MVFL: Multivariate Vertical Federated Learning for Time-Series Forecasting

1st Xicun Yang
SJTU Paris Elite Institute of Technology
Shanghai Jiao Tong University
Shanghai, China
yangxicun@sjtu.edu.cn

2nd JunePyo Jung
LTCI, Telecom Paris
Institut Polytechnique de Paris
Palaiseau, France
june.jung@telecom-paris.fr

3rd Jialiang Lu
SJTU Paris Elite Institute of Technology
Shanghai Jiao Tong University
Shanghai, China
jialiang.lu@sjtu.edu.cn

4th Keun-Woo Lim
LTCI, Telecom Paris
Institut Polytechnique de Paris
Palaiseau, France
keunwoo.lim@telecom-paris.fr

5th Leonardo Linguaglossa
LTCI, Telecom Paris
Institut Polytechnique de Paris
Palaiseau, France
linguaglossa@telecom-paris.fr

***Abstract*—Extending multivariate time series forecasting to resource-constrained edge devices is essential for enabling intelligent and sustainable IoT services. A common scenario involves vertically partitioned data across devices, where each device must forecast its own variables while benefiting from others' information. This paper studies a resource-efficient solution for this scenario based on vertical federated learning (VFL). Prior VFL frameworks are designed for situations where only one party holds the labels and would struggle to meet the demand of the targeted scenario, as storage resources usage would increase dramatically with the number of devices. Going beyond VFL, we design multivariate vertical federated learning (MVFL) as a novel federated learning framework, where we separate communication features and local features in an embedded feature space. This design enables MVFL to utilize storage and communication resources more efficiently by eliminating redundant models. On four real-world benchmarks, MVFL outperforms the VFL approach in both efficiency and accuracy, demonstrating its practical value for distributed IoT systems.**

***Index Terms*—vertical federated learning, multivariate time series forecasting, resource-limited devices**

I. Introduction

Sustainable and intelligent service management in IoT systems increasingly relies on efficient edge intelligence. Among these, time series forecasting on end devices plays a central role in enabling predictive services such as energy scheduling, traffic optimization, weather alerts, and disease outbreak warnings. As emerging technologies such as digital twins [1], multimodal modeling [2], and ubiquitous IoT connectivity [3] continue to evolve, the demand for real-time, distributed, and privacy-preserving forecasting grows rapidly.

In these scenarios, accurate predictions require the aggregation of diverse and often non-overlapping signals held by distributed, heterogeneous devices [4]. At the same time, protecting local data privacy remains a foundational constraint [5]. Federated learning (FL), particularly vertical federated learning (VFL), has emerged as a promising solution for enabling collaborative intelligence across devices with vertically partitioned features [6].

However, practical deployments expose a critical sustainability gap: VFL imposes significant computational, storage, and communication burdens on devices that are inherently resource-constrained [7]. For IoT-based service infrastructures that prioritize long-term deployment, energy efficiency, and minimal maintenance, such overhead is unsustainable—especially when scaling up. Furthermore, conventional VFL often fails to support autonomous local service provision, where each device must forecast based on its own data stream, while still benefiting from relevant knowledge across the network.

To address this challenge from a sustainability-driven service management perspective, we propose a simple but powerful idea: enable each device to identify and share only the useful information it holds—namely, the components of its knowledge that are beneficial to others. For example, in an urban climate network, a rise in local temperature may simultaneously inform humidity or wind patterns in the same area. Instead of maintaining one model per communication partner (as in classical VFL), we hypothesize that a unified local model with structured internal representation can achieve the same collaborative effect at a fraction of the cost.

Based on this insight, we introduce Multivariate Vertical Federated Learning (MVFL), a new framework tailored for sustainable, distributed time series forecasting. MVFL decomposes model representation into local features (for on-device prediction) and communication features (for shared learning across devices). Each device maintains only one compact model, significantly reducing storage and training overhead. Unlike prior VFL methods that rely on predefined communication roles (e.g., active/passive parties), MVFL allows for symmetrical, privacy-preserving exchange of internal features, learned dynamically throughout training.

Extensive experiments on four real-world datasets show that

978-3-903176-75-1 ©2025 IFIP

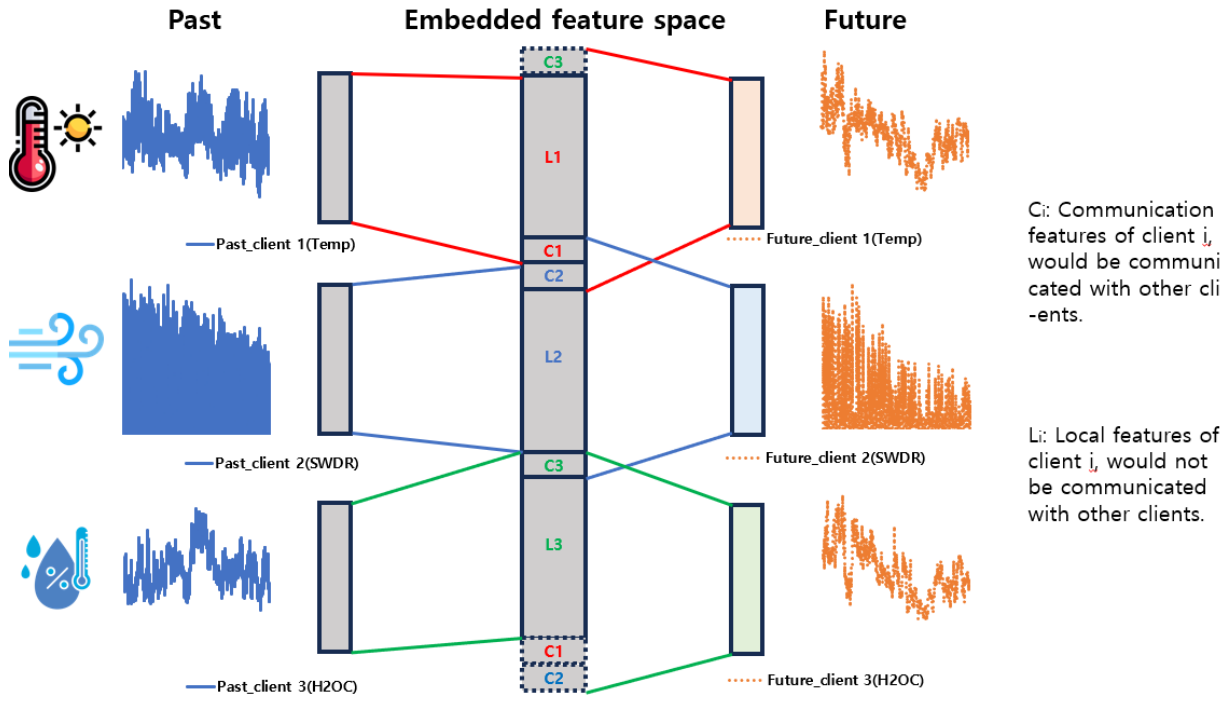

Fig. 1. Illustration of MVFL with a simple example of three devices separately holding data of temperature, wind, and h2o density. They exchange communication features in order to make predictions of own data.

MVFL delivers strong predictive performance while achieving major improvements in resource efficiency: compared to conventional VFL, MVFL could drastically decrease the storage and communication usages while maintaining the accuracy. These results demonstrate MVFL's potential as a sustainable, scalable, and service-friendly learning framework for next-generation IoT forecasting systems.

- We identify the challenges that classical FL approaches face in a new research scenario of multivariate time series forecasting, where the variables of the dataset are vertically distributed across different devices, and each device must predict its own future data.
- We propose MVFL with a compact structure and the separation of *communication features* and *local features*. We adopt an approach *a posteriori* rather than *a priori*, meaning the features evolve throughout the training process rather than being predefined.
- MVFL achieves up to 87.9% of VFL's loss while using only 83% of its storage and 57% of its communication. In more constrained settings, MVFL maintains accuracy with just 25% of the storage required by VFL.

II. Related Work

A. Federated learning & vertical federated learning

Federated Learning (FL) has emerged as a powerful paradigm to enable decentralized machine learning, where devices collaboratively train a model without sharing raw data, thereby preserving privacy [8]. Early foundational works [9] introduced a framework that reduces communication costs by averaging model updates across devices. Since then, FL has been extensively applied to domains where data privacy is crucial [10][11], including healthcare, financial services, and IoT [12][13]. Notably, a hierarchical organization for split federated learning to improve efficiency is proposed in [14]. However, traditional FL techniques primarily assume horizontally partitioned data, where different devices hold different samples of the same features.

In contrast, VFL is designed for scenarios where different parties hold different features for the same set of data instances [15][16]. VFL has proved to be effective in domains like finance and healthcare, where features are naturally distributed across different organizations [17][18]. Yang et al. [6] presented a comprehensive overview of VFL methodologies and challenges, highlighting issues such as communication efficiency, model design, and privacy concerns when multiple parties collaborate under a vertically partitioned data framework.

Nevertheless, current VFL frameworks are primarily designed for scenarios where only one party holds the labels [19][20], incompetent when dealing with the targeted scenario. From the best of our knowledge, there is one single work [21] that deals with a similar scenario with the targeted scenario, where labels are horizontally partitioned and the parties only hold partial labels. However, in the targeted scenario, both labels and data are vertically distributed. Besides, in the above mentioned work, there is still the distinction of active parties, passive parties and a collaborator, since the framework isn't specially designed for time series forecasting tasks. As a result,

none of the parties are equipped with sufficient labels to perform a complete local training, which is not the case in the targeted scenario, as local future data are natural labels for time series forecasting tasks.

B. Multivariate time series forecasting

Multivariate time series forecasting is becoming increasingly important in IoT contexts. Recent approaches include transformer-based models[22] and other approaches [23][24] that further integrate decomposition methods. However, many of these devices are resource-limited, with constraints on processing power, memory, and energy [7]. Federated Learning (FL) has been applied to IoT systems to mitigate the challenges of decentralized data processing. Specially, Chen et al.[25] introduces a prompt learning mechanism to accommodate the communication and computational constraints of low-resource sensors, Seo et al. [26] combines federated learning with SDN to enhance resource efficiency in network management. However, most research has not adequately addressed how to handle multivariate forecasting with vertically distributed variates: For example, Chen et al. [25] only consider the scenario where the time series data are horizontally distributed (any client would hold a complete set of variates and would not need data from other clients to perform local forecasting) and could not serve as comparable baselines for the targeted scenario.

Efforts have been made to reduce FL's resource footprint [27][28], but these methods do not address the redundancy issues inherent in VFL (see Section 3) for large-scale multivariate time series forecasting with vertically partitioned data, where each device needs to build separate models for any other variate. The lack of research addressing these specific challenges represents a critical gap in the literature.

To the best of our knowledge, our study addresses this gap by proposing a novel framework, Multivariate Vertical Federated Learning (MVFL), which is tailored for the targeted scenario. Unlike previous methods, MVFL improves both storage efficiency and forecasting accuracy by eliminating redundant models and efficiently utilizing device resources.

III. Limitations of VFL

In a typical VFL setting with n devices [29], each device holds its local private data X_i, where $i \in \{1, 2, \ldots, n\}$. Only one device (we denote it as device $k \in \{1, 2, \ldots, n\}$) contains the labels Y_k and is referred to as the active party. The active party is equipped with a model M_k with parameters θ_k. The other devices are called passive parties. For these passive parties, each device i also maintains a local model M_i with parameters θ_i. During the forward propagation, the device i would compute the communicated features $C_i = M_i(X_i)$ and send them to the active party. The active party would aggregate the communicated features received from passive parties with its own local data (a common practice is concatenation) and compute $\hat{Y_k} = M_k(aggregation(X_i))$. Define the loss as $\ell(\hat{Y_k}, Y_k)$, then during the back propagation, the active party would compute $g_k = \frac{\partial \ell}{\partial \theta_k}$ in order to update its own model. It would also compute $g_{i,c} = \frac{\partial \ell}{\partial C_i}$ and send them back separately to the passive parties. The passive parties would then compute $g_i = g_{i,c} \cdot \frac{\partial C_i}{\partial \theta_i}$ in order to update the local models.

However, in our targeted scenario, all n devices need to perform forecasting for the local variates using useful information extracted from other devices. Specifically, for any device i, apart from a model intended to perform local forecasting, which we refer to as the main model, it should also maintain $(n-1)$ models to extract information from its own raw data in order to provide useful information for all other devices, which we refer to as exchange models. Consequently, as the number of devices increases, the total number of models that need to be maintained also grows, leading to significant storage and computational overhead. This reality motivates us to propose a novel federated learning framework tailored for this scenario.

IV. Multivariate vertical federated learning

With the above discussions, we could now renovate the typical VFL framework for the targeted scenario. We name the proposed method the multivariate vertical federated learning framework (MVFL).

Concretely, suppose that there is a multivariate time series dataset $D \in \mathbb{R}^{l \times n}$, where l denotes the size of any variate in this dataset and n denotes the number of variates in this dataset, then the targeted scenario is where the variates are distributed exactly on n devices, and any local dataset could be denoted as $D_i \in \mathbb{R}^l, i \in \{1, 2, \ldots, n\}$. The objective of a device is to predict the most probable length-O series in the future given the past length-I series of its own variate combined with information sent from other devices. We note the input $X_{iI} \subseteq D_i$ and the output $X_{iO} \subseteq D_i$.

A simple illustrative example of MVFL for three devices is shown in Figure 1.

A. Forward propagation

In MVFL, a device only needs to maintain one model: its own forecasting model. What are sent to other devices are simply the communication features on one of the hidden layers of its own model.

Concretely, denote the model to maintain for device i as M_i. M_i could be thought of as consisted of two parts: $M_{i,1}$ and $M_{i,2}$. We note C_i and L_i separately for communication features and local features.The details of the forward propagation could be described as below:

$$C_i, L_i = M_{i,1}(X_{iI}) \tag{1}$$

$$C_{i,other} = \text{concatenation}(C_j,\ j \text{ from 1 to } n,\ j \neq i) \tag{2}$$

$$\hat{X}_{iO} = M_{i,2}(C_i, L_i, C_{i,other}) \tag{3}$$

Specifically, the features $C_{i,other}$ concatenate the communication features from all other devices. This is feasible because during the forward propagation, communication features from all the devices would be sent to a trusted server, and the

server would do the concatenation and send those concatenated features (Concretely, $C_{i,other}$ for device i) separately to the devices.

B. Back propagation

The back propagation of MVFL is a modified version of the classical VFL back propagation procedure. Concretely, for a client i, we denote $\theta_{i,1}$ as the model parameters for $M_{i,1}$ and $\theta_{i,2}$ the model parameters for $M_{i,2}$. We note also the loss of the overall model as $\ell_i(\hat{X}_{iO}, X_{iO})$, then during the back propagation, the following gradients are calculated by client i:

$$g_i^1 = \frac{\partial \ell_i}{\partial \theta_{i,1}}, g_i^2 = \frac{\partial \ell_i}{\partial \theta_{i,2}}, g_i^o = \frac{\partial \ell_i}{\partial C_{i,other}} \tag{4}$$

While the gradients g_i^1 and g_i^2 remain local, the client would send g_i^o to the server. We notice that g_i^o = concatenation($g_{i,j}$, j from 1 to n, $j \neq i$) with $g_{i,j}$ denoting the gradients from client i for communication features of client j. After having collected the gradients from all clients, the server would calculate:

$$g_i^c = \sum_{j \neq i} g_{j,i} \tag{5}$$

and send this gradient to client i. Consequently, client i could calculate another gradient:

$$g_i^3 = g_i^c \cdot \frac{\partial C_i}{\partial \theta_{i,1}} = \sum_{j \neq i} \frac{\partial \ell_j}{\partial \theta_{i,1}} \tag{6}$$

Finally, the client i could update its gradients for $\theta_{i,1}$ using $(g_i^1 + g_i^3)$ and the gradients for $\theta_{i,2}$ using g_i^2.

C. Separation of communication and local features

A key challenge in implementing MVFL is how to distinguish between communication features and local features. Our solution is more an approach *a posteriori* than *a priori*, which ensures that we can distinguish between different features without any prior knowledge of the dataset itself, thus greatly improving the applicability of the method.

Concretely, at the beginning of the training process, the only distinction between communication features and local features lies in their positions within the embedded feature space of the local model. However, for any round of training, for a device i, the gradients for the communication features C_i could be expressed as

$$\sum_{j \in \{1, 2, \ldots, n\}} \frac{\partial \ell_j}{\partial C_j} \tag{7}$$

while the gradients for the local features L_i could be expressed as $\frac{\partial \ell_i}{\partial L_i}$. Consequently, as the training process progresses, the communication features and local features will gradually diverge and converge into their respective identities as suggested by their names.

D. Communication resources usage

To compare the communication resources usage of VFL and MVFL, we could fix the size of communication features of any client of MVFL as e. Meanwhile, for VFL, we fix the size of the exchanged features of any client to any other client also as e.

In MVFL framework, for every round of training, for a client i, what are communicated with the server include:

- What are sent to the server: the communication features C_i obtained from local model of size e and the gradients g_i^o for the communication features of other clients of size $e \cdot (n-1)$.
- What are received from the server: The communication features from all other clients $C_{i,other}$ of size $e \cdot (n-1)$ and the gradients for own communication features g_i^c of size e.

The overall communication resource usage of MVFL is thus of size $\mathbf{2n \cdot e}$.

In VFL framework, for every round of training, for a client i, what are communicated with the server include:

- What are sent to the server: the communicated features C_i obtained from local model of size $e \cdot (n-1)$ and the gradients g_i^o for the communicated features of other clients of size $e \cdot (n-1)$.
- What are received from the server: The communicated features from all other clients $C_{i,other}$ of size $e \cdot (n-1)$ and the gradients for own communicated features g_i^c of size $e \cdot (n-1)$.

The overall communication resource usage of VFL is thus of size $\mathbf{(4n-4) \cdot e}$.

We remark that under the same e, the communication resources usage of MVFL is far below that of VFL. Concretely, with the increase of the number of devices, the ratio of communication resource usage of MVFL over VFL would tend to 50%.

V. Experiments

A. Datasets

We evaluate the proposed MVFL by comparing it to the VFL approach on four real-world benchmarks, covering the mainstream time series applications of energy, weather and economics.

The datasets that we use are *ETTm*, *ETTh*, *weather* and *exchange*. All are multivariate datasets. *ETTm* and *ETTh* each has 7 variates. The *weather* dataset has 21 variates, the *exchange* dataset has 8 variates. All data are normalized using a standard scaler.

Here is a brief description [23] of the datasets used: (1) *ETT* [22] dataset contains the data collected from electricity transformers, including load and oil temperature between July 2016 and July 2018. For *ETTh*, the data are recorded every 1 hour. For *ETTm*, the data are recorded every 15 minutes. (2) *weather*(https://www.bgc-jena.mpg.de/wetter/) is recorded every 10 minutes for 2023 whole year, which contains 21 meteorological indicators, such as air temperature, humidity,

TABLE I
COMPARISON OF MVFL AND VFL UNDER DIFFERENT INPUT SIZES FOR THE LOSS(MSE) AND THE COMMUNICATION(SIZE OF COMMUNICATED INFORMATION PER ROUND PER CLIENT)

Datasets		ETTm		ETTh		weather		exchange	
Input size		96	24	96	24	96	24	96	24
MVFL1	Loss(MSE)	**1.21**	**1.72**	**2.14**	**2.39**	4.10	**3.84**	**0.23**	**0.20**
	Communication(bytes)	**168**	**168**	**168**	**168**	**504**	**504**	**192**	**192**
VFL	Loss(MSE)	1.31	1.74	2.28	2.51	4.29	3.87	0.36	0.31
	Communication(bytes)	288	288	288	288	960	960	336	336
MVFL2	Loss(MSE)	1.28	1.79	2.33	2.52	**4.02**	3.90	0.29	0.27
	Communication(bytes)	168	168	168	168	504	504	192	192

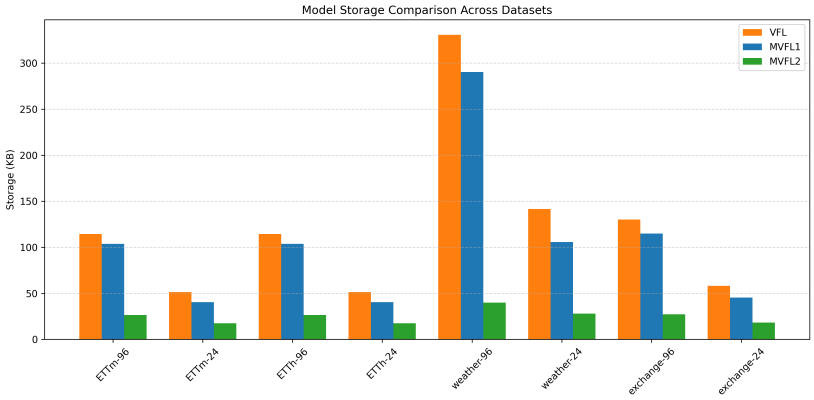

Fig. 2. Comparison of MVFL and VFL under different input sizes for the storage (sum of model sizes) resources usage per client.

etc. (3) *exchange* [30] records the daily exchange rates of eight different countries ranging from 1990 to 2016.

B. Implementation details

Our models are trained with the L2 loss ($l(a,b) = (a-b)^2$), using the ADAM optimizer with an initial learning rate of 0.001. Batch size is set to 32. The training process has its learning rate decayed if there is a stagnation of loss. All experiments are implemented in PyTorch and conducted on a single NVIDIA GPUs. We use mean squared error (MSE) as the core metric to compare performances.

Both the MVFL model and the VFL main model contain 3 hidden layers. For MVFL, the 2nd hidden layer serves as the embedded feature space to obtain the *communication features*. To maximize control over variables and ensure best performances, for VFL, the exchange models all have 1 hidden layer, and the exchanged features obtained from other devices are concatenated with own past data. The storage resources usage are in KB and are obtained by saving all the models locally and adding the sizes of the models saved. The communication resources usage is in bytes and is calculated as shown in Section 4.D.

C. Main results

To compare performances under long and short horizons, we fix the output length and evaluate models with two input lengths: 96, 24. We set the communication features size to 6 and implement the VFL comparison experiments with the same exchanged features sizes.

Note that MVFL could greatly reduce storage and communication resources usage compared to VFL. Consequently, we focus on three indexes for the experiments: loss, communication resources usage and storage resources usage. The first comparison experiments, denoted as MVFL1, control the same storage resources usage, where the size of MVFL models is extended in order to approach the storage resources usage of VFL. The second set of comparison experiments, denoted as case MVFL2, controls the same main model size, where MVFL models are not extended and have the same size of the VFL main models. The results are as shown in Table I.

We remark that MVFL achieves consistent advantageous performance in all benchmarks and for both input size settings. Overall, MVFL yields a loss that is 87.9% of that of VFL with a 83% storage resources usage and a 57% communication resources usage. Or in more extreme situations, MVFL could maintain the loss to 96% of that of VFL with only 25% storage resources usage and 57% communication resources usage.

D. Storage resources usage analysis

The benefits of MVFL over VFL are firstly related to the number of devices involved in the process. In a general case, a client using VFL would require an extra model for each of the other clients, while in the MVFL case only an extra communication features buffer is needed. Therefore, the more devices there are, the more storage resources that MVFL saves.

TABLE II
LOSS OF MVFL AND VFL OVER DIFFERENT EXCHANGE SIZES UNDER THE INPUT-96 SETTING

Datasets	ETTm			ETTh			weather			exchange		
exchange size	3	6	9	3	6	9	3	6	9	3	6	9
MVFL	1.22	1.21	1.24	2.14	2.14	2.15	4.03	4.10	4.02	0.20	0.23	0.25
VFL	1.29	1.31	1.30	2.27	2.28	2.29	4.23	4.29	4.19	0.30	0.36	0.35

TABLE III
LOSS OF CMVFL, SMVFL AND MVFL OVER DIFFERENT EXCHANGE SIZES UNDER THE INPUT-24 SETTING

Datasets	ETTm			ETTh			weather			exchange		
exchange size	3	6	9	3	6	9	3	6	9	3	6	9
CMVFL	1.88	1.87	1.85	2.60	2.63	2.62	4.15	4.13	4.13	0.21	0.23	0.25
SMVFL	1.87	1.82	1.77	2.56	2.51	2.51	3.95	3.93	3.88	0.26	0.25	0.31
MVFL	1.75	1.72	1.70	2.41	2.39	2.39	3.83	3.84	3.85	0.19	0.20	0.24

In addition, the exchange models of VFL take the same input size of the main local model. This means that the exchange model size m increases with the input size. As a result, MVFL is more advantageous when the input size is large. This could be verified by the experiment results: the storage results could be saved more when input size is 96 than when input size is 24 (case MVFL2). In Figure 2, we provide a more illustrative figure to show the advantage of MVFL in terms of storage usages.

E. Ablation studies

In order to study the impact of different communication feature sizes over the performances of MVFL and VFL, we fix the input size and set communication features sizes to 3, 6, 9. Same storage usage is set for MVFL and VFL. The results are shown in the Tables II. We find no clear patterns indicating whether the performance of MVFL relative to VFL improves or deteriorates as the exchange size increases.

We have conducted another set of ablation studies with a more compacted version of MVFL, which we refer to as compact MVFL (CMVFL). In CMVFL, the communication features from different devices are not concatenated, but rather averaged. As a result, the model size of CMVFL would always be constant, no matter the exchange size or the total devices number. Comparison results between MVFL and CMVFL under the same model settings are as shown in Table III. The results show that MVFL is advantageous over CMVFL in all cases, indicating that merging the communication features from all devices does not perform well.

The last set of ablation studies discusses another version of MVFL, which we refer to as simple MVFL (SMVFL). In SMVFL, the communication features are sent to the server, but devices wouldn't send the gradients for other communication features to the server, nor would they receive the feedback gradients of their own communication features. Consequently, the only difference between communication features and local features would be their relative position in the embedded feature space. The comparison results of SMVFL and VFL under the same model settings are as shown in Table III.

The results show that MVFL is advantageous over SMVFL in all cases. This further proves that the MVFL framework is valid and the approach *a posteriori* to separate communication features and local features could indeed improve the performance of the framework.

VI. CONCLUSIONS

In this paper, we proposed Multivariate Vertical Federated Learning (MVFL), a novel framework tailored for resource-efficient multivariate time series forecasting on constrained IoT devices. MVFL introduces a unified model architecture that separates local features for private inference and communication features for efficient inter-device collaboration, significantly reducing per-device storage and communication costs while maintaining or improving the prediction accuracy. Future work may further optimize its performance under diverse conditions.

VII. ACKNOWLEDGDMENTS

This work is supported by NSFC Project with No: 62432009. This work is partially supported by the ANR under the France 2030 program, grant "NF-NAI: ANR-22-PEFT-0003".

REFERENCES

[1] M. Fahim, V. Sharma, T.-V. Cao, B. Canberk, and T. Q. Duong, "Machine learning-based digital twin for predictive modeling in wind turbines," *IEEE Access*, vol. 10, pp. 14 184–14 194, 2022.

[2] N. Hayat, K. J. Geras, and F. E. Shamout, "Medfuse: Multi-modal fusion with clinical time-series data and chest x-ray images," in *Machine Learning for Healthcare Conference*. PMLR, 2022, pp. 479–503.

[3] M. Bauer, L. Sanchez, and J. Song, "Iot-enabled smart cities: Evolution and outlook," *Sensors*, vol. 21, no. 13, p. 4511, 2021.

[4] H. Yang, S. Kumara, S. T. Bukkapatnam, and F. Tsung, "The internet of things for smart manufacturing: A review," *IISE transactions*, vol. 51, no. 11, pp. 1190–1216, 2019.

[5] L. Tawalbeh, F. Muheidat, M. Tawalbeh, and M. Quwaider, "Iot privacy and security: Challenges and solutions," *Applied Sciences*, vol. 10, no. 12, p. 4102, 2020.

[6] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[7] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1–24, 2021.

[8] P. M. Mammen, "Federated learning: Opportunities and challenges," *arXiv preprint arXiv:2101.05428*, 2021.

[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[10] E. Zeydan, L. Blanco, J. Mangues, S. Arslan, and Y. Turk, "Blockchain-based self-sovereign identity for federated learning in vehicular networks," in *2023 19th International Conference on Network and Service Management (CNSM)*. IEEE, 2023, pp. 1–7.

[11] A.-N. Mays, V. Karagiannis, T. De Block, and B. Volckaert, "Federated scheduling of fog-native applications over multi-domain edge-to-cloud ecosystem," in *2023 19th International Conference on Network and Service Management (CNSM)*. IEEE, 2023, pp. 1–7.

[12] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and trends® in machine learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[13] K. Bonawitz, "Towards federated learning at scale: Syste m design," *arXiv preprint arXiv:1902.01046*, 2019.

[14] T. Xia, Y. Deng, S. Yue, J. He, J. Ren, and Y. Zhang, "Hsfl: An efficient split federated learning framework via hierarchical organization," in *2022 18th International Conference on Network and Service Management (CNSM)*. IEEE, 2022, pp. 1–9.

[15] D. Romanini, A. J. Hall, P. Papadopoulos, T. Titcombe, A. Ismail, T. Cebere, R. Sandmann, R. Roehm, and M. A. Hoeh, "Pyvertical: A vertical federated learning framework for multi-headed splitnn," *arXiv preprint arXiv:2104.00489*, 2021.

[16] B. Gu, A. Xu, Z. Huo, C. Deng, and H. Huang, "Privacy-preserving asynchronous vertical federated learning algorithms for multiparty collaborative learning," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 11, pp. 6103–6115, 2021.

[17] Y. Liu, Y. Kang, T. Zou, Y. Pu, Y. He, X. Ye, Y. Ouyang, Y.-Q. Zhang, and Q. Yang, "Vertical federated learning: Concepts, advances, and challenges," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[18] P. Chen, X. Du, Z. Lu, J. Wu, and P. C. Hung, "Evfl: An explainable vertical federated learning for data-oriented artificial intelligence systems," *Journal of Systems Architecture*, vol. 126, p. 102474, 2022.

[19] T. Chen, X. Jin, Y. Sun, and W. Yin, "Vafl: a method of vertical asynchronous federated learning," *arXiv preprint arXiv:2007.06081*, 2020.

[20] B. Gu, Z. Dang, X. Li, and H. Huang, "Federated doubly stochastic kernel learning for vertically partitioned data," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 2483–2493.

[21] W. Xia, Y. Li, L. Zhang, Z. Wu, and X. Yuan, "A vertical federated learning framework for horizontally partitioned labels," *arXiv preprint arXiv:2106.10056*, 2021.

[22] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 12, 2021, pp. 11 106–11 115.

[23] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," *Advances in neural information processing systems*, vol. 34, pp. 22 419–22 430, 2021.

[24] G. Yu, J. Zou, X. Hu, A. I. Aviles-Rivero, J. Qin, and S. Wang, "Revitalizing multivariate time series forecasting: Learnable decomposition with inter-series dependencies and intra-series variations modeling," *arXiv preprint arXiv:2402.12694*, 2024.

[25] S. Chen, G. Long, T. Shen, and J. Jiang, "Prompt federated learning for weather forecasting: Toward foundation models on meteorological data," *arXiv preprint arXiv:2301.09152*, 2023.

[26] E. Seo, D. Niyato, and E. Elmroth, "Auction-based federated learning using software-defined networking for resource efficiency," in *2021 17th International Conference on Network and Service Management (CNSM)*. IEEE, 2021, pp. 42–48.

[27] A. Imteaj and M. H. Amini, "Fedparl: Client activity and resource-oriented lightweight federated learning model for resource-constrained heterogeneous iot environment," *Frontiers in Communications and Networks*, vol. 2, p. 657653, 2021.

[28] S. Shen, R. Li, Z. Zhao, Q. Liu, J. Liang, and H. Zhang, "Efficient deep structure learning for resource-limited iot devices," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–6.

[29] T. Zou, Z. Gu, Y. He, H. Takahashi, Y. Liu, G. Ye, and Y.-Q. Zhang, "Vflair: A research library and benchmark for vertical federated learning," *arXiv preprint arXiv:2310.09827*, 2023.

[30] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st international ACM SIGIR conference on research & development in information retrieval*, 2018, pp. 95–104.