Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105 (Online)
Volume **10**, Number **3**, August **2025**

A Vertical Federated Learning-Based Anomaly Prediction Method for Smart Grid Data in Distributed Systems

Junyi Ma, Xin Yang *, Jie Cai, Zhimin Li, Weiheng Huang, Ximing Xie

Guangdong Zhongshan Power Supply Bureau of China Southern Power Grid, Zhongshan 528400, China
majunyi@zs.gd.csg.cn, 15616036783@163.com, caijie@zs.gd.csg.cn
lz646079171@sina.com, huangweiheng@zs.gd.csg.cn, 360455693@qq.com

*Corresponding author: Xin Yang

Received September 28, 2024, revised January 15, 2025, accepted June 1, 2025.

Abstract.

Smart power grids are considered crucial components of the infrastructure on which modern society relies. As smart grid technology integrates numerous distributed systems, power faults within the grid must be managed and analyzed swiftly to protect framework components and maintain normal operations. Effective operation and maintenance of a smart grid necessitate accounting for all distributed systems. It is vital to utilize maintenance data from these systems to identify abnormalities and address deficiencies before a total breakdown occurs. AI techniques provide a way to predict deficiencies promptly. However, several challenges arise when predicting the health condition of a smart grid with distributed management systems. First, it is essential to utilize as much data from the distributed systems as possible to enhance the accuracy of anomaly prediction. Second, the communication burden among these systems must be minimized to ensure efficient operation. Third, safeguarding data privacy and integrity is paramount. Balancing these three critical factors—data utilization, communication efficiency, and data privacy—poses a significant challenge. To address this, we propose a novel anomaly prediction method that effectively leverages maintenance data from distributed systems within a smart grid. This method employs a vertical federated learning model to extract diverse features stored across different systems, achieving high-accuracy predictions while maintaining data privacy. Experimental results indicate that the proposed method offers a promising solution for anomaly prediction in the operation and maintenance of smart grids with distributed management systems.

Keywords: Vertical Federated Learning; Distributed Systems; Smart Grids; Anomaly Prediction

1. **Introduction.** The provision of continuous, high-quality electricity in a safe and efficient manner is increasingly challenged by aging and overburdened conventional distribution networks. Independent System Operators and Regional Transmission Organizations heavily depend on distributed management systems to improve grid reliability and efficiency. Meanwhile, smart grid technology, which can integrate distributed systems, has emerged as a promising solution for enhancing existing electrical infrastructures [1]. However, integrating cyber and physically distributed systems into smart grid systems introduces several challenges, including concerns over data privacy, security, and reliability in communication and performance. Consequently, the operation and maintenance of smart grids become both crucial and complex [2]. It is essential to identify and address

deficiencies in the smart grid before they lead to a complete breakdown. One of the important strategies adopted by smart grids is predictive maintenance which is a proactive maintenance strategy that uses data analysis tools and techniques to detect anomalies and predict failures before they occur [3]. In this manner, the smart grid ought to utilize analytic data to identify abnormalities in distributed systems.

In this paper, we deal with anomaly prediction from smart grid maintenance data in distributed systems. Anomaly detection encompasses a wide range of applications, from system intrusion detection to fraud detection [4]. To enhance the applicability of our solution for the operation and maintenance of smart grids, we concentrate on the research leveraging maintenance data from distributed systems to predict failures. Conventional methods usually try to find specific patterns in the distribution of the data [5, 6, 7]. As the power sector grows more complex, intelligent tools and mechanisms [8,9,10] are being introduced to manage the system effectively and enable timely predictions. Notably, artificial neural networks (ANNs), reinforcement learning (RL), genetic algorithms (GA), and multi-agent systems are among the most prominent AI techniques employed for these purposes [11].

However, much of the research to date has not adequately addressed several key challenges. Firstly, communication within the smart grid must be considered. Information generated by different devices in the smart grid is often contaminated by measurement noise and quantization noise [12]. Therefore, to achieve high anomaly prediction accuracy, maintenance data from various distributed systems in the smart grid must be collected and integrated as much as possible. However, this integration of communication for analytics increases the risk of assaults and adds to the communication burden on the systems [3]. Despite numerous data fusion techniques developed over the past decades [13], the limited resources within these systems continue to pose significant challenges. Additionally, the heterogeneity of maintenance data across different systems poses a significant challenge for training deep learning models directly on these datasets. Finally, information privacy and data integrity are critical concerns when analyzing maintenance data stored in different distributed systems. To address these challenges, we propose a novel anomaly prediction method that leverages maintenance data from distributed systems within the smart grid. This method is based on federated learning [14], which protects information privacy across different systems and reduces the communication burden. Specifically, our contributions are summarized as follows:

- The proposed anomaly prediction method constructs a novel vertical federated learning model to leverage diverse features within maintenance data across different systems, effectively addressing the challenges posed by data heterogeneity. In this framework, distributed systems within the smart grid independently train on their local maintenance data. The outputs from these local models are then used to train a global model, ensuring comprehensive integration and analysis of the distributed data.
- As only the outputs from the local models are transmitted to the central server, the proposed model effectively reduces the communication burden while simultaneously protecting local data privacy and integrity.
- Compared to models trained directly on local data without aggregation, the proposed model demonstrates superior performance.

The structure of the paper is organized as follows: Section 2 reviews related research on anomaly prediction using smart grid maintenance data in distributed systems. Section 3 explains why we chose the proposed method and gives an analysis to show the rationale behind the approach. Section 4 presents an overview of the proposed model and describes

the algorithm in detail. Section 5 evaluates the performance of the proposed model in comparison to the average performance of local models. Finally, Section 6 provides a summary and concluding remarks on the proposed system.

2. Related work.

One of the key solutions to improve anomaly detection performance in the literature is data fusion technology [15]. Data fusion integrates information from multiple sources to produce more consistent, accurate, and useful outcomes than those provided by individual sources. Among the probabilistic methods developed for data fusion, Bayesian inference is particularly notable [16,17]. This approach requires a relatively small number of sample data to train the system and effectively handles the heterogeneity of information based on the probabilistic occurrence of events in the environment. Evidence theory is another robust method for data fusion, which extracts precise information from multiple sensor nodes [18, 19]. It converts multi-source subjective and conflicting information into a decision-making result by combining mass functions from different sources. To mitigate the communication burden, artificial intelligence-based data aggregation and fusion techniques have gained popularity. These techniques can effectively classify and abstract information, extracting important features and knowledge from the data [20, 21].

In 2017, McMahan et al. introduced a new learning paradigm called federated learning [22]. Federated learning decouples data collection and model training through multi-party computation, avoiding the need to exchange private data. Within federated learning, horizontal federated learning assumes that all parties' datasets are homogeneous, sharing the same features and structures [23]. However, maintenance data in different systems can be highly diverse, necessitating data fusion from multiple heterogeneous datasets. Vertical federated learning is better suited for these scenarios. In vertical federated learning, each data owner has different features and labels, allowing for the integration of diverse data sources [23]. This method aligns well with the nature of maintenance data in distributed systems, facilitating the effective use of diverse data without compromising privacy.

3. Methodology.

3.1. Dataset and Solution Availability Analysis.

Since the core concept of our solution is to utilize a federated learning framework to extract knowledge from diverse features in maintenance datasets distributed across different systems within a smart grid, it is crucial to ensure that a deep learning model trained on a greater number of relevant features outperforms those trained on fewer features. This hypothesis can be easily validated.

First, we generate a synthetic dataset, as detailed in [24]. Table 1 outlines the dataset, which includes 10 features and a target column indicating anomalies. Each feature is assigned a random effect score relative to the target value, with the synthetic algorithm for feature generation described in Algorithm 1. Feature values follow a normal distribution, and each feature randomly influences the risk score in each row, ultimately determining the target value. The "flag" in Line 12 indicates whether a feature has a positive or negative effect on the risk score. The relationship between all features and the target value is nonlinear. This dataset is then split into several sub-datasets, each stored in a different system within the smart grid, where each sub-dataset contains a distinct set of features. For example, one sub-dataset may consist of columns 1 and 2 along with the target column, as shown in Table 1.

To verify our solution's rationale, we generate several deep learning models with identical architectures and initial parameters, specifically using Long Short-Term Memory (LSTM) [25]. These models are trained on varying numbers of features extracted from

TABLE 1. A Sample from the Dataset

Feature1	**Feature2**	**...**	**Feature10**	**target**
0.427202272	0.569079297	...	0.373574315	0
0.429195462	0.65435159	...	0.331320297	0
0.45473194	0.556842705	...	0.513657834	1

Algorithm 1 Data Synthesization

Require: $num\_samples$, $mean$, std, $count$
$list \leftarrow []$
for $i \leftarrow 1$ to $count$ **do**
  $feature \leftarrow \text{normal}(mean, std, num\_samples)$
  $list$.append($feature$)
end for
$risk \leftarrow \text{generateInt}([0, 1], size = num\_samples)$
for $i \leftarrow 0$ to $num\_samples - 1$ **do**
  $score \leftarrow 0$
  for $j \leftarrow 0$ to $count - 1$ **do**
    $feature \leftarrow list[j][i]$
    $times \leftarrow \text{random\_int}(1, 6)$
    $flag \leftarrow \text{random\_choice}([-1, 1])$
    $score \leftarrow score + \frac{1}{count} \times (feature^{times}) \times flag$
  end for
  $threshold \leftarrow 0.45$
  $risk[i] \leftarrow \text{int}(score > threshold)$
end for

the synthetic dataset. As illustrated in Figure 1, the orange curve represents the performance of Model I which is trained on columns 1, 2, and the target column. The green curve represents the performance of Model II which is trained on columns 1, 2, 3, 4, and the target column. Subsequently, other models are trained on different subsets of features from the whole dataset. The results indicate that models trained on a greater number of features achieve higher accuracy compared to those trained on fewer features. This supports our solution's key idea of constructing a federated learning framework to maximize knowledge extraction from the maintenance data of distributed systems.

3.2. **LSTM And Federated Learning. LSTM.** Long Short-Term Memory (LSTM) networks [25] are a specialized type of recurrent neural network (RNN) designed to capture long-term dependencies in sequential data. In our framework, LSTM units are employed both for learning from local datasets and for constructing the global model. Each LSTM unit comprises a cell state (C_t) and three gates: the input gate (i_t), the forget gate (f_t), and the output gate (o_t). These gates control the flow of information into and out of the cell state. The governing equations for the LSTM unit are as follows:

1. Forget Gate:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

2. Input Gate:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

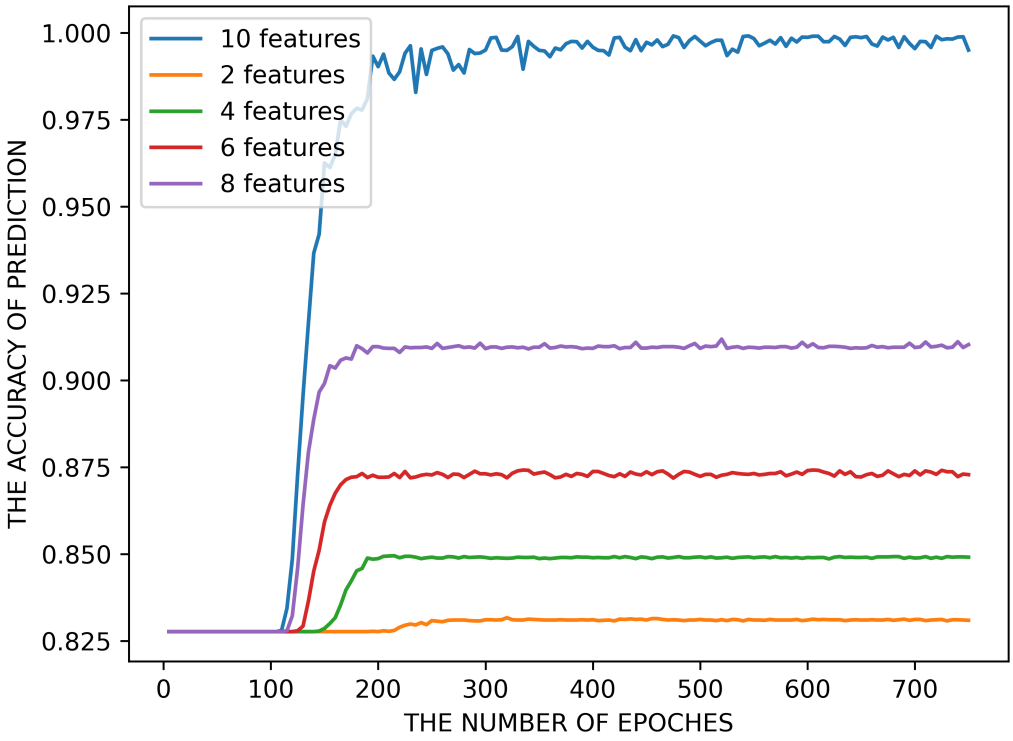

FIGURE 1 Learning Results on Different Number of Features

3. Cell State Update:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

4. Output Gate:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \cdot \tanh(C_t)$$

In these equations, σ denotes the sigmoid function, and tanh represents the hyperbolic tangent function. The weight matrices are represented by W, while the bias vectors are denoted by b. The input at time step t is x_t, and the hidden state at time step t is h_t.

Federated Learning. Federated Learning [23] is a decentralized machine learning approach in which multiple clients collaborate to train a model while keeping their data localized. This approach can be divided into two main categories: horizontal and vertical federated learning. Horizontal federated learning assumes that all parties' datasets are homogeneous, sharing the same features and structures. In contrast, vertical federated learning assumes that each data owner has different features and labels. Given the heterogeneity of maintenance data in our context, we employ a vertical federated learning framework. The general process of federated learning involves the following steps. Generally, the federated learning process involves the following steps:

- 1. Initialization: A global model is initialized and distributed to all participating clients.
- 2. Local Training: Each client trains the model on its local data, resulting in local model updates. The local training process can be represented as:

$$\theta_i^{t+1} = \theta_i^t - \eta \nabla L(\theta_i^t; D_i)$$

where θ_i^t represents the model parameters for client i at iteration t, η is the learning rate, ∇L is the gradient of the loss function, and D_i is the local dataset of client i.

- 3. Aggregation: The local updates are sent to a central server, which aggregates them to update the global model. A common aggregation method is Federated Averaging [22] :

$$\theta^{t+1} = \sum_{i=1}^{N} \frac{n_i}{n} \theta_i^{t+1}$$

where N is the number of clients, n_i is the number of data points on client i, and n is the total number of data points across all clients.

- 4. Iteration: Steps 2 and 3 are repeated for several rounds until the global model converges.

In our solution, we build upon the core principles of federated learning, enhancing the general learning process to better suit our specific scenario. The detailed design and implementation of our approach are discussed in the following section.

4. The proposed framework.

4.1. **Overview.** As illustrated in Figure 2, the proposed framework comprises two key components: local model training utilizing the maintenance data stored in distributed systems within the smart grid and global model training. The central server begins by initializing an LSTM model and distributing its parameters to each system within the smart grid. Subsequently, each client/system trains the LSTM model on its respective local dataset. The outputs from the local models are then aggregated, and a global LSTM model is trained using this aggregated data. The process of local and global training is iterated until the global model converges.

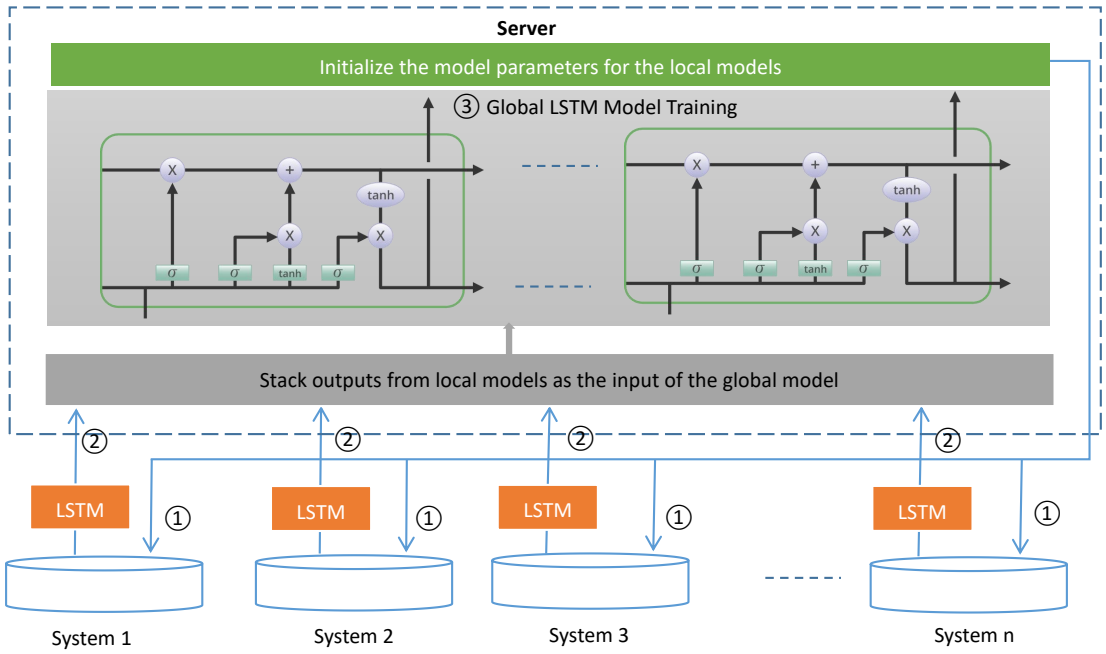

FIGURE 2 The proposed framework

4.2. **Local Model Training.** As outlined in Algorithm 2, each client initially receives identical LSTM model parameters from the central server within the smart grid. Once these parameters are obtained, local systems start training the LSTM model using their respective maintenance data. The training process involves utilizing sequences of features to predict the target column. Upon completion of the training, each local model generates outputs from the trained model along with the corresponding targets. These outputs and targets are then transmitted to the central server, while the local model parameters are retained within the local system.

Algorithm 2 Local Model Training with LSTM

```
client_model ← LSTMModel()
if isFirstGlobalTrainingRound then
    local_model ← getFromServer()
else
    local_model ← getFromLocalStorage()
end if
copy_weights(local_model, client_model)
criterion ← nn.CrossEntropyLoss()
optimizer ← optim.SGD()
for epoch ← 0 to num_epochs − 1 do
    train(client_model, local_dataset, criterion, optimizer)
end for
outputs ← []
flag ← 0
for each (X, y) in train_loader do
    if flag == 0 then
        outputs ← client_model(X)
        global_targets ← y
        flag ← 1
    else
        outputs ← np.vstack([outputs, client_model(X)])
        global_targets ← np.hstack([global_targets, y])
    end if
end for
StoreLocalModel()
sendToServer([outputs, global_targets])
```

4.3. **Global Model Training.** As outlined in Algorithm 3, the central server first initializes an LSTM model and distributes its parameters to the distributed systems. The server then waits to receive the outputs from the local models in each system. Once these outputs are obtained, the server combines them into a new dataset, including the corresponding targets, and splits this dataset into training and test sets. The global LSTM model is subsequently trained using the aggregated data. Finally, the server evaluates the performance of the trained global model. If the performance is below the established threshold, the process of local and global training iterates until satisfactory performance is achieved or the maximum training round is reached.

5. Experimental Results and Analysis.

Algorithm 3 Global Model Training with LSTM

```
initial_local_model ← LSTMModel()
global_model ← LSTMModel()
sendToLocalSystems(initial_local_model.params)
globalTrainingRound ← 1000 {maximum}
for i ← 0 to global_rounds do
    output_list, targets ← receiveFromClients()
    inputs ← np.hstack([output for output in output_list])
    global_targets ← targets.reshape(−1, 1)
    data ← np.hstack([inputs, targets])
    train_data, test_data ← split(data, test_size = 0.2)
    criterion ← nn.CrossEntropyLoss()
    optimizer ← optim.SGD()
    for epoch ← 0 to num_epochs − 1 do
        train(global_model, train_data, criterion, optimizer)
    end for
    res ← evaluate(global_model)
    if res > threshold then
        break
    end if
end for
```

5.1. **Experimental Setup.** The experimental setup for this work was carefully designed to ensure the optimal performance of the proposed framework. The hardware configuration included a computing server with the following high-performance specifications:

- Processor: Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz, 32 cores
- Memory: 125 GB in total
- Graphics Processing Unit (GPU): Two NVIDIA Tesla P40

To maintain consistency across different runs, we used an identical random seed for all experiments. The test model in the proposed federated learning framework was an LSTM, configured with a learning rate of 0.001 and 64 hidden units. For each iteration, the training epoch is 20. The software environment was set up as follows:

- Operating System: CentOS Linux 7
- Deep Learning Framework: PyTorch 1.13.1+cu117
- Python Version: 3.7.13, with necessary libraries such as NumPy and Pandas for data preprocessing .

5.2. **Evaluation Metrics.** The issue addressed by the proposed model is a binary classification task. Evaluating model performance in binary classification is crucial for understanding its effectiveness. Four key metrics are commonly employed: **accuracy**, **precision** and **F1-score**, each offering unique insights into the model's effectiveness.

Accuracy. This is the proportion of correctly predicted instances among the total instances, providing an overall sense of the model's correctness. The calculation formula is as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where TP (True Positives) indicates the positive cases that have been correctly predicted, TN (True Negatives) indicates negative cases that have been correctly predicted,

FP (False Positives) indicates positive cases that have been incorrectly predicted and FN (False Negatives) indicates negative cases that have been incorrectly predicted.

Precision. This metric quantifies the accuracy of positive predictions, representing the ratio of true positive instances to the total predicted positive instances. The calculation formula is as follows:

$$\text{Precision} = \frac{TP}{TP + FP}$$

High precision indicates a low false positive rate, signifying the model's reliability when predicting positive instances.

F1-score. This is the harmonic mean of precision and recall, providing a balance between the two metrics. It is particularly useful when class distribution is imbalanced. The calculation formula is as follows:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1-score ranges from 0 to 1, with 1 being the optimal score, indicating perfect precision and recall. These metrics collectively offer a comprehensive understanding of a model's performance, facilitating more informed decisions about model improvements and deployment strategies.

5.3. **Results Analysis.** To assess the effectiveness of the proposed method, referred to as VFL, we carried out a series of experiments, comparing its performance against the average results of local models trained solely on individual datasets without any aggregation. The evaluation focused on key metrics, including prediction accuracy, precision, and F1-score, as discussed in Section 5.2. To mimic real-world scenarios, the entire dataset introduced in Section 3 was partitioned into five sub-datasets, each representing a local maintenance dataset within a smart grid system. We employed three distinct data partitioning strategies to simulate different scenarios.

In the first scenario, the dataset was divided into five sub-datasets, each containing two feature columns, with no common features shared across all local datasets. In the second scenario, each sub-dataset comprised three feature columns. Since the entire dataset contains ten feature columns, adjacent local datasets share one common feature. Specifically, the dataset was split as follows: with feature columns numbered 1 to 10 and local datasets numbered 1 to 5, the first local dataset included feature columns 1, 2 and 3; the second included columns 3, 4 and 5; the third, columns 5, 6 and 7; the fourth, columns 7, 8 and 9; and the fifth, columns 9, 10 and 1. Similarly, in the third scenario, the dataset was divided into five sub-datasets, each with four feature columns, resulting in adjacent local datasets sharing two common features. For instance, the first local dataset included columns 1, 2, 3 and 4, while the second included columns 3, 4, 5 and 6. It is important to note that the sequence numbers are used solely to illustrate the data partitioning method and are not directly related to the experiments themselves.

The performance curves for the three scenarios are illustrated in Figures 3 through 5. A notable observation across all scenarios is that both the VFL and local models tend to stabilize after a certain number of learning rounds. The figures clearly demonstrate a significant improvement in the performance of the VFL method across all scenarios when compared to the average performance of the local models. As the number of feature columns increases from the first to the third scenario, the average performance of the local models also improves. However, despite this improvement, the performance gap between VFL and the local models remains substantial.

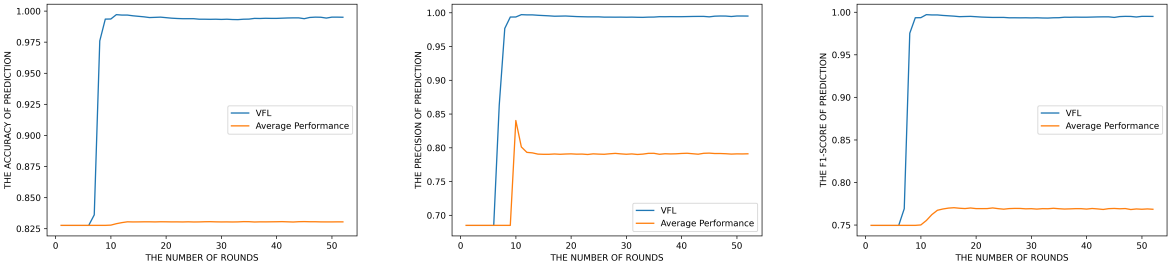

FIGURE 3. Each Local Dataset with 2 Feature Columns

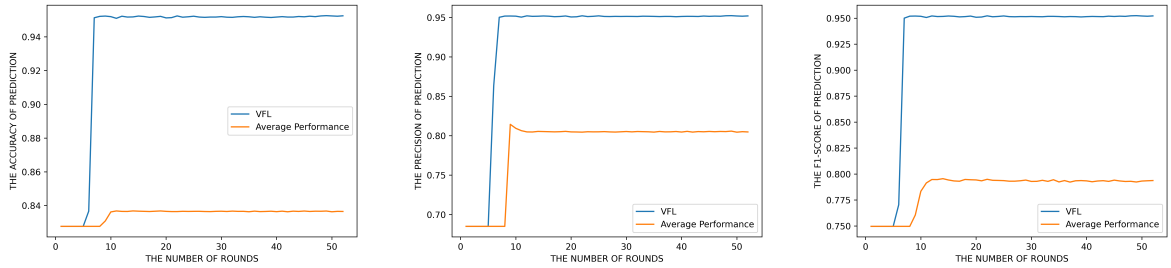

FIGURE 4. Each Local Dataset with 3 Feature Columns

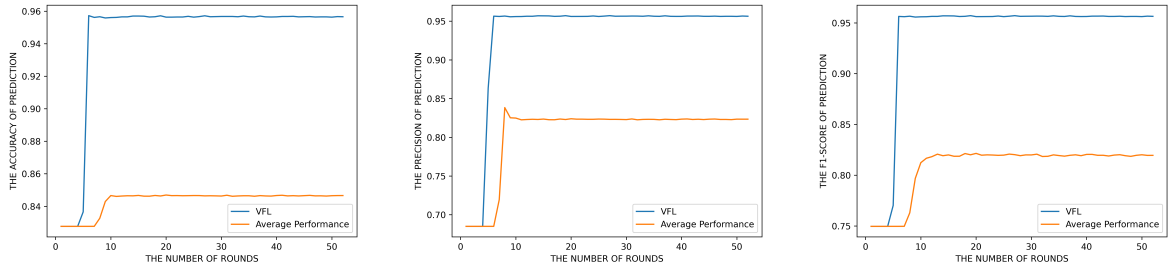

FIGURE 5. Each Local Dataset with 4 Feature Columns

To provide a thorough evaluation of the proposed framework, we also tested the model using various sizes of test data. The results are summarized in Tables 2 through 4. During model training, each dataset was divided into a training set and a test set. The model was initially trained on the training data and subsequently evaluated on the test data. A 20% test size indicates an 80 : 20 split between the training and test datasets, with 30% and 40% test sizes following similar split ratios.

In the tables, the "Model" column differentiates between the proposed VFL model and the AVG, which represents the average performance of the local models. The numbers following VFL and AVG correspond to the different scenarios based on the number of feature columns in each local dataset. For instance, VFL-2 refers to the Vertical Federated Learning-Based Anomaly Prediction model applied in the scenario where each local dataset contains two feature columns. The values presented in Tables 2 through 4 were recorded after the models reached a relatively stable state, which occurred after approximately 50 communication rounds. These results provide an estimation of the models' optimal performance. The data reveal that the average accuracy for VFL across all scenarios is approximately 96.84%, compared to 83.74% for the AVG model. Similarly, the average precision for VFL is around 96.77%, while AVG achieves 80.95%. Furthermore,

TABLE 2. Performance with 20% Test size

Model	Accuracy	Precision	F1
VFL-2	0.9950	0.9950	0.9950
AVG-2	0.8307	0.7928	0.7680
VFL-3	0.9527	0.9520	0.9523
AVG-3	0.8363	0.8045	0.7927
VFL-4	0.9567	0.9564	0.9565
AVG-4	0.8462	0.8227	0.8183

TABLE 3. Performance with 30% Test size

Model	Accuracy	Precision	F1
VFL-2	0.9943	0.9943	0.9943
AVG-2	0.8303	0.7891	0.7691
VFL-3	0.9512	0.9507	0.9509
AVG-3	0.8359	0.8031	0.7931
VFL-4	0.9584	0.9582	0.9583
AVG-4	0.8457	0.8219	0.8190

TABLE 4. Performance with 40% Test size

Model	Accuracy	Precision	F1
VFL-2	0.9938	0.9937	0.9937
AVG-2	0.8301	0.8276	0.7689
VFL-3	0.9509	0.9506	0.9507
AVG-3	0.8357	0.8025	0.7914
VFL-4	0.9585	0.9583	0.9584
AVG-4	0.8455	0.8216	0.8181

the average F1-score for VFL is about 0.9678, significantly higher than the 0.7932 observed for AVG. The results consistently show that VFL outperforms the local models across all scenarios and evaluation metrics, demonstrating the robustness and effectiveness of the proposed approach.

6. **Conclusion.** In this study, we introduce a novel vertical federated learning-based anomaly prediction method for maintenance in distributed systems. This approach maximizes the utilization of maintenance data stored across different systems to achieve high prediction accuracy. Within the proposed federated learning framework, local deep learning models are initially trained within each distributed system in the smart grid, using only their respective local data, which may feature unique characteristics distinct from other systems. The insights gained from these local models are then aggregated and utilized as input for a global model, which is trained on a central server. This methodology enables the global vertical federated learning model to leverage the diverse maintenance data across different systems, resulting in superior prediction performance. Additionally, since only the outputs of each local model, rather than the entire local datasets, are transmitted, the integrity and privacy of the original data are preserved. To assess the effectiveness of the proposed method, we conducted a series of experiments comparing its performance against the average results of deep learning models trained on individual local datasets without aggregation. The experimental results demonstrate the enhanced

prediction accuracy and robustness of the proposed method, confirming its potential as a valuable solution for anomaly prediction in the maintenance of smart grids with distributed management systems.

REFERENCES

[1] O. M. Butt, M. Zulqarnain, and T. M. Butt, "Recent advancement in smart grid technology: Future prospects in the electrical power network," *Ain Shams Engineering Journal*, vol. 12, no. 1, pp. 687–695, 2021.

[2] G. Dileep, "A survey on smart grid technologies and applications," *Renewable energy*, vol. 146, pp. 2589–2625, 2020.

[3] M. A. Mahmoud, N. R. Md Nasir, M. Gurunathan, P. Raj, and S. A. Mostafa, "The current state of the art in research on predictive maintenance in smart grid distribution network: Fault's types, causes, and prediction methods—a systematic review," *Energies*, vol. 14, no. 16, p. 5078, 2021.

[4] S. Chren, B. Rossi, B. Bühnova, and T. Pitner, "Reliability data for smart grids: Where the real data can be found," in *2018 smart city symposium prague (scsp)*. IEEE, 2018, pp. 1–6.

[5] M. He and J. Zhang, "A dependency graph approach for fault detection and localization towards secure smart grid," *IEEE Transactions on Smart Grid*, vol. 2, no. 2, pp. 342–351, 2011.

[6] V. Calderaro, C. N. Hadjicostis, A. Piccolo, and P. Siano, "Failure identification in smart grids based on petri net modeling," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 10, pp. 4613–4623, 2011.

[7] C. Promper, D. Engel, and R. C. Green, "Anomaly detection in smart grids with imbalanced data methods," in *2017 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2017, pp. 1–8.

[8] I. Siniosoglou, P. Radoglou-Grammatikis, G. Efstathopoulos, P. Fouliras, and P. Sarigiannidis, "A unified deep learning anomaly detection and classification approach for smart grid environments," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1137–1151, 2021.

[9] W. Danilczyk, Y. L. Sun, and H. He, "Smart grid anomaly detection using a deep learning digital twin," in *2020 52nd North American Power Symposium (NAPS)*. IEEE, 2021, pp. 1–6.

[10] H. Karimipour, S. Geris, A. Dehghantanha, and H. Leung, "Intelligent anomaly detection for large-scale smart grids," in *2019 IEEE Canadian conference of electrical and computer engineering (CCECE)*. IEEE, 2019, pp. 1–4.

[11] S. S. Ali and B. J. Choi, "State-of-the-art artificial intelligence techniques for distributed smart grids: A review," *Electronics*, vol. 9, no. 6, p. 1030, 2020.

[12] H. Leung, C. Seneviratne, and M. Xu, "A novel statistical model for distributed estimation in wireless sensor networks," *IEEE transactions on signal processing*, vol. 63, no. 12, pp. 3154–3164, 2015.

[13] C. Seneviratne, P. A. D. S. N. Wijesekara, and H. Leung, "Performance analysis of distributed estimation for data fusion using a statistical approach in smart grid noisy wireless sensor networks," *Sensors*, vol. 20, no. 2, p. 567, 2020.

[14] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.

[15] H. B. Mitchell, *Data fusion: concepts and ideas*. Springer Science & Business Media, 2012.

[16] S. Mil and M. Piantanakulchai, "Modified bayesian data fusion model for travel time estimation considering spurious data and traffic conditions," *Applied Soft Computing*, vol. 72, pp. 65–78, 2018.

[17] C. N. Taylor and A. N. Bishop, "Homogeneous functionals and bayesian data fusion with unknown correlation," *Information Fusion*, vol. 45, pp. 179–189, 2019.

[18] N. Nesa and I. Banerjee, "Iot-based sensor data fusion for occupancy sensing using dempster–shafer evidence theory for smart buildings," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1563–1570, 2017.

[19] O. Kanjanatarakul and T. Denœux, "Distributed data fusion in the dempster-shafer framework," in *2017 12th System of Systems Engineering Conference (SoSE)*. IEEE, 2017, pp. 1–6.

[20] J.-P. Gao, C.-B. Xu, L. Zhang, J.-L. Zheng, H. Shu, and X. Yuan, "A method of information fusion based on fuzzy neural network and its application," in *ITM Web of Conferences*, vol. 11. EDP Sciences, 2017, p. 01015.

[21] B. Bigdeli, P. Pahlavani, and H. A. Amirkolaee, "An ensemble deep learning method as data fusion system for remote sensing multisensor classification," *Applied Soft Computing*, vol. 110, p. 107563, 2021.

[22] 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.

[23] X. Zhang, A. Mavromatis, A. Vafeas, R. Nejabati, and D. Simeonidou, "Federated feature selection for horizontal federated learning in iot networks," *IEEE Internet of Things Journal*, vol. 10, no. 11, pp. 10 095–10 112, 2023.

[24] Pumigyit, "A synthetic maintenance dataset in smart grid," 2024, accessed: 2024-08-06. [Online]. Available: https://www.kaggle.com/datasets/pumigyit/a-synthetic-maintenance-dataset-in-smart-grid

[25] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.