LSTM	144		7	294	28	118

Five datasets were constructed from data sources donated by various data providers to be used as benchmarks for machine learning [36,37]. The benchmark data in [36,37] are the de facto official data used in time series research. We also selected data sources suitable for our research goals from [36,37] and constructed five datasets for the experiment. In particular, all datasets selected for experiments are IoT time-series data collected in the real world. We organized representative detailed domains or sensor types, tasks, and data characteristics in the IoT field through existing studies related to multivariate IoT time-series analysis and all IoT time-series datasets in [36,37]. And the representative datasets in Table 5 that can cover all of them as much as possible are selected.

Four streams were selected from each of the original data sources [38–40], and regression datasets of AIRQUAL, HOMENV, and GASVOLT were constructed by sliding windowing the selected streams into a window of the size described in the Pattern Length column in Table 5. For the two classification datasets, ACCHAR and TRAFFIC, we selected three streams to be used as inputs from the original data sources [41,42] and constructed patterns of the size specified in the Pattern Length column of Table 5.

We performed necessary preprocessing such as scaling, interpolation, and noise removal for each dataset in Table 5. In particular, each training dataset of ACCHAR and TRAFFIC for the classification target task was reorganized to have equal class distribution. The reason for reorganizing the training datasets of ACCHAR and TRAFFIC into a balanced dataset is to exclude as much as possible the data-driven characteristics that may affect the performance of the target task in addition to the mechanism of each VFDL method. As a result, it was confirmed that precision, recall, sensitivity, specificity, and accuracy had similar distributions of values in all experiments on the classification target task.

5.1.2. Target Tasks

We constructed a multivariate target task for the experiment from each dataset in Table 5. AIRQUAL is a dataset that measures the concentration of air pollutants [38]. Based on this, a regression target task was created to predict the concentration of benzene from changes in the concentrations of non-methane hydrocarbon, carbon monoxide, and ozone. HOMENV is a dataset that measures changes in the indoor environment of a home and changes in external weather [39]. We created a regression target task to predict the temperature of the laundry room based on the temperature change in the parents' room and kitchen located on different floors and the external temperature change. GASVOLT is a dataset that monitors the state of a gas chamber [40]. We constructed a regression target task to predict the heater voltage of a gas chamber from three streams of gas concentration changes measured from two different gas sensors.

ACCHAR is a human activity recognition (HAR) dataset that classifies four movements including epileptic seizures [41]. The inputs to this classification target task are each X, Y, and Z streams of a 3-axis accelerometer, and the output is an encoded label stream for the four motion classes. TRAFFIC was constructed from the traffic volume dataset [42] measured by loop sensors installed on each road in California. The classification target task predicts encoded labels for 7 days of the week from Monday to Sunday using 3 traffic streams randomly selected in [42] as input.

5.1.3. Target Task Models

We constructed three types of multivariate deep learning models as shown in Figure 1 for the target tasks derived from each dataset and used them as target task models for experiments. In other words, for each dataset, three target task models were constructed by varying only the FM part structure for the same FE part structure. The three target task models for each dataset can be distinguished according to the FM part structure, and the FM part structures are SLFM, MLFM, and MFCMLFM as shown in Figure 1. The FE part structure for each dataset was constructed as summarized in the FE Structure column of Table 5.

Through preliminary experiments based on centralized training, the type of FE part structure optimized for each dataset was determined, and the values of various hyperparameters related to the specific structure of each target task model were determined. In other words, the type of FE part structure for each dataset, detailed layer-by-layer design of each target task model, and various hyperparameter values were determined to ensure the best performance in centralized training.

5.1.4. Target System Architecture and Comparison Methods

The target system architecture of TT-VFDL-SIM is a completely distributed architecture in which each input and output are distributed. As shown in Figure 2, TT-VFDL-SIM and other comparison methods were implemented assuming a completely distributed architecture consisting of three guest parties and one host party. There are a total of four other comparison methods for evaluating the performance of TT-VFDL-SIM, as shown in Figure 2. CNTR is a centralized training mechanism. P-VFDL-SL, P-VFDL-SLG, and DT-VFDL-AE are three representative mechanisms of the existing VFDL methods introduced in Section 2.4.

Table 6 summarizes the characteristics of the comparison methods used in the experiment and the proposed method. This study was conducted from the perspective of the VFDL mechanism, and the goal is to design a new VFDL mechanism suitable for the characteristics of the IoT domain, unlike the existing VFDL mechanism. Therefore, existing studies that can represent existing VFDL methods from a VFDL mechanism perspective were selected as comparison methods. From a VFDL mechanism perspective, not only the existing [2,3,8–16] but also the recent VFDL methods [4–7] belong to one of P-VFDL-SL, P-VFDL-SLG, and DT-VFDL-AE in Table 6.

P-VFDL-SL applies Split Learning [2] to our target system architecture. The submodel of each guest party is the FE part for each input of the target task model, and the submodel of the host party is the FM part of the target task model. P-VFDL-SL parallelizes the forward propagation and back propagation that constitute the training loop through each guest party and host party. The mechanism of existing VFDL methods [2,11–15] of the training loop parallelization approach is generally based on Split Learning [2]. Therefore, P-VFDL-SL is a representative mechanism of existing VFDL methods based on the training loop parallelization approach. P-VFDL-SLG implemented [16], which improved performance by adding a local gradient update mechanism to Split Learning, to suit the experimental scenario.

DT-VFDL-AE is a representative mechanism of existing VFDL methods of the data-driven transfer approach [3,8–10]. Each guest party trains data-driven features through unsupervised representation learning and transfers them to the host party. Since autoencoders are generally used for the unsupervised representation learning of each guest party, each guest party in DT-VFDL-AE also trains an autoencoder. This autoencoder is based on an encoder with the same structure as the FE part structure of the target task model for each guest party's input.

Table 6. Brief comparison of TT-VFDL-SIM and other comparison methods.

Related Study	Methods	Approach	Communication	Trained Features	Strategy
Proposed	TT-VFDL-SIM *	Task-driven Transfer Approach *(Proposed)*	One-shot Communication	Task-driven Features for Univariate Partial Task Pre-trained on Similar Feature Spaces	Transfer Learning Univariate to Multivariate and Transfer Learning Between Similar Feature Spaces
Implementation of Split Learning [2] Mechanism of Existing Studies of [2,11–15]	P-VFDL-SL	Training Loop Parallelization Approach	Communication Per Every Step	Task-driven Features for Multivariate Target Task	Parallelized Forward and Back Propagation
Implementation of [16]	P-VFDL-SLG			Task-driven Features for Multivariate Target Task Considering Univariate Partial Task	Parallelized Forward and Back Propagation and Local Gradient Update
Implementation of [3] Mechanism of Existing Studies of [3,8–10]	DT-VFDL-AE	Data-driven Transfer Approach	One-shot Communication	Data-driven Feature Trained by Autoencoder	Transfer Learning with Unsupervised Trained Representation
Generalized Multivariate Model of Existing Study of [17–21]	CNTR	Centralized Training	-	Task-driven Features for Multivariate Target Task	Centralized Training

* The proposed method.

5.1.5. Experimental Setup

In this paper, experiments were conducted on each of the five datasets in Table 5 based on three target task models composed of different types of FM parts to the same FE part. As summarized in Section 2.1, these target task models are a generalization of the multivariate deep learning model structure of existing studies on multivariate IoT time-series analysis. Through this, we attempted to conduct a performance analysis that was as generalized as possible and was not dependent on a specific target task model structure, at least in the multivariate IoT time-series domain.

As a result, a total of 9 target task models are constructed for the 3 regression datasets in Table 5, and a total of 6 target task models are constructed for the 2 classification datasets. The total of 15 generated (target task model, dataset) pairs are defined as independent experimental units. In the tables organizing the experimental results, the types of the datasets and FM parts are described so that each experimental unit can be distinguished into (FM Structure, Dataset) pairs. We compared the performance by applying CNTR, P-VFDL-SL, DT-VFDL-AE, and TT-VFDL-SIM to each experimental unit. All experiments were implemented in TensorFlow [43]. In addition, by applying early stopping, we trained to achieve optimal performance for each method applied to each experimental unit.

In particular, when applying TT-VFDL-SIM, three types of decoded data with different reconstruction performance were additionally introduced for each experimental unit. Therefore, the number of experimental units in TT-VFDL-SIM increased from 15 to 45. Because the reconstruction performance of decoded data depends on the Compression Ratio of LSTM-AE, each of the 45 experimental units can be distinguished into (FM Structure, Dataset, Compression Ratio) pairs. We applied all method options and feature options of TT-VFDL-SIM to each of these experimental units.

As described in Section 3.3, the learning rate of the merging task training when applying TT-VFDL-SIM to each experimental unit was determined through experiments. A

learning rate value of 0.001 is known to generally provide good performance in the Adam optimizer used in our experiments. We selected this value (0.001), a value larger than this value (0.01), and a value smaller than this value (0.0005) as learning rate candidates for merging task training. And the detailed options of TT-VFDL-SIM applied to each experimental unit were trained by changing the learning rate values of merging task training.

Meanwhile, as summarized in Section 5.1.1, since all training datasets of the classification datasets were composed of balanced datasets, accuracy and F1-Score were adopted as representative performance metrics among several performance metrics for the classification target task. In addition, MSE and R^2 were adopted as representative performance metrics for the regression target task. The subsequent experimental results are organized in separate tables for the regression target task and classification target task.

5.1.6. Summary of Experimental Results

The main experimental results of the subsequent experiments are briefly summarized as follows. Through this, the effectiveness of the task-driven transfer approach in the IoT domain was experimentally verified:

1. In Section 5.3, for both the regression target task and the classification target task, the average performance was good in the following order: proposed TT-VFDL-SIM, P-VFDL-SLG, CNTR, P-VFDL-SL, and DT-VFDL-AE.

- Task-driven features are more effective in improving performance than data-driven features.
- Task-driven features that also consider univariate partial tasks, such as P-VFDL-SLG, are more effective in improving performance than task-driven features that are optimized only for multivariate target tasks, such as P-VFDL-SL or CNTR.
- Ultimately, fine-tuning the multivariate target task based on task-driven features optimized only for each univariate partial task, such as TT-VFDL-SIM, is most effective in improving performance.

2. In Section 5.3, method option DD had higher average performance than method option RD in TT-VFDL-SIM for the regression target task. In TT-VFDL-SIM for the classification target task, method option RR and method option DR showed similar performance.

- Introducing similar feature spaces through a task-driven transfer approach is effective in improving performance.

3. Compared to P-VFDL-SL of the training loop parallelization approach, TT-VFDL-SIM reduced the communication amount required to achieve optimal performance by up to 721.87 times and at least 33.79 times.
4. In the detailed experiment results in Section 5.4, feature option DD(R) or DD(D)-Test(R) was effective in improving performance for the regression target task. For the classification target task, feature option DR(R) or DR(D)-Test(R) was effective in improving performance.

- In the task-driven transfer approach, not only transfer learning for the target task but also transfer learning between similar feature spaces for partial tasks is effective in improving performance.

5. The performance of TT-VFDL-SIM was the highest when introducing similar feature spaces generated from decoded data with reconstruction performance with an MSE of up to 0.00007 or less and R^2 of at least 0.99.

5.2. LSTM-AE-Based Data Augmentation for the TT-VFDL-SIM

One of the main strategies of the task-driven transfer approach for the design of TT-VFDL-SIM is to introduce similar feature spaces. We generated decoded data by applying the LSTM-AE-based Data Augmentation technique in Figure 5 to each IoT time-series

stream that consists of each dataset in the preparation stage of TT-VFDL-SIM, and the results are summarized in Table 7. Average MSE and average R^2 in Table 7 are the averages of MSE and R^2 for the decoded data generated from streams that consist of each dataset, and they refer to the reconstruction performance of the decoded data from each raw dataset.

Table 7. Performance of the LSTM-AE-based Data Augmentation for each dataset by Compression Ratio.

Dataset	Original Window Size	Tumbling Window Size	Latent Dimension	Compressed Window Size	$\frac{Compressed\ W.}{Original\ W.}$	Compression Ratio	Average MSE	Average R^2
AIRQUAL	24	6	2	8	0.33	3.00	0.00081	0.96
			3	12	0.50	2.00	0.00017	0.99
			5	20	0.83	1.20	0.00007	1.00
HOMENV	72	6	2	24	0.33	3.00	0.00004	0.99
			3	36	0.50	2.00	0.00002	1.00
			5	60	0.83	1.20	0.00002	1.00
GASVOLT	70	5	1	14	0.20	5.00	0.00163	0.95
			3	42	0.60	1.67	0.00004	1.00
			4	56	0.80	1.25	0.00002	1.00
ACCHAR	204	6	2	68	0.33	3.00	0.00149	0.90
			3	102	0.50	2.00	0.00007	1.00
			5	170	0.83	1.20	0.00002	1.00
TRAFFIC	144	6	2	48	0.33	3.00	0.00008	0.96
			3	72	0.50	2.00	0.00003	0.99
			5	120	0.83	1.20	0.00002	0.99

In order to generate similar feature spaces, the reconstruction performance of decoded data generated from raw data needs to be high. To achieve this, as summarized in Figure 5, the original data patterns of each dataset are split into small-sized sub-patterns by tumbling windowing. The Original Window Size in Table 7 is the length of the original data pattern of each dataset, and the Tumbling Window Size is the length of the sub-pattern generated by tumbling windowing. Latent Dimension in Table 7 is the latent vector size of LSTM-AE for the sub-pattern, and Compressed Window Size is the latent vector size of LSTM-AE for one original data pattern.

To analyze the impact of similar feature spaces on TT-VFDL-SIM, three decoded datasets with different reconstruction performance were generated for each dataset. And the three decoded datasets were introduced into the experiment. The reconstruction performance of decoded data is related to the latent vector size of LSTM-AE. We adjusted the degree to which raw data were compressed by changing the dimension of the latent vector of LSTM-AE. Compression Ratio in Table 7 is the ratio of Original Window Size to Compressed Window Size. In other words, the larger the Compression Ratio, the more the raw data are compressed and then reconstructed.

In Table 7, for all datasets, the smaller the Compression Ratio, the smaller the average MSE and the larger the average R^2, and the best reconstruction performance is shown when the Compression Ratio is the smallest. In other words, the less the raw data are compressed, the more similar decoded data can be generated to the raw data. In Table 7, HOMENV and GASVOLT have the highest reconstruction performance, with an average MSE of 0.00002 and an average R^2 of 1.00.

However, even in the worst reconstruction performance, average MSE and average R^2 are 0.00163 and 0.95, respectively. This is very high performance in terms of general regression tasks. Therefore, it can be seen that the reconstruction performance of decoded data generated by the LSTM-AE-based Data Augmentation technique applied to each

dataset is overall very excellent. Additionally, this also means that there is a baseline for similarity between similar feature spaces that can be introduced into TT-VFDL-SIM.

5.3. Performance Evaluation between the TT-VFDL-SIM and Other Comparison Methods

In this section, we compare the performance of TT-VFDL-SIM and other comparison methods. As described in Section 5.1.5, TT-VFDL-SIM was applied to 45 experimental units distinguished into (FM Structure, Dataset, Compression Ratio). Other comparison methods were applied to 15 experimental units distinguished into (FM Structure, Dataset). The highest performance for each method is selected and organized as the main performance for 15 experimental units distinguished into (FM Structure, Dataset). Detailed experimental results for various detailed conditions of TT-VFDL-SIM will be summarized in Section 5.4.

By comparing metric values of the main performance for each method, TT-VFDL-SIM and other comparison methods are macroscopically analyzed as follows:

- Compared to approaches of existing VFDL methods, experimental verification of the suitability of the task-driven transfer approach, which was used in designing the TT-VFDL-SIM, for the multivariate IoT time-series domain.
- Experimental verification of the effectiveness of partial task construction of the task-driven transfer approach.
- Experimental verification of the effectiveness of generating and introducing similar feature spaces in a task-driven transfer approach.

5.3.1. Experimental Results

Tables 8 and 9 summarize the main performances of TT-VFDL-SIM, DT-VFDL-AE, P-VFDL-SL, P-VFDL-SLG, and CNTR for the regression target task and classification target task, respectively. The main performance for each method was measured for each experimental unit of (FM Structure, Dataset). Among the methods applied to each experimental unit, the method with the highest performance is marked in red, and the method with the second highest performance is marked in blue. Methods that show the same or improved performance as centralized training (CNTR) are indicated in bold.

For the regression target tasks in Table 8, both method options RD and DD of TT-VFDL-SIM showed higher performance than CNTR. In addition, TT-VFDL-SIM showed the highest performance for all experimental units. The method with the second highest performance was also TT-VFDL-SIM. In addition, method option DD of TT-VFDL-SIM showed higher performance than method option RD in all experimental units except two experimental units. DT-VFDL-AE showed higher performance than CNTR for only 4 experimental units. P-VFDL-SL showed higher performance than CNTR for only 3 experimental units. However, they all showed lower performance than TT-VFDL-SIM. The MSE of TT-VFDL-SIM decreased by up to 0.00181 compared to CNTR and decreased by up to 0.00188 and 0.01143 compared to P-VFDL-SL and DT-VFDL-AE, respectively.

Regarding the classification target tasks in Table 9, both method options RR and DR of TT-VFDL-SIM showed higher performance than CNTR for all experimental units except one. In addition, TT-VFDL-SIM showed the highest performance for all experimental units except for the one experimental unit where CNTR showed the highest performance. The method with the second highest performance was also mostly TT-VFDL-SIM. In the classification target task, method option RR and method option DR of TT-VFDL-SIM showed similarly good performance.

For some experimental units of the classification target task, CNTR or P-VFDL-SL showed the second highest performance. However, the performance of CNTR or P-VFDL-SL was not higher than that of TT-VFDL-SIM for the same experimental unit. DT-VFDL-AE performed lower than CNTR for all experimental units. P-VFDL-SL outperformed CNTR for some experimental units. The accuracy of TT-VFDL-SIM increased by up to 4.82% compared to CNTR and increased by up to 4.82% and 33.73% compared to P-VFDL-SL and DT-VFDL-AE, respectively.

Table 8. Best performance of the proposed method and other comparison methods for the regression task.

FM Structure	Dataset	Metric	TT-VFDL-SIM *		DT-VFDL-AE	P-VFDL-SL	P-VFDL-SLG	CNTR
			(RD)	(DD)				
MFCMLFM	AIRQUAL	**MSE**	0.00372	0.00382	0.00427	0.00424	0.00429	0.00422
		R^2	0.75	0.74	0.71	0.71	0.71	0.71
	HOMENV	**MSE**	0.00129	0.00104	**0.00229**	0.00259	**0.00223**	0.00240
		R^2	0.37	0.49	**−0.12**	−0.27	**−0.10**	−0.18
	GASVOLT	**MSE**	0.00373	0.00328	0.00783	0.00491	**0.00449**	0.00459
		R^2	0.98	0.98	0.95	0.97	**0.97**	0.97
MLFM	AIRQUAL	**MSE**	0.00385	0.00357	**0.00409**	**0.00423**	**0.00418**	0.00441
		R^2	0.74	0.76	**0.72**	**0.71**	**0.72**	0.70
	HOMENV	**MSE**	0.00106	0.00100	**0.00225**	0.00288	**0.00227**	0.00281
		R^2	0.48	0.51	**−0.10**	−0.41	**−0.12**	−0.38
	GASVOLT	**MSE**	0.00395	0.00398	0.00789	0.00489	**0.00432**	0.00479
		R^2	0.98	0.98	0.95	0.97	**0.98**	0.97
SLFM	AIRQUAL	**MSE**	0.00375	0.00353	0.00446	0.00445	0.00451	0.00442
		R^2	0.75	0.76	0.70	0.70	0.69	0.70
	HOMENV	**MSE**	0.00182	0.00119	**0.00216**	**0.00227**	**0.00220**	0.00235
		R^2	0.11	0.42	**−0.06**	**−0.12**	**−0.08**	−0.15
	GASVOLT	**MSE**	0.00428	0.00353	0.01496	**0.00447**	**0.00451**	0.00473
		R^2	0.98	0.98	0.91	**0.97**	**0.97**	0.97

* The proposed method.

Table 9. Best performance of the proposed method and other comparison methods for the classification task.

FM Structure	Dataset	Metric	TT-VFDL-SIM *		DT-VFDL-AE	P-VFDL-SL	P-VFDL-SLG	CNTR
			(RR)	(DR)				
MFCMLFM	ACCHAR	**Accuracy (%)**	98.80	98.80	85.54	95.18	95.18	96.39
		F1-Score	0.99	0.99	0.86	0.95	0.95	0.96
	TRAFFIC	**Accuracy (%)**	95.76	94.92	85.59	95.76	**96.61**	96.61
		F1-Score	0.96	0.94	0.85	0.96	**0.97**	0.97
MLFM	ACCHAR	**Accuracy (%)**	97.59	98.80	72.29	95.18	97.59	96.39
		F1-Score	0.98	0.99	0.55	0.95	0.98	0.96
	TRAFFIC	**Accuracy (%)**	98.31	**95.76**	85.59	**95.76**	97.46	94.92
		F1-Score	0.98	**0.96**	0.87	**0.96**	0.97	0.95
SLFM	ACCHAR	**Accuracy (%)**	98.80	100.00	66.27	**95.18**	93.98	95.18
		F1-Score	0.99	1.00	NaN	**0.95**	0.95	0.95
	TRAFFIC	**Accuracy (%)**	97.46	96.61	72.03	**95.76**	**95.76**	94.07
		F1-Score	0.97	0.96	0.64	**0.96**	**0.96**	0.94

* The proposed method.

Table 10 compares the communication amount of TT-VFDL-SIM, DT-VFDL-AE, and P-VFDL-SL. The "Ratio between P-VFDL-SL and T-VFDL" column shows the ratio of the communication amount between P-VFDL-SL and TT-VFDL-SIM or DT-VFDL-AE. P-VFDL-SL takes the training loop parallelization approach and TT-VFDL-SIM and DT-VFDL-AE take the transfer learning-based approach. The case with the largest difference is marked in red, and the case with the smallest difference is marked in blue. As described in Section 5.1.5, early stopping was applied to training for each experimental unit. Therefore, even for the

same experimental unit, the number of epochs for each method is different. However, each method was trained with the number of epochs that can achieve optimal performance through early stopping. Therefore, the communication amount summarized in Table 10 means the communication amount required for optimal performance of each method.

Table 10. Communication amount.

FM Structure	Dataset	Task-Driven Transfer		Data-Driven Transfer	Training Loop Parallelization	Ratio between	
		TT-VFDL-SIM *		DT-VFDL-AE	P-VFDL-SL	P-VFDL-SL	P-VFDL-SL
		Preparation	Total				
		Communication Amount (Bytes)				DT-VFDL-AE	TT-VFDL-SIM *
MFCMLFM	AIRQUAL	15,360	257,280	241,920	30,373,056	125.55	118.05
	HOMENV	15,360	257,280	241,920	19,248,768	79.57	74.82
	GASVOLT	14,361	249,561	235,200	37,566,144	159.72	150.53
	ACCHAR	4452	96,852	92,400	22,579,200	244.36	233.13
	TRAFFIC	10,392	158,232	147,840	114,222,528	772.61	721.87
MLFM	AIRQUAL	15,360	257,280	241,920	29,248,128	120.90	113.68
	HOMENV	15,360	257,280	241,920	8,692,992	35.93	33.79
	GASVOLT	14,361	249,561	235,200	56,814,912	241.56	227.66
	ACCHAR	4452	96,852	92,400	31,997,952	346.30	330.38
	TRAFFIC	10,392	158,232	147,840	85,034,880	575.18	537.41
SLFM	AIRQUAL	15,360	257,280	241,920	32,622,912	134.85	126.80
	HOMENV	15,360	257,280	241,920	52,468,416	216.88	203.94
	GASVOLT	14,361	249,561	235,200	69,543,936	295.68	278.67
	ACCHAR	4452	96,852	92,400	21,805,056	235.99	225.14
	TRAFFIC	10,392	158,232	147,840	81,817,344	553.42	517.07

* The proposed method.

For the same experimental unit, the communication amount of TT-VFDL-SIM and DT-VFDL-AE which take the transfer learning-based approach were noticeably smaller than the communication amount of P-VFDL-SL which takes the training loop parallelization approach. For the same experimental unit, TT-VFDL-SIM could reduce the communication amount required for optimal performance by at least 33.79 times and up to 721.87 times compared to P-VFDL-SL. Even the communication amount occurring in one step of each epoch of P-VFDL-SL was larger than the total communication amount of TT-VFDL-SIM or DT-VFDL-AE. This is because of gradients transmitted to each guest party in back propagation. The reason why TT-VFDL-SIM requires a slightly larger communication amount than DT-VFDL-AE is because communication occurs for the generation of similar feature spaces in the preparation stage.

5.3.2. Discussion

The experimental results in Section 5.3.1 support that the task-driven transfer approach, which uses transfer learning based on task-driven features, is more suitable for multivariate IoT time-series analysis than existing VFDL design approaches. This can be explained as due to the characteristics of the multivariate IoT time-series domain. In the multivariate IoT time-series analysis scenario, each variable is information that reflects the specific state in which the task is defined. Therefore, in multivariate IoT time-series analysis, it may be suitable to leverage task-driven features rather than data-driven features. DT-VFDL-AE leverages data-driven features, while CNTR, P-VFDL-SL, and TT-VFDL-SIM leverage task-driven features. As a result, DT-VFDL-AE had relatively lower performance than CNTR, P-VFDL-SL, and TT-VFDL-SIM. This experimentally supports that task-driven features are more suitable than data-driven features in multivariate IoT time-series analysis.

The task-driven transfer approach derives univariate partial tasks from a multivariate target task based on the DFS Model Split technique to train task-driven features. In

other words, the task-driven transfer approach trains a multivariate target task based on task-driven features optimized for univariate partial tasks. On the other hand, CNTR and P-VFDL-SL train task-driven features suitable for multivariate target tasks. The experimental results in Section 5.3.1 show that the task-driven transfer approach is more effective. The variables in the scenario of multivariate IoT time-series analysis are dependent on each other because they are created simultaneously from a specific state and are linearly or non-linearly related to each other. Therefore, in TT-VFDL-SIM, task-driven features suitable for each univariate partial task can also be used for multivariate target task training.

Another strategy of the task-driven transfer approach is to introduce similar feature spaces to ensure basic security and prevent overfitting for univariate partial tasks. In TT-VFDL-SIM, RD and DD were introduced for the regression target task, and RR and DR were introduced for the classification target task. And because the performance of TT-VFDL-SIM was excellent for all similar feature spaces, it can be said that the introduction of similar feature spaces contributes to improving the performance of TT-VFDL-SIM.

Additionally, comparing the method options of TT-VFDL-SIM, the performance of DD was better than RD in the regression target task, and the performance of DR and RR was similarly good in the classification target task. In other words, a similar feature space based on decoded data helps improve the overall performance of TT-VFDL-SIM than a similar feature space based on raw data. These experimental results also support that similar feature spaces contribute to the performance improvement of TT-VFDL-SIM.

However, in TT-VFDL-SIM for the classification target task, DR tended to show relatively high performance for ACCHAR and RR for TRAFFIC. In addition to differences in domains, the two datasets also had differences in correlation coefficient values between input variables. In TRAFFIC, the correlation coefficient between each input variable was very high at 0.82, but in ACCHAR, the correlation coefficient between each input variable was very low at 0.04. In the regression target task, the correlation coefficients between the input variables of AIRQUAL, HOMENV, and GASVOLT were 0.88, 0.17, and 0.96, respectively. But it was advantageous to introduce a similar feature space based on decoded data in most experimental units. Therefore, it is still difficult to conclude that the lower the correlation coefficient between input variables in each dataset, the more advantageous it is to introduce a similar feature space based on decoded data.

As a result, TT-VFDL-SIM was designed based on strategies of a task-driven transfer approach suitable for the characteristics of multivariate IoT time-series analysis, greatly improving performance and significantly reducing the communication amounts required for distributed learning. In particular, the strategy of leveraging task-driven features extracted for univariate partial tasks for merging task training and the strategy of introducing similar feature spaces for security and overfitting prevention were a great help to improve the performance of TT-VFDL-SIM. Ultimately, the task-driven transfer approach combines the advantages of communication efficiency from transfer learning-based approaches with the performance enhancement benefits of task-driven features. This makes the TT-VFDL-SIM a suitable VFDL design technique for multivariate IoT time-series analysis.

5.4. Detailed Analysis for the TT-VFDL-SIM

In this section, the operational principles of each step of TT-VFDL-SIM which was designed leveraging the task-driven transfer approach are experimentally validated. These are based on the detailed experimental results of applying TT-VFDL-SIM to 45 experimental units distinguished into (FM Structure, Dataset, Compression Ratio). Additionally, based on this analysis, we summarize the contribution of the transfer learning mechanism in the task-driven transfer approach. The contents to be analyzed in this section are summarized as follows:

- Impact of similarity between similar feature spaces on the performance of TT-VFDL-SIM.
- Experimental verification of the effect of introducing similar feature spaces using the SIM Partial Training mechanism of TT-VFDL-SIM.
- Learning rate setting problem for merging task training in TT-VFDL-SIM.

- Relationship between partial task training performance and merging task training performance in TT-VFDL-SIM.
- Contribution of transfer learning mechanism in task-driven transfer approach.

5.4.1. Experimental Results

For the regression target task, the detailed experimental results of applying TT-VFDL-SIM with the method option RD are summarized in Table 11, and the detailed experimental results of applying TT-VFDL-SIM with the method option DD are summarized in Table 12. The experiment was conducted on experimental units (FM Structure, Dataset, Compression Ratio). Compression Ratio is used to distinguish the reconstruction performance of decoded data for each dataset and is already summarized in Table 7. The SIM Partial Training (method option) column summarizes the average performance of SIM Partial Training in three guest parties. The Merging Task Training column summarizes the performance of merging task training in the host party, which is the performance when the target task is trained with TT-VFDL-SIM.

Table 11. Performance of the TT-VFDL-SIM with method option as RD (for regression) under various conditions.

FM Structure	Dataset	Compression Ratio	SIM Partial Training (RD) *		Merging Task Training		
			MSE	R^2	MSE	R^2	Best Learning Rate
MFCMLFM	AIRQUAL	**1.20**	**0.00765**	**0.47**	**0.00372**	**0.75**	**0.01**
		2.00	0.00645	0.56	**0.00376**	**0.75**	0.0005
		3.00	0.00745	0.49	**0.00389**	**0.74**	0.0005 0.001
	HOMENV	**1.20**	**0.00338**	**−0.60**	**0.00129**	**0.37**	**0.001**
		2.00	0.00346	−0.80	**0.00148**	**0.27**	0.01
		3.00	0.00257	−0.35	**0.00133**	**0.35**	0.01
	GASVOLT	**1.25**	**0.00520**	**0.97**	**0.00373**	**0.98**	**0.01**
		1.67	0.00547	0.97	**0.00378**	**0.98**	0.01
		5.00	0.00569	0.97	**0.00396**	**0.98**	0.01
MLFM	AIRQUAL	**1.20**	**0.00760**	**0.47**	**0.00385**	**0.74**	**0.0005**
		2.00	0.00528	0.63	**0.00406**	**0.72**	0.001
		3.00	0.00814	0.45	**0.00410**	**0.72**	0.001
	HOMENV	**1.20**	**0.00265**	**−0.28**	**0.00106**	**0.48**	**0.0005**
		2.00	0.00284	−0.46	**0.00108**	**0.47**	0.01
		3.00	0.00293	−0.52	**0.00137**	**0.33**	0.01
	GASVOLT	**1.25**	**0.00484**	**0.97**	**0.00395**	**0.98**	**0.001**
		1.67	0.00475	0.97	**0.00420**	**0.98**	0.01
		5.00	0.00476	0.97	**0.00441**	**0.97**	0.01
SLFM	AIRQUAL	**1.20**	**0.00862**	**0.40**	**0.00375**	**0.75**	**0.0005**
		2.00	0.00517	0.64	**0.00379**	**0.74**	0.001
		3.00	0.00762	0.48	**0.00386**	**0.74**	0.0005
	HOMENV	**1.20**	**0.00287**	**-0.37**	**0.00182**	**0.11**	**0.0005**
		2.00	0.00264	−0.37	**0.00192**	**0.06**	0.001
		3.00	0.00253	−0.33	**0.00212**	**−0.04**	0.01
	GASVOLT	**1.25**	**0.00531**	**0.97**	**0.00428**	**0.98**	**0.01**
		1.67	0.00465	0.97	**0.00431**	**0.98**	0.01
		5.00	0.00618	0.97	**0.00444**	**0.97**	0.001

* MSE and R^2 values of the SIM Partial Training are the averaged values of all guest parties which are participating in the TT-VFDL-SIM (for regression).

Table 12. Performance of the TT-VFDL-SIM with method option as DD (for regression) under various conditions.

FM Structure	Dataset	Compression Ratio	SIM Partial Training (DD) *		Merging Task Training			Best Feature Option
			MSE	R^2	MSE	R^2	Best Learning Rate	
MFCMLFM	AIRQUAL	**1.20**	**0.00525**	**0.63**	**0.00382**	**0.74**	**0.01**	**DD(D)**
		2.00	0.00730	0.50	**0.00407**	**0.72**	0.001	DD(D)
		3.00	0.00780	0.46	**0.00404**	**0.73**	0.0005	DD(R)
	HOMENV	**1.20**	**0.00269**	**−0.29**	**0.00104**	**0.49**	**0.0005**	**DD(R)**
		2.00	0.00274	−0.42	**0.00161**	**0.21**	0.01	DD(R)
		3.00	0.00233	−0.23	**0.00172**	**0.16**	0.001	DD(D)-Test(R)
	GASVOLT	**1.25**	**0.00620**	**0.97**	**0.00328**	**0.98**	**0.01**	**DD(D)-Test(R)**
		1.67	0.00439	0.98	**0.00390**	**0.98**	0.01	DD(D)
		5.00	0.00610	0.97	**0.00403**	**0.98**	0.0005	DD(D)
MLFM	AIRQUAL	**1.20**	**0.00969**	**0.33**	**0.00357**	**0.76**	**0.0005**	**DD(R)**
		2.00	0.00726	0.51	**0.00379**	**0.74**	0.0005	DD(R) DD(D)-Test(R)
		3.00	0.00753	0.47	**0.00394**	**0.73**	0.0005	DD(R)
	HOMENV	**1.20**	**0.00261**	**−0.27**	**0.00100**	**0.51**	**0.01**	**DD(D) DD(D)-Test(R)**
		2.00	0.00306	−0.57	**0.00104**	**0.49**	0.01	DD(D)-Test(R)
		3.00	0.00246	−0.30	**0.00117**	**0.42**	0.01	DD(D)-Test(R)
	GASVOLT	**1.25**	**0.00643**	**0.96**	**0.00398**	**0.98**	**0.0005**	**DD(R)**
		1.67	0.00488	0.97	**0.00402**	**0.98**	0.01	DD(R)
		5.00	0.00673	0.96	**0.00426**	**0.98**	0.0005	DD(D)
SLFM	AIRQUAL	**1.20**	**0.00780**	**0.46**	**0.00353**	**0.76**	**0.001**	**DD(R)**
		2.00	0.00829	0.44	**0.00367**	**0.75**	0.001	DD(R)
		3.00	0.00530	0.61	**0.00367**	**0.75**	0.0005	DD(R)
	HOMENV	**1.20**	**0.00255**	**−0.22**	**0.00119**	**0.42**	**0.0005**	**DD(R)**
		2.00	0.00242	−0.26	**0.00154**	**0.24**	0.001	DD(R)
		3.00	0.00271	−0.40	**0.00150**	**0.27**	0.0005	DD(R)
	GASVOLT	**1.25**	**0.00509**	**0.97**	**0.00353**	**0.98**	**0.0005**	**DD(R)**
		1.67	0.00561	0.97	**0.00433**	**0.97**	0.01	DD(R)
		5.00	0.00803	0.95	**0.00460**	**0.97**	0.01	DD(R)

* MSE and R^2 values of the SIM Partial Training are the averaged values of all guest parties which are participating in the TT-VFDL-SIM (for regression).

As described in Section 5.1.5, we experimented by changing the learning rate of merging task training to 0.01, 0.001, and 0.0005 for each experimental unit, and the learning rate that achieved the highest performance was organized in the Best Learning Rate column. The experimental results for (FM Structure, Dataset, Compression Ratio) showing the highest performance for each (FM Structure, Dataset) are shown in red. These are summarized as main performance for each (FM Structure, Dataset) in Table 8. In addition, the merging task training performance, which is higher than that of CNTR for the same experimental unit, is indicated in bold. Meanwhile, for Method option DD, there are three feature options for extracting task-driven features. The feature options that achieved the highest performance for each experimental unit are listed in the Best Feature Option column of Table 12.

According to Tables 11 and 12, for all the (FM Structure, Dataset, Compression Ratio), method option RD and DD showed higher performance than CNTR for the same (FM Structure, Dataset). Additionally, for the same (FM Structure, Dataset), the experimental unit with the smallest Compression Ratio had the highest performance. For the main performances of Table 8, that is, the performances marked in red in Tables 11 and 12, method

option DD was almost always higher than RD. For the remaining experimental units other than the main performances, method option DD also showed higher performance than RD. In addition, among the feature options of method option DD, DD(R) or DD(D)-Test(R) showed the best performance.

In merging task training, setting the learning rate of merging task training to 0.01, which is larger than the learning rate of the partial task training 0.001, or 0.0005, which is smaller than the learning rate of partial task training 0.001, tended to help improve performance. Meanwhile, for all experimental units, the performance of merging task training improved compared to the average performance of partial task training. Additionally, the correlation coefficient between the average performance of partial task training and the performance of merging task training for all experimental units were 0.76 and 0.83 for method options RD and DD, respectively.

For the classification target task, the detailed experimental results of applying TT-VFDL-SIM with the method option RR are summarized in Table 13, and the detailed experimental results of applying TT-VFDL-SIM with the method option DR are summarized in Table 14. The experiment was conducted on experimental units (FM Structure, Dataset, Compression Ratio), and the column configuration and specifications of each table are same as Tables 11 and 12.

According to Tables 13 and 14, for most of (FM Structure, Dataset, Compression Ratio) except experimental units based on MFCMLFM for TRAFFIC, method options RR and DR showed higher performance than CNTR for the same (FM Structure, Dataset). Additionally, method option DR had the highest performance for experimental units with the smallest Compression Ratio. However, for some experimental units, the second smallest Compression Ratio also showed the highest performance.

For the main performances in Table 9, that is, the performances in Table 13 and the performances marked in red in Table 14, the method options RR and DR were similarly high. For the remaining experimental units other than the main performances, the method option RR had a similarly high performance to DR. However, as described in Section 5.3.2, method option DR tended to show relatively high performance for ACCHAR and method option RR for TRAFFIC. In addition, among the feature options of method option DR, DR(R) or DR(D)-Test(R) showed the best performance.

Table 13. Performance of the TT-VFDL-SIM with method option as RR (for classification) under various conditions.

FM Structure	Dataset	SIM Partial Training (RR) *		Merging Task Training		
		Accuracy (%)	F1-Score	Accuracy (%)	F1-Score	Best Learning Rate
MFCMLFM	ACCHAR	85.94	0.86	**98.80**	**0.99**	0.01 0.001 0.0005
	TRAFFIC	73.73	0.72	95.76	0.96	0.01
MLFM	ACCHAR	89.96	0.90	**97.59**	**0.98**	0.01 0.001 0.0005
	TRAFFIC	76.27	0.76	**98.31**	**0.98**	0.01 0.001
SLFM	ACCHAR	89.56	0.89	**98.80**	**0.99**	0.01 0.001 0.0005
	TRAFFIC	77.69	0.78	**97.46**	**0.97**	0.01 0.001

* Accuracy (%) and F1-Score values of the SIM Partial Training are the averaged values of all guest parties which are participating in the TT-VFDL-SIM (for classification).

Table 14. Performance of the TT-VFDL-SIM with method option as DR (for classification) under various conditions.

FM Structure	Dataset	Compression Ratio	SIM Partial Training (DR) *		Merging Task Training			Best Feature Option
			Accuracy (%)	F1-Score	Accuracy (%)	F1-Score	Best Learning Rate	
MFCMLFM	ACCHAR	**1.20**	**90.77**	**0.91**	**98.80**	**0.99**	**0.001**	**DR(R)**
		2.00	**89.56**	**0.90**	**98.80**	**0.99**	**0.001**	**DR(R)**
		3.00	85.95	0.86	**96.39**	**0.96**	0.01 0.001	DR(R) DR(D)
	TRAFFIC	1.20	85.31	0.85	94.92	0.94	0.01	DR(R)
		2.00	87.57	0.88	94.07	0.95	0.01	DR(R) DR(D)
		3.00	83.05	0.83	94.07	0.95	0.001	DR(R)
MLFM	ACCHAR	**1.20**	**90.36**	**0.90**	**98.80**	**0.99**	**0.01** **0.0005**	**DR(R)**
		2.00	**89.16**	**0.89**	**98.80**	**0.99**	**0.0005**	**DR(R)**
		3.00	88.35	0.88	**97.59**	**0.98**	0.01	DR(R) DR(D)
	TRAFFIC	**1.20**	**79.94**	**0.80**	**95.76**	**0.96**	**0.001**	**DR(D)** **DR(D)-Test(R)**
		2.00	**86.16**	**0.86**	**95.76**	**0.96**	**0.01** **0.001** **0.0005**	**DR(R)**
		3.00	85.59	0.86	93.22	0.92	0.0005	DR(R)
SLFM	ACCHAR	**1.20**	**88.76**	**0.89**	**100.00**	**1.00**	**0.01**	**DR(R)**
		2.00	90.76	0.91	**98.80**	**0.99**	0.01	DR(R)
		3.00	86.35	0.86	**97.59**	**0.98**	0.0005	DR(R)
	TRAFFIC	**1.20**	**86.73**	**0.87**	**96.61**	**0.96**	**0.01**	**DR(R)**
		2.00	88.42	0.88	**95.76**	**0.96**	0.01 0.001	DR(R)
		3.00	86.72	0.87	**94.92**	**0.94**	0.01	DR(R)

* Accuracy (%) and F1-Score values of the SIM Partial Training are the averaged values of all Guest Parties' which participating in the TT-VFDL-SIM (for classification).

In merging task training, setting the learning rate of merging task training to 0.001, which is the learning rate in partial task training, or 0.01, which is greater than the learning rate in partial task training, generally helped improve performance. Meanwhile, for all experimental units, the performance of merging task training improved compared to the average performance of partial task training. Additionally, the correlation coefficients between the average performance of partial task training and the performance of merging task training for all experimental units were 0.62 and 0.82 for method options RR and DR, respectively.

5.4.2. Discussion

Through Section 5.3, it was validated that introducing similar feature spaces in the task-driven transfer approach can contribute to improving the performance of TT-VFDL-SIM. Through the experimental results in Section 5.4.1, the baseline for the similarity of similar feature spaces, which can contribute to improving the performance of TT-VFDL-SIM, can be validated. For both the regression target task and the classification target task, performance was highest when a similar feature space generated from decoded data with the highest reconstruction performance was introduced. In other words, the performance of TT-VFDL-SIM was highest when introducing similar feature spaces generated from decoded data

with the reconstruction performance of Table 7 with MSE of up to 0.00007 or less and R^2 of at least 0.99.

The SIM Partial Training mechanism introduces similar feature spaces to partial task training based on various method options and feature options. In the regression target task, feature option DD(R) or DD(D)-Test(R) of method option DD had the best performance. For method option DR of the classification target task, feature option DR(R) and DR(D)-Test(R) had the best performance. As a result, it can be said that extracting task-driven features based on the raw data leveraging a trained partial task model by introducing a similar feature space generated from decoded data is effective in improving the performance of TT-VFDL-SIM. The introduction of these similar feature spaces in SIM Partial Training can be seen as a kind of transfer learning mechanism.

Meanwhile, the learning rate that can achieve the highest performance in merging task training basically differed depending on the type of FM Structure and dataset. However, in the regression target task, setting the learning rate of merging task training to be larger or smaller than the learning rate of partial task training tended to help improve performance. Meanwhile, in classification target tasks, setting the learning rate of merging task training to be equal to or greater than the learning rate of partial task training tended to help improve performance.

The performance of merging task training and the average performance of partial task training showed a significant linear correlation for all experimental units. In other words, in TT-VFDL-SIM, generally, the higher the average performance of partial task training, the higher the performance of merging task training. Meanwhile, in the regression target task, the average performance of partial task training using method option DD showed a higher linear correlation with the performance of merging task training than the method option RD. Additionally, in the classification target task, the average performance of partial task training using method option DR showed a higher linear correlation with the performance of merging task training than method option RR. In other words, task-driven features trained from similar feature spaces generated based on decoded data, rather than raw data, are not overfitted to univariate partial tasks and can be considered more suitable for merging tasks.

The task-driven transfer approach improves the performance of TT-VFDL-SIM by leveraging transfer learning mechanisms in various ways. First, the target task is trained based on a transfer learning mechanism that fine-tunes the merging task based on task-driven features pre-trained from partial tasks. TT-VFDL-SIM is communication-efficient due to this transfer learning mechanism. Also, TT-VFDL-SIM can achieve excellent performance, unlike existing VFDL methods of the data-driven transfer approach, by leveraging task-driven features rather than data-driven features.

Meanwhile, VFDL methods of transfer learning-based approaches, including TT-VFDL-SIM, only need to align features to train the merging task. In other words, TT-VFDL-SIM and DT-VFDL-AE do not need to consider data alignment between each guest party during partial task training. On the other hand, existing VFDL methods of the training loop parallelization approach, such as P-VFDL-SL, must align the raw data of each guest party [44–46]. TT-VFDL-SIM leverages a transfer learning mechanism to enable distributed learning with improved performance while being independent of data alignment issues between guest parties.

Finally, feature options for SIM Partial Training's method options DD or DR extracted task-driven features by applying the raw data to a trained partial task model based on similar feature spaces generated from decoded data. Because the raw data were not used to train the partial task model, this is a transfer learning mechanism that applies the knowledge trained on the decoded data to the raw data. In this way, partial task training using transfer learning helped improve the performance of TT-VFDL-SIM compared to partial task training without it. As a result, the task-driven transfer approach achieved improved performance in a communication-efficient manner compared to CNTR and existing VFDL methods by utilizing various advantages of transfer learning.

6. Conclusions

In this paper, we propose the TT-VFDL-SIM and the task-driven approach to design the TT-VFDL-SIM. The proposed TT-VFDL-SIM is a vertical federated deep learning method that can achieve improved performance in a more communication-efficient manner than centralized training or existing VFDL methods for multivariate IoT time-series analysis. The task-driven transfer approach derives univariate partial tasks and a merging task from a multivariate target task and applies a transfer learning mechanism for the target task based on them. Univariate partial tasks and a merging task are reorganized as pre-training and fine-tuning tasks for the target task, respectively. The merging task is fine-tuned based on task-driven features trained through univariate partial tasks.

TT-VFDL-SIM can achieve improved performance over centralized training and existing VFDL methods in a communication-efficient manner by taking advantage, which is suitable for the multivariate IoT time-series domain, of both task-driven features and the transfer learning mechanism. In particular, TT-VFDL-SIM introduces similar feature spaces to partial task training in various ways based on the SIM Partial Training mechanism. Through the introduction of these similar feature spaces, it is possible to obtain basic security for not exposing raw data in a completely distributed architecture and prevent overfitting for univariate partial tasks.

In the experiment, the impact of introducing similar feature spaces and task-driven features on improving the performance of TT-VFDL-SIM was analyzed, and through this, the effectiveness of the proposed task-driven transfer approach was verified. In addition, the detailed operational principles and suitability of each stage of the task-driven transfer approach, such as the SIM Partial Training mechanism and merging task training, were also experimentally verified. In the experiment, the MSE of TT-VFDL-SIM applied to the regression target tasks decreased by up to 0.00181, 0.00188, and 0.01143 compared to CNTR, P-VFDL-SL, and DT-VFDL-AE, respectively. In addition, the accuracy of TT-VFDL-SIM applied to the classification target task increased by up to 4.82%, 4.82%, and 33.73% compared to CNTR, P-VFDL-SL, and DT-VFDL-AE, respectively.

TT-VFDL-SIM has been able to improve the performance of the target task by specifying various methods of introducing similar feature spaces through various method options and feature options of the SIM Partial Training mechanism. However, there is a limitation in that it does not present a single mechanism that can generalize various similar feature spaces' introduction techniques implemented with various method options and feature options. Up to this point, we experimentally analyzed the impact of various method options and feature options in SIM Partial Training on the performance of the target task and concluded that it is advantageous to introduce a similar feature space generated based on decoded data. However, this has not been generalized into the design of fixed similar feature spaces' introduction techniques that can be consistently applied to the various datasets and structures of the target task model. In future works, we plan to study generalized methods for various method options and feature options of the SIM Partial Training mechanism.

Author Contributions: Methodology, software, and writing, S.O.; review and editing, M.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2021R1F1A1062559).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. The program codes to perform the experiments using five IoT time-series datasets are available online at http://dwlab.ewha.ac.kr/mlee/codes/tt-vfdl-sim/source.7z (accessed on 1 March 2024). The programs are written in Python.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Li, Q.; Thapa, C.; Ong, L.; Zheng, Y.; Ma, H.; Camtepe, S.A.; Fu, A.; Gao, Y. Vertical Federated Learning: Taxonomies, Threats, and Prospects. *arXiv* **2023**, arXiv:2302.01550. [CrossRef]
2. Gupta, O.; Raskar, R. Distributed Learning of Deep Neural Network over Multiple Agents. *J. Netw. Comput. Appl.* **2018**, *116*, 1–8. [CrossRef]
3. Wu, Z.; Li, Q.; He, B. Practical Vertical Federated Learning with Unsupervised Representation Learning. *IEEE Trans. Big Data* **2022**; *Early access*. [CrossRef]
4. Wang, Q.; Yang, K. Privacy-Preserving Data Fusion for Traffic State Estimation: A Vertical Federated Learning Approach. *arXiv* **2024**, arXiv:2401.11836. [CrossRef]
5. Wang, Z.; Xiao, J.; Wang, L.; Yao, J. A Novel Federated Learning Approach with Knowledge Transfer for Credit Scoring. *Decis. Support. Syst.* **2024**, *177*, 114084. [CrossRef]
6. Chang, W.; Zhu, T. Gradient-Based Defense Methods for Data Leakage in Vertical Federated Learning. *Comput. Secur.* **2024**, *139*, 103744. [CrossRef]
7. Abedi, A.; Khan, S.S. FedSL: Federated Split Learning on Distributed Sequential Data in Recurrent Neural Networks. *Multimed. Tools Appl.* **2023**, *83*, 28891–28911. [CrossRef]
8. Li, Y.; Sha, T.; Baker, T.; Yu, X.; Shi, Z.; Hu, S. Adaptive Vertical Federated Learning via Feature Map Transferring in Mobile Edge Computing. *Computing* **2024**, *106*, 1081–1097. [CrossRef]
9. Cha, D.; Sung, M.; Park, Y.-R. Implementing Vertical Federated Learning Using Autoencoders: Practical Application, Generalizability, and Utility Study. *JMIR Med. Inform.* **2021**, *9*, e26598. [CrossRef]
10. Khan, A.; Ten Thij, M.; Wilbik, A. Communication-Efficient Vertical Federated Learning. *Algorithms* **2022**, *15*, 273. [CrossRef]
11. Vepakomma, P.; Gupta, O.; Swedish, T.; Raskar, R. Split Learning for Health: Distributed Deep Learning without Sharing Raw Patient Data. *arXiv* **2018**, arXiv:1812.00564. [CrossRef]
12. Duan, Q.; Hu, S.; Deng, R.; Lu, Z. Combined Federated and Split Learning in Edge Computing for Ubiquitous Intelligence in Internet of Things: State-of-the-Art and Future Directions. *Sensors* **2022**, *22*, 5983. [CrossRef]
13. Ji, J.; Yan, D.; Mu, Z. Personnel Status Detection Model Suitable for Vertical Federated Learning Structure. In Proceedings of the 2022 The 6th International Conference on Machine Learning and Soft Computing, Haikou, China, 15–17 January 2022; ACM: New York, NY, USA, 2022; pp. 98–104.
14. Hu, Y.; Niu, D.; Yang, J.; Zhou, S. FDML: A Collaborative Machine Learning Framework for Distributed Features. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; ACM: New York, NY, USA, 2019; pp. 2232–2240.
15. Dai, M.; Xu, A.; Huang, Q.; Zhang, Z.; Lin, X. Vertical Federated DNN Training. *Phys. Commun.* **2021**, *49*, 101465. [CrossRef]
16. Han, D.-J.; Bhatti, H.I.; Lee, J.; Moon, J. Accelerating Federated Learning with Split Learning on Locally Generated Losses. In Proceedings of the ICML 2021 Workshop on Federated Learning for User Privacy and Data Confidentiality, Virtual Only, 18–24 July 2021; ICML Board: San Diego, CA, USA, 2021.
17. Bao, J.; Liu, P.; Ukkusuri, S.V. A Spatiotemporal Deep Learning Approach for Citywide Short-Term Crash Risk Prediction with Multi-Source Data. *Accid. Anal. Prev.* **2019**, *122*, 239–254. [CrossRef]
18. Lee, S.; Shin, J. Hybrid Model of Convolutional LSTM and CNN to Predict Particulate Matter. *IJIEE* **2019**, *9*, 34–38. [CrossRef]
19. Peng, H.; Li, H.; Zhang, Y.; Wang, S.; Gu, K.; Ren, M. Multi-Sensor Vibration Signal Based Three-Stage Fault Prediction for Rotating Mechanical Equipment. *Entropy* **2022**, *24*, 164. [CrossRef] [PubMed]
20. Cui, R.; Zhu, A.; Zhang, S.; Hua, G. Multi-Source Learning for Skeleton -Based Action Recognition Using Deep LSTM Networks. In Proceedings of the 2018 24th International Conference on Pattern Recognition (ICPR), Beijing, China, 20–24 August 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 547–552.
21. Zheng, Y.; Liu, Q.; Chen, E.; Ge, Y.; Zhao, J.L. Time Series Classification Using Multi-Channels Deep Convolutional Neural Networks. In *Web-Age Information Management*; Li, F., Li, G., Hwang, S., Yao, B., Zhang, Z., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2014; Volume 8485, pp. 298–310. ISBN 978-3-319-08009-3.
22. Mekruksavanich, S.; Jitpattanakul, A. LSTM Networks Using Smartphone Data for Sensor-Based Human Activity Recognition in Smart Homes. *Sensors* **2021**, *21*, 1636. [CrossRef]
23. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A. y Communication-Efficient Learning of Deep Networks from Decentralized Data. *arXiv* **2023**, arXiv:1602.05629. [CrossRef]
24. Dean, J.; Corrado, G.; Monga, R.; Chen, K.; Devin, M.; Mao, M.; Ranzato, M.; Senior, A.; Tucker, P.; Yang, K.; et al. Large Scale Distributed Deep Networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–8 December 2012; Curran Associates, Inc.: New York, NY, USA, 2012; Volume 25.
25. Chilimbi, T.; Suzue, Y.; Apacible, J.; Kalyanaraman, K. Project Adam: Building an Efficient and Scalable Deep Learning Training System. In Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation, Broomfield, CO, USA, 6–8 October 2014; USENIX Association: Berkeley, CA, USA, 2014; pp. 571–582.

26. Castiglia, T.; Das, A.; Wang, S.; Patterson, S. Compressed-VFL: Communication-Efficient Learning with Vertically Partitioned Data. *arXiv* **2023**, arXiv:2206.08330. [CrossRef]
27. Thapa, C.; Mahawaga Arachchige, P.C.; Camtepe, S.; Sun, L. SplitFed: When Federated Learning Meets Split Learning. *AAAI* **2022**, *36*, 8485–8493. [CrossRef]
28. Su, L.; Lau, V.K.N. Hierarchical Federated Learning for Hybrid Data Partitioning Across Multitype Sensors. *IEEE Internet Things J.* **2021**, *8*, 10922–10939. [CrossRef]
29. Das, A.; Patterson, S. Multi-Tier Federated Learning for Vertically Partitioned Data. In Proceedings of the ICASSP 2021–2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, ON, Canada, 6–11 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 3100–3104.
30. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A Survey of Transfer Learning. *J. Big Data* **2016**, *3*, 9. [CrossRef]
31. Han, T.; Hao, K.; Ding, Y.; Tang, X. A New Multilayer LSTM Method of Reconstruction for Compressed Sensing in Acquiring Human Pressure Data. In Proceedings of the 2017 11th Asian Control Conference (ASCC), Gold Coast, QLD, Australia, 17–20 December 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 2001–2006.
32. Wen, L.; Gao, L.; Li, X. A New Deep Transfer Learning Based on Sparse Auto-Encoder for Fault Diagnosis. *IEEE Trans. Syst. Man. Cybern. Syst.* **2019**, *49*, 136–144. [CrossRef]
33. Luo, S.; Huang, X.; Wang, Y.; Luo, R.; Zhou, Q. Transfer Learning Based on Improved Stacked Autoencoder for Bearing Fault Diagnosis. *Knowl.-Based Syst.* **2022**, *256*, 109846. [CrossRef]
34. Alvi, M.; Cardell-Oliver, R.; French, T. Utilizing Autoencoders to Improve Transfer Learning When Sensor Data Is Sparse. In Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation, Boston, MA, USA, 9–10 November 2022; ACM: New York, NY, USA, 2022; pp. 500–503.
35. Thakur, D.; Biswas, S.; Ho, E.S.L.; Chattopadhyay, S. ConvAE-LSTM: Convolutional Autoencoder Long Short-Term Memory Network for Smartphone-Based Human Activity Recognition. *IEEE Access* **2022**, *10*, 4137–4156. [CrossRef]
36. Home-UCI Machine Learning Repository. Available online: https://archive.ics.uci.edu/ (accessed on 22 May 2024).
37. Time Series Classification Website. Available online: https://www.timeseriesclassification.com/dataset.php (accessed on 22 May 2024).
38. Vito, Saverio. Air Quality. UCI Machine Learning Repository. 2016. Available online: https://doi.org/10.24432/C59K5F (accessed on 19 November 2023).
39. Candanedo, Luis. Appliances Energy Prediction. UCI Machine Learning Repository. 2017. Available online: https://doi.org/10.24432/C5VC8G (accessed on 19 November 2023).
40. Burgus, Javier. Gas Sensor Array Temperature Modulation. UCI Machine Learning Repository. 2019. Available online: https://doi.org/10.24432/C5S302 (accessed on 19 November 2023).
41. Time Series Classification Website—Epilepsy. Available online: https://www.timeseriesclassification.com/description.php?Dataset=Epilepsy (accessed on 19 November 2023).
42. Time Series Classification Website—PEMS-SF. Available online: https://www.timeseriesclassification.com/description.php?Dataset=PEMS-SF (accessed on 19 November 2023).
43. API Documentation | TensorFlow v2.10.1. Available online: https://www.tensorflow.org/versions/r2.10/api_docs (accessed on 19 November 2023).
44. Zhang, J.; Jiang, Y. A Data Augmentation Method for Vertical Federated Learning. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 6596925. [CrossRef]
45. Wu, Z.; Li, Q.; He, B. A Coupled Design of Exploiting Record Similarity for Practical Vertical Federated Learning. *arXiv* **2023**, arXiv:2106.06312.
46. Yang, Y.; Ye, X.; Sakurai, T. Multi-View Federated Learning with Data Collaboration. In Proceedings of the 2022 14th Inter-national Conference on Machine Learning and Computing (ICMLC), Guangzhou, China, 18–21 February 2022; ACM: New York, NY, USA, 2022; pp. 178–183.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.