Supplementary Materials For

“A Hybrid Federated Learning Framework with Dynamic Task Allocation for Multi-Party Distributed Load Prediction”

Haizhou Liu, Xuan Zhang*, Xinwei Shen*, Hongbin Sun and Mohammad Shahidehpour

A. Probabilistic XGBoost

Probabilistic XGBoost is a probabilistic machine learning architecture, on top of the XGBoost algorithm already introduced in the manuscript. However, instead of training one tree in each epoch, probabilistic XGBoost trains multiple trees in each epoch, one for each distributional parameter.

In this supplementary material, we take the example of the Gaussian distribution:

$$P(\mathbf{x}) \sim \mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x})) \tag{S1}$$

where $P(\mathbf{x})$ refers to the load consumption label of the input sample $\mathbf{x}$. $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ denote the mean and standard deviation of the corresponding Gaussian distribution.

Then, in each epoch, we train *a pair of* gradient-boosted trees, namely, $(\mathcal{T}_1^{\mu}, \mathcal{T}_1^{\sigma})$, $(\mathcal{T}_2^{\mu}$, $\mathcal{T}_2^{\sigma})$, $\cdots$, $(\mathcal{T}_S^{\mu}, \mathcal{T}_S^{\sigma})$, where $\mathcal{T}_t^{\mu}$ and $\mathcal{T}_t^{\sigma}$ denote the trees constructed in the t-th epoch, on the mean μ and standard deviation σ, respectively. The final prediction model is therefore changed from (2) into:

$$\mu(\mathbf{x}) \sim \mathcal{T}^{\mu}(\mathbf{x}) = \mathcal{T}_1^{\mu}(\mathbf{x}) + \mathcal{T}_2^{\mu}(\mathbf{x}) + \cdots + \mathcal{T}_S^{\mu}(\mathbf{x}) \tag{S2}$$

$$\sigma(\mathbf{x}) \sim \mathcal{T}^{\sigma}(\mathbf{x}) = \mathcal{T}_1^{\sigma}(\mathbf{x}) + \mathcal{T}_2^{\sigma}(\mathbf{x}) + \cdots + \mathcal{T}_S^{\sigma}(\mathbf{x}) \tag{S3}$$

where $\mathcal{T}^{\mu}$ and $\mathcal{T}^{\sigma}$ are the as-trained final model for μ and σ, respectively. Each pair of trees is trained as follows: in the $(t+1)$-th epoch, based on the current prediction values of μ and σ, which are $\mu_t = \sum_{i=1}^{t} \mathcal{T}_i^{\mu}(\mathbf{x})$ and $\sigma_t = \sum_{i=1}^{t} \mathcal{T}_i^{\sigma}(\mathbf{x})$, respectively, the negative-likelihood (NLL) loss $\mathcal{L}$ is computed:

$$\mathcal{L}(\mu_t(\mathbf{x}), \sigma_t(\mathbf{x}), P(\mathbf{x})) = -\log[\frac{1}{\sqrt{2\pi}\sigma_t(\mathbf{x})} \exp(-\frac{(P(\mathbf{x}) - \mu_t(\mathbf{x}))^2}{2\sigma_t^2(\mathbf{x})})]. \tag{S4}$$

Based on the NLL loss, the first-order and second-gradients of μ and σ, denoted as g_i^{μ}, h_i^{μ}, g_i^{σ} and h_i^{σ}, are computed. During the hybrid federated learning process, they will be playing a similar role to the deterministic gradients g_i, h_i in the manuscript.

B. Hybrid Federated Learning Based on Probabilistic XGBoost

Corresponding to the deterministic setting, the hybrid federated learning framework under the probabilistic setting also consists of 5 different subframeworks, enumerated in subframeworks **S1-1** to **S1-5** below. These subframeworks under the probabilistic setting also operate sequentially following the lead of **Framework 1** in the manuscript. All variables with superscripts μ and σ denote the μ and σ components of their deterministic counterparts in the manuscript.

Subframework S1-1: District-Level Sample Alignment *prior to training.*

```
1  For district $m = 1, 2, \cdots, M$:
2     Parties $C_m$ and $D_m$ perform secure sample alignment.
3     Party $C_m$ computes first-order gradients $g_i^{\mu}$, $g_i^{\sigma}$ and second-order gradients $h_i^{\mu}$, $h_i^{\sigma}$ based on (S4).
4     Party $C_m$ encrypts all first-order gradients into $[[g_i^{\mu}]]$ and $[[g_i^{\sigma}]]$.
5     Party $C_m$ encrypts all second-order gradients into $[[h_i^{\mu}]]$ and $[[h_i^{\sigma}]]$.
6  End For
```

Subframework S1-2: Secure XGBoost Binning *prior to training.*

```
1  For district $m = 1, 2, \cdots, M$:
2     $C_m$ transfers Paillier-encrypted gradients $[[g_i^{\mu}]]$, $[[g_i^{\sigma}]]$, $[[h_i^{\mu}]]$ and $[[h_i^{\sigma}]]$ to $D_m$.
3  End For
4  Select an active binning district $m_A$.
5  Based on FederBoost's secure bin construction algorithm,
6  Party $C_{m_A}$ leads the bin construction on features of all $C_m$s.
7  Party $D_{m_A}$ leads the bin construction on features of all $D_m$s.
```

Subframework S1-3: Secure Gradient Aggregation *for every node of a tree.*

```
1   Select an active training party $C_{m_B}$ from the label-holding parties.
2   For district $m = 1, 2, \cdots, M$:
3      Party $C_m$ filters out samples not belonging to the current node.
4      Party $C_m$ calculates all gradient statistics $G_{k,v,m}^{C,\mu}$,$G_{k,v,m}^{C,\sigma}$, $H_{k,v,m}^{C,\mu}$ and $G_{k,v,m}^{C,\sigma}$.
5      Party $C_m$ transfers $[[G_{k,v,m}^{C,\mu}]]$, $[[G_{k,v,m}^{C,\sigma}]]$, $[[H_{k,v,m}^{C,\mu}]]$ and $[[H_{k,v,m}^{C,\sigma}]]$ to $C_{m_B}$.
6      Party $D_m$ filters out samples not belonging to the current node.
7      Party $D_m$ calculates encrypted statistics $[[G_{k,v,m}^{D,\mu}]]$, $[[G_{k,v,m}^{D,\sigma}]]$, $[[H_{k,v,m}^{D,\mu}]]$ and $[[H_{k,v,m}^{D,\sigma}]]$.
8      Party $D_m$ transfers $[[G_{k,v,m}^{D,\mu}]]$, $[[G_{k,v,m}^{D,\sigma}]]$, $[[H_{k,v,m}^{D,\mu}]]$ and $[[H_{k,v,m}^{D,\sigma}]]$ to $C_{m_B}$.
9   End For
10  Party $C_{m_B}$ decrypts and sums gradient statistics
    $G_{k,v}^{C,\mu} = \sum_m \text{Dec}\{[[G_{k,v,m}^{C,\mu}]]\}$, $G_{k,v}^{C,\sigma} = \sum_m \text{Dec}\{[[G_{k,v,m}^{C,\sigma}]]\}$,
    $H_{k,v}^{C,\mu} = \sum_m \text{Dec}\{[[H_{k,v,m}^{C,\mu}]]\}$, $H_{k,v}^{C,\sigma} = \sum_m \text{Dec}\{[[H_{k,v,m}^{C,\sigma}]]\}$.
```

Subframework S1-4: Node Splitting *for every non-leaf node of a tree.*

```
1   Party $C_{m_B}$ calculates $G^{\mu} = \sum_k G_{k,1}^{C,\mu}$, $G^{\sigma} = \sum_k G_{k,1}^{C,\sigma}$, $H^{\mu} = \sum_k H_{k,1}^{C,\mu}$ and $H^{\sigma} = \sum_k H_{k,1}^{C,\sigma}$.
2   For feature $k = 1, 2, \cdots, K_C + K_D$:
3      For bin $v = 1, 2, \cdots, V$:
4         Party $C_{m_B}$ calculates Gain$^{\mu}(k, v)$ and Gain$^{\sigma}(k, v)$ based on (3).
5      End For
6   End For
7   Party $C_{m_B}$ obtains $k^{\mu *}$, $v^{\mu *}$ from Gain$^{\mu}(k, v)$, and $k^{\sigma *}$ , $v^{\sigma *}$ from Gain$^{\sigma}(k, v)$.
8   If $k^{\mu *}$ belongs to $C_{m_B}$, inform all $C_m$s of $(k^{\mu *}, v^{\mu *})$, otherwise inform all $D_m$s of it.
9   If $k^{\sigma *}$ belongs to $C_{m_B}$, inform all $C_m$s of $(k^{\sigma *}, v^{\sigma *})$, otherwise inform all $D_m$s of it.
10  The parties holding $k^{\mu *}$ (i.e., all $C_m$s or all $D_m$s) perform splitting of node samples on $\mathcal{T}_t^{\mu}$,
    and inform the other parties NOT holding it of the sample subspace of its child nodes.
11  The parties holding $k^{\sigma *}$ (i.e., all $C_m$s or all $D_m$s) perform splitting of node samples on $\mathcal{T}_t^{\sigma}$,
    and inform the other parties NOT holding it of the sample subspace of its child nodes.
```

Subframework S1-5: Leaf Value Calculation *for every leaf node of a tree.*

```
1  Party $C_{m_B}$ calculates $G^{\mu} = \sum_k G_{k,1}^{C,\mu}$, $G^{\sigma} = \sum_k G_{k,1}^{C,\sigma}$, $H^{\mu} = \sum_k H_{k,1}^{C,\mu}$ and $H^{\sigma} = \sum_k H_{k,1}^{C,\sigma}$.
2  Party $C_{m_B}$ sets the leaf value in $\mathcal{T}_i^{\mu}$ and $\mathcal{T}_i^{\sigma}$, respectively, based on (4).
```

C. Accuracy Performance under the Probabilistic Setting

Table SI (Corresponds to **Table II**) compares the Gaussian prediction losses using different machine learning algorithms: probabilistic Multi-Layer Perceptron, probabilistic Gated Recurrent Unit, probabilistic Random Forest and probabilistic XGBoost. One can see that probabilistic XGBoost is equally competent (and even slightly better) with the given datasets.

Table SI: NLL loss of bottom-level learning architectures on predicting load consumption.

Algorithm	**Training Set NLL**	**Test Set NLL**
Probabilistic Multi-Layer Perceptron	0.4092	-0.0443
Probabilistic Gated Recurrent Unit	-0.0639	-0.2644
Probabilistic Random Forest	0.8200	0.9420
Probabilistic XGBoost	-0.2688	-0.4200

Moreover, the four cases illustrated in Section V-A of the paper are again being constructed and compared. **Figure S1** (Corresponds to **Figure 6**) plots the NLL loss during training, using probabilistic XGBoost as the bottom-level learning algorithm. The same conclusion can be reached: The training losses of all four cases manage to converge, but the case with hybrid federated learning manages to reach the lowest NLL loss in 150 epochs, due to full collaboration on all external districts' and parties' data. Also, The Paillier homomorphic encryption technique exerts negligible influence on the accuracy of the as-trained model, indicating that it can help to preserve data privacy in collaborative learning, but leading to negligible accuracy compromises.

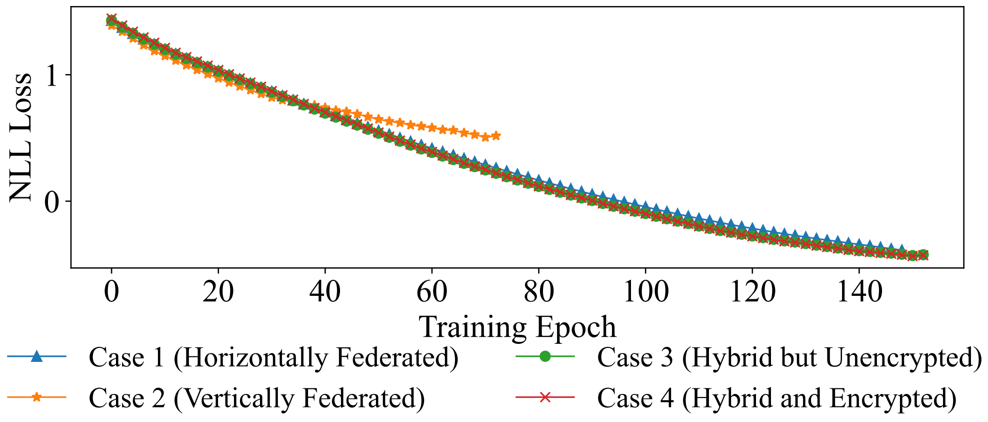

Figure S1: Normalized NLL loss on the test set for the 4 cases of federated learning.

Figure S2 (Corresponds to **Figure 7**) plots the probabilistic load prediction results on Districts 1 and 10, respectively, with different federated learning methods. One can see that vertical federated learning generates the poorest prediction results; on districts with more "regular" load profiles (e.g. District 1), horizontal federated learning might seem to be better, but hybrid federated learning would generalize better to more "unpredictable" load profiles (e.g., District 10).

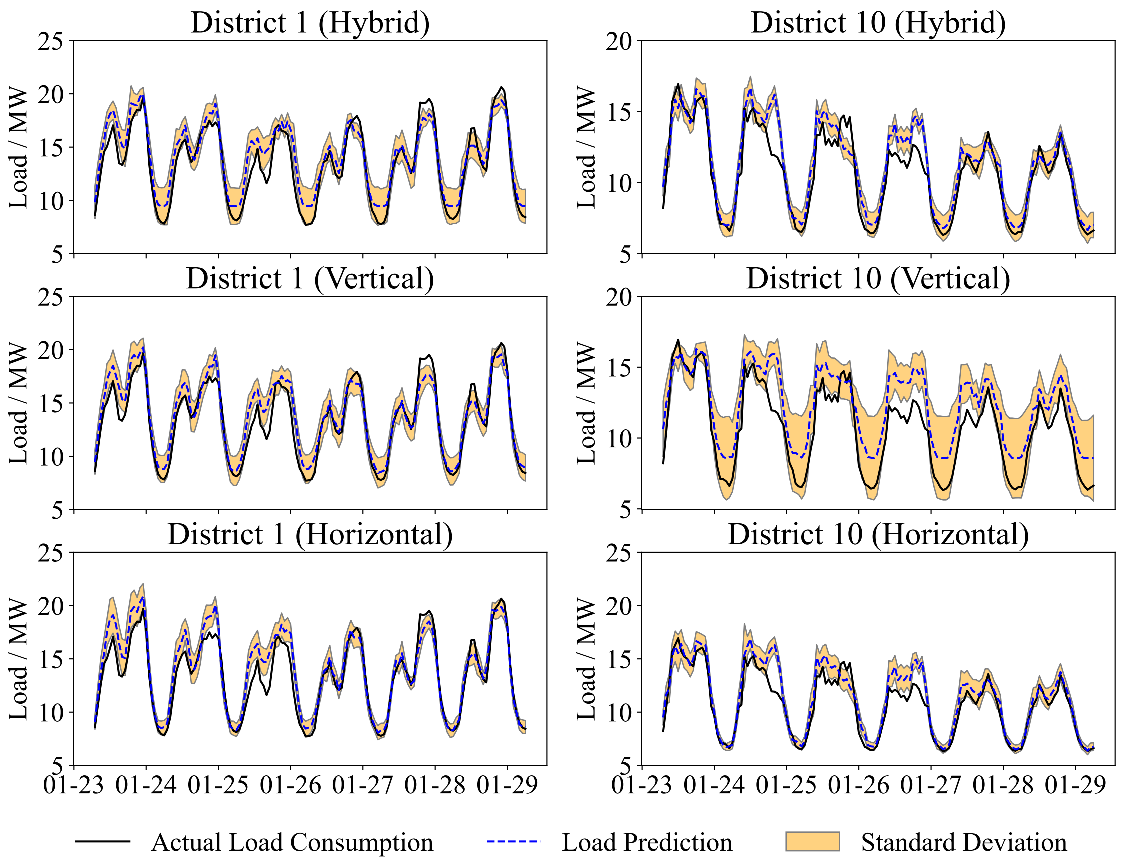

Figure S2: Probabilistic load prediction results on Districts 1 and 10, with different federated learning approaches.

Table SII (Corresponds to **Table III**) compares the prediction performances, in terms of the NLL loss, of models trained by horizontal, vertical and hybrid federated learning frameworks. XGBoost is selected as the bottom-level learning algorithm for FederBoost, SecureBoost and HybridFL, while GRU is selected for FedAvg, FedSGD and HeteroNN. Again, no framework achieves comparable performance to the hybrid federated learning framework, due to a lack of supplemental data from outside parties.

Table SII: NLLs of models trained by horizontal, vertical and hybrid federated learning frameworks.

Algorithm	**Type**	**Test Set NLL**
Horizontal	FedAvg	0.0512
	FedSGD	0.0513
	FederBoost (Case 1)	-0.3910
Vertical	HeteroNN	0.2533
	SecureBoost (Case 2)	0.5184
Hybrid	HybridFL (Case 4)	-0.4200

D. Generalization Performance with the Probabilistic Approach

Finally, **Figure S3** (Corresponds to **Figure 8**) plots a heatmap of the pairwise district NLL losses, to visualize the generalization ability of the single-district models. The same conclusion as that of the deterministic version can be drawn: single-district models generalize poorly to the data of other districts, while the model trained by hybrid federated learning generalizes well to the data of all districts.

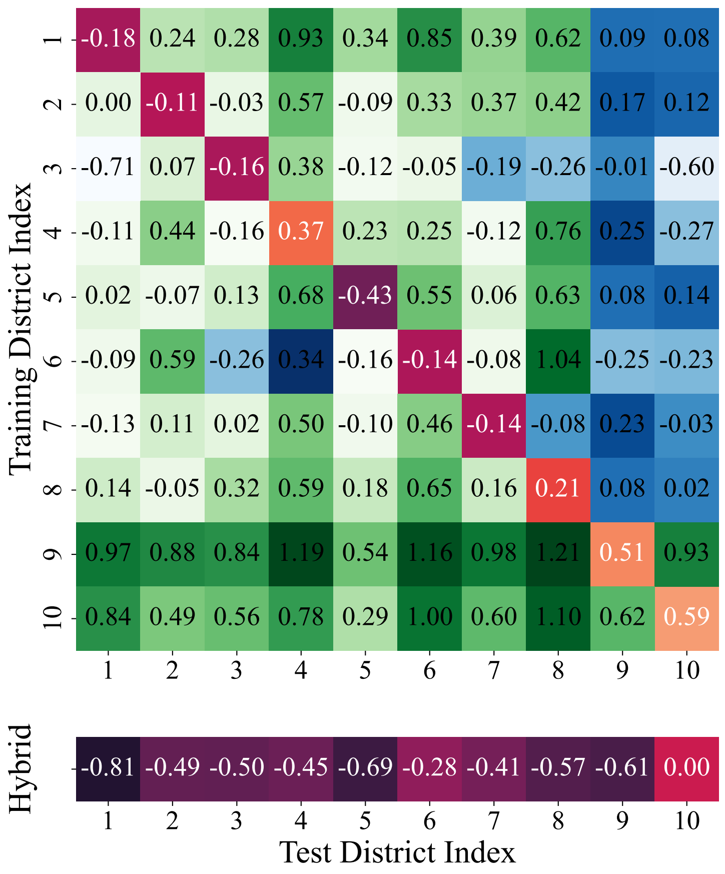

Figure S3: **Top**: Prediction NLLs with models vertically trained and tested on each single district. **Bottom**: Prediction NLLs with the hybrid federated learning model.