Federated Synchrophasor Data Prediction, Aggregation and Inference Using Deep Learning: A Case of Proactive Control for Short-Term Stability

Arman Ahmed, Sagnik Basumallik, *Member, IEEE*, Anurag K. Srivastava, *Fellow, IEEE*, Yinghui Wu, *Senior Member, IEEE*, Sutanay Choudhury

***Abstract*—A novel asynchronous federated architecture is proposed in this work for data prediction, aggregation, and inference with a use case for fast short-term instability mitigation in transmission systems. Existing machine learning (ML) prediction approaches rely on centralized-schemes that accumulate measurements from distributed regions at a central computing node. This centralized-schemes increase *computational overhead* leading to prediction delay. Federated-learning collaboratively learns a global model without centralized data processing. Nevertheless, its performance degrades when data becomes *non-independent and identically-distributed* due to the changes in grid dynamics, leading to expensive model re-training. This research proposes a *federated-inference architecture* that can exploit data-dependency in different regions to effectively exchange model parameters among high-quality local models and improve local predictive results, hence avoiding the need for global model and re-training effort. The architecture includes several major modules: a) federated deep ML models using gated-recurrent unit for local synchrophasor data prediction in each region, b) computationally-efficient data anomaly detection and mitigation using statistical and unsupervised autoencoders, c) a federated inference algorithm that aggregates model weights and outputs from top-k correlated neighbors to improve local performance, upon system dynamics change, d) instability prediction and e) a case analysis for its application in proactive short-term stability control. The efficiency and effectiveness of the proposed approach is validated on modified IEEE test systems with multiple test event scenarios with varying factors, including number of PMUs, error tolerances, and length of input and prediction windows.**

***Index Terms*—federated inference, machine learning, short-term stability, synchrophasor data, unsupervised learning**

I. Introduction

DIGITAL-automation, availability of synchrophasor data and advancement in machine learning (ML) enables real-time predictive analysis and proactive control to realize secure and resilient power grid. Among the critical applications is the proactive control to avoid short-term instability over a time frame of few seconds following a major disturbance [1], [2] such as fault or generator disconnection [3]. An unstable system results in large rotor angle oscillations, which if not mitigated promptly, may evolve into cascading failures.

While fast predictive analysis is important for proactive control, it is typically performed using (a) time domain simulations (TDS), which solves a non-linear differential algebraic equation system [4], or (b) ML-based predictive models [5], [6], [7], [8], [9], [10], [11], [12], [13] that learn from high resolution real-time phasor measurement unit (PMU) data. While TDS-based approaches often incur heavy computational overhead, ML models enable real-time prediction with prior training. A variety of ML models have been adopted for stability prediction, including support vector machine [6], autoencoders [5], artificial neural networks [7], graph neural networks [8], [9], convolutional neural network (CNN) [10], [11], hierarchical CNN [12] and long-short term memory networks (LSTM) [13].

While capable of accurate prediction of instability onset, these ML methods do not account for distributed nature of PMU data stored at different regional local servers [14]. Such regions are naturally formed based on their geographical proximity. Existing research often follows a centralized approach, where (geographically) distributed PMU data gets transmitted from the local servers to a centralized computing node to train a global predictive model. Nevertheless, it is often computationally expensive. Sending terabytes of PMU data to a central site from distributed regions incur significant transfer overhead, storage overload and packet drop [15], leading to delays in real-time short-term stability predictions.

Federated Learning (FL) is a recent formalism that does not require data sharing [16]. FL is a decentralized learning scheme where multiple sites collaboratively train a global model by locally updating shared model parameters that are synchronized at the central node, thereby reducing large data exchange. Nevertheless, FL often assumes that the data is independently and identically distributed (i.i.d), an assumption that may not hold with non-i.i.d PMU data from geographically dispersed regions having different data distribution due to different power grid operational characteristics [17], [18]. Changes in data distribution of particular regions often degrade the performance of the global model in prediction, reducing the overall accuracy [19]. This requires models to be re-trained (either locally or globally), which becomes time-prohibitive for rapid short-term applications such as instability prediction.

Can we do better and avoid expensive model retraining, especially over PMU data managed at distributed sites, when data sharing is expensive if not impossible? In this work, we develop a novel asynchronous *federated inference* (FI) architecture for data prediction. The fast output of the FI algorithm can be fed into any downstream prediction task in power system applications, such as proactive short-term stability mitigation. We summarize the contributions below.

1) Development of a federated prediction, aggregation and inference algorithm utilizing multiple computing agents

in local regions. Each agent independently monitors the quality of the results of pre-trained local models by only accessing locally observed PMU data; and calls a FI algorithm that dynamically determines a set of highly correlated local models from "neighboring" regions using top-k correlated neighboring models (top-kCN), and update local predicted results by aggregating their weighted output depending on spatiotemporal correlation. This *strikes a balance* between predictive quality and the maintenance cost of predictive models.

2) Development of an *asynchronous federated inference* minimizing unnecessary data exchange and training by (a) posing a guard condition to trigger data exchange only when necessary (b) only exchanging light-weighted messages that encode model weights and outputs, (c) performing at most two rounds of communication, all in parallel, in an asynchronous manner We also detect data anomalies and mitigate using statistical approach, Chebyshev inequality and unsupervised autoencoder not requiring labeled data before utilizing for prediction. We remark that our approach, unlike federated learning, is *not* to learn a global model or retrain local models from scratch, but aims to perform light-weighted fine-tuning of local models by exchanging small amount of correlated high-quality counterparts from nearby regions.
3) Introduction of a proactive short-term stability control to reduce the unnecessary need of remedial action schemes by taking proactive action based on the data-driven short-term stability prediction from the FI architecture. This is superior to existing solutions to address short-term instability problem such as: a) out-of-step protection triggering generator disconnection [20] and b) advanced event-based remedial action schemes (RAS) for generation rejection or load shedding without consideration of system-level impact [7], [21]. Our proposed idea of FI-based proactive short-term stability control is illustrated in Fig. 1. To the best of our knowledge, *this is the first effort that introduces federated inference-based proactive control action for fast prediction and short-term instability mitigation.*
4) Evaluation of the FI architecture experimentally on IEEE test system under diverse unstable event scenarios. The advantages of the FI-based proactive stability control approach is demonstrated over centralized and federated learning, where FI achieves an average of 30% improvement in accuracy.

The rest of the paper is organized as follows. Section II formulates the problem statement. Section III discusses the federated inference architecture. Section IV presents the results and discussions, and Section V presents the conclusion.

II. Problem Statement

We start with the following set up. Let $\mathcal{G}$ be a power system network of N (geographical) regions $\mathcal{G} = \{G_1, ..., G_N\}$. Each region G_i is a graph (network) with a set of nodes (buses) and edges (connected buses), and a corresponding agent A_i for coordinating data and local models with other agents. Moreover, at time t, each G_i has the following components:

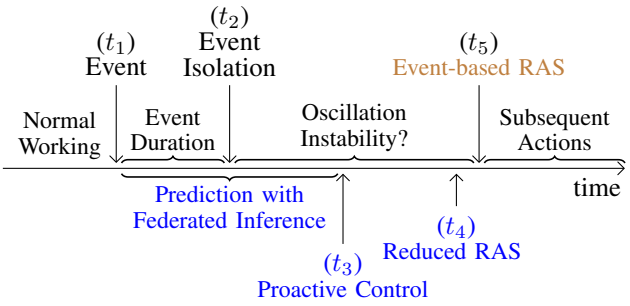

Fig. 1. Illustration of federated inference-based proactive short-term stability architecture. Let event onset and clearing be at t_1 and t_2 respectively. When instability is detected, pre-set event-based remedial action schemes trigger at t_5 (orange). With federated inference based early instability prediction, proactive actions are taken at $t_3 < t_5$ (blue), and reduced corrective actions, if needed, at t_4 (blue) to eliminate any unstable oscillations.

- Local data: *local PMU measurement data* X_i^t, and
- Local models: a pair of *local models* $M_i^t = (M_{bi}^t, M_{pi}^t)$.

We remark that our study does not pose assumptions on how the regions are constructured. In practice, the regions can be determined by criterias from network structures, administrative areas, or functional zones.

Local data. The local dataset $X_i^t = \{x_{t-\omega}, \ldots, x_t\}$ $(t \geq \omega)$ is a sequence of observed PMU measurements (packets) at time t in region G_i, induced by a sliding window of size ω. Each x_j $(j \in [t-\omega, t])$ is a vector of PMU data packets, where each packet is a vector of readings from a distinct PMU, encoding (featurized) $3-\phi$ voltage, current magnitudes and angles.

Local model. Agent A_i for region G_i $(i \in [1, N])$ manages a pair of pre-trained local models $M_i^t = (M_{bi}^t, M_{pi}^t)$ at time t. (a) The model M_{bi}^t refers to a *bad data detector*. It is a binary classifier that detects, for each entry $x_j^t \in X_i^t$, a boolean flag ('0' means 'normal', and '1' indicates a 'bad data'). (b) The model M_{pi}^t is a *predictive model* that takes as input a sequence of historical data D_i^t at time t and predicts a sequence of future measurements $\hat{D}_i^t$ at region G_i. When the context is clear that t refers to a "current" time, we simply denote D_i^t (resp. $\hat{D}_i^t$) as D_i (resp. $\hat{D}_i$) and M_{bi}^t (resp. M_{pi}^t) as M_{bi} (resp. M_{pi}).

As will be discussed, our architecture adopts a pipeline that couples local "bad data imputation" (with M_{bi}) and "stability prediction" (with M_{pi}) at each region G_i over currently observed data D_i (see Fig. 2). To "cold-start" and maintain FI-based stability control that tolerate low-quality or non i.i.d PMU input, we also use the following notation.

Neighbors. We say a local model M_i (resp. Agent A_i) is a neighbor of another M_j (resp. A_j) if there is an edge between two nodes, one in region G_i, and the other in region G_j. Our architecture explores neighboring models via message passing among corresponding neighboring agents to "propagate" high-quality predicted results. Each agent A_i maintains a set of all its neighbors, denoted as $\mathcal{N}(A_i)$.

Federated PMU Data Inference Problem. Given a power system network $\mathcal{G} = \{G_1, \ldots G_N\}$ with N agents, our goal is to develop a *federated inference architecture* that predicts short-term instability, such that (a) it only consults the local models M_i at each region G_i, (b) upon changed distribution

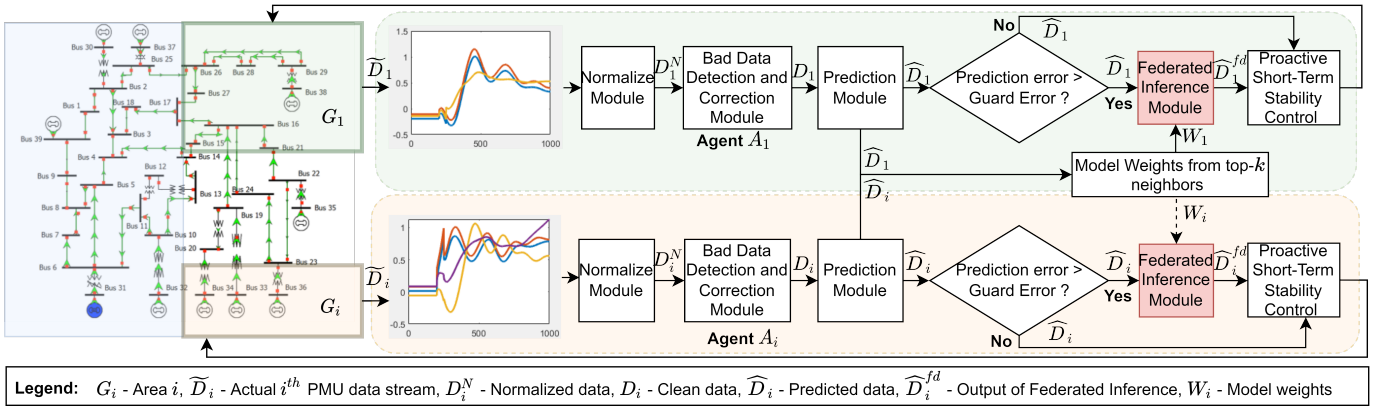

Fig. 2. Proposed federated inference architecture - for each area G_i, agent A_i analyzes PMU measurement data over window $\tilde{D}_i$. Data is first normalized, and bad data are imputed. The imputed data D_i is fed to the prediction module. When performance of prediction module degrades due to changes in PMU data distribution, the federated inference module is triggered, which aggregates model weights from top-k highly correlated neighbors to improve predictions. The output of the federated inference module now serves as an input for proactive short-term stability control actions against unstable oscillations.

of X_i^t, it only performs the necessary data exchange among neighboring agents to refine the local predicted results $\hat{D}_i$, *without* retraining M_i from scratch. With the early oscillation instability detection FI architecture, we then implement proactive short-term stability controls to stabilize the grid.

III. Federated Inference Architecture

We start with an overview of the proposed federated inference architecture (see Fig. 2), followed by module details.

A. Framework Overview

Bad Data Correction Module: Bad data in PMU occurs due to various reasons [22] e.g., harmonic interference, incorrect calibration of instrument transformers and gross errors. This module (discussed in Section III-B) detects and refines bad data in the input PMU measurements X_i. Bad data, usually detected as data anomalies, are labeled by the classifier M_{bi}, and imputed with built-in imputation method (using e.g., median of the data points [23]). This generates corrected inputs D_i to be fed to the prediction module.

Prediction: Given observed (imputed) input D_i (of length ω), the prediction module (Section III-C) uses a predictive model M_{pi} to forecast the expected measurements $\hat{D}_i$ in the next s timestamps ahead (where s is a configurable hyperparameter of M_{pi}). Predicting future measurement helps capture critical indicators ("events"), which in turn help early suggestion for taking suitable actions to mitigate unstable oscillations.

Federated Inference: Power events such as faults in region G_i introduce dynamics that change the distribution of the PMU measurements. As a result of non-i.d.d PMU data, the performance of the local prediction model M_{pi} may degrade. In such scenarios, it becomes imperative to retrain the ML model, which becomes computationally expensive and time-consuming over streaming PMU data. This shortcoming is addressed by the FI module, discussed in Section III-D. As shown in Fig. 2, when the performance of M_{pi} degrades due to non-i.i.d PMU data and the prediction error exceeds guard error ϵ_i, the FI module is initiated by agent A_i for a region G_i. Inside the FI module, performance of prediction model M_{pi} is improved by aggregating model weights from top-kCN of G_i. As shown in Fig. 2, when the prediction error of M_{pi} for region G_i is greater than guard error, then output $\{D_i, \hat{D}_i\}$ from the prediction module serves as an input to the FI module. The output of FI module, denoted as $\hat{D}_i^{fd}$, represents the refined predictions using the local model M_{pi} and its neighbors. The dataset $\hat{D}_i^{fd}$ serves as an input to the proactive short-term stability control module.

Proactive Stability Control: Availability of the high-resolution data prediction from the FI algorithm can enable multiple applications such as proactive control for mitigating impending power system instability. The proactive stability control module, discussed in Section III-E, enables rapid during-event actions that prevent system wide short-term instability power by accurate short-term prediction of PMU measurement data. When the FI-based prediction module forecasts that the system will continue to evolve towards instability in near future, proactive control actions such as generator reduction, reactive power injection and load rejection are triggered to arrest impending grid instability. The proactive actions are triggered prior to pre-set event-based RAS to reduce the magnitude of subsequent corrective actions.

B. Bad Data Detection and Correction Module

This module uses an autoencoder-based anomaly detection to capture bad data, M_{bi} and subsequently impute detected bad data to improve the quality of prediction. At any time t, the module (a) caches a sequence of historical (observed) PMU data $\tilde{D}_i$ with a window with length ω at each region G_i, and (b) normalizes $\tilde{D}_i$ and scales it to have mean $\mu = 0$ and standard deviation $\sigma = 1$ with $\frac{\tilde{D}_i - \mu}{\sigma}$ to ensure measurements with different range of values are standardized to a common scale. It "cold starts" an unsupervised training phase to obtain a pre-trained *autoencoder* M_{bi} (if M_{bi} does not exist yet), in a *training phase*. Otherwise, it enters a *detection phase* that applies M_{bi} to detect bad data in $\tilde{D}_i$.

Training phase. The training makes a practical assumption that a large fraction of the observed PMU data contains no bad data. The encoder compresses normal input D_i to a low dimension representation Φ using an encoder function f,

$$\Phi = f(D_i) = \boldsymbol{\sigma}(W_f \times D_i + b_f) \tag{1}$$

where b_f is the bias vector, W_f is the weight matrix and $\boldsymbol{\sigma}(.)$ is the sigmoid activation function. The decoder reconstructs the low dimension representation Φ to an output D_i' using a decoder function g,

$$D_i' = g(\Phi) = \boldsymbol{\sigma}(W_g \times \Phi + b_g) \tag{2}$$

where b_g is the bias vector, W_g is the weight matrix and $\boldsymbol{\sigma}(.)$ is the sigmoid activation function. The goal is to minimize a reconstruction error between D_i' and D_i, quantified by the root mean squared error (RMSE):

$$RMSE = \sqrt{\frac{1}{\gamma}\sum_{i=1}^{\gamma}(D_i - D'_i)^2} \tag{3}$$

where γ is the total number of data points in the training batch.

Detection Phase. Upon receiving observed input $\tilde{D}_i$, the pre-trained autoencoder M_{pi} is applied to infer a sequence of expected "normal" measurements D_i'. As M_{bi} learns how to recover "normal data", a high reconstruction error indicates the occurrence of bad data ("anomalies"). These measurement data points are then processed in the post processing phase using Chebyshev inequality.

Chebyshev's inequality. The Chebyshev's inequality is used to detect bad data when the distribution of data is unknown [24]. Let K represent the number of standard deviations σ from the mean μ of the data window $\tilde{D}_i$. For a given K, first a strict threshold $K\sigma$ is applied on $\tilde{D}_i$. Values of $\tilde{D}_i$ exceeding the first threshold are filtered out. A second threshold is then applied as $P(|\tilde{D}_i - \mu| \geq K\sigma) \leq \frac{1}{K^2}$, where the values μ and σ are *dynamically* computed over every sliding window, ensuring an adaptive detection process. The data points with values in $\tilde{D}_i$ that do not satisfy the above equation are labeled as '1' ("bad data"), and '0' otherwise.

Data correction. The labeled measurements are then imputed. Multiple data imputation approaches are available [23], [25]. By default, we choose median-imputation [23] as a pragmatic, light-weighted method to support fast data stream processing. The imputed data D_i is then fed to the prediction module.

C. Prediction Module

The objective of the prediction module is to predict PMU measurements s timestamps ahead using a pre-trained prediction model M_{pi} to identify trajectory of the power system oscillations. Our predictive model M_{pi} exploits gated recurrent units (GRUs) [26]. GRUs are a type of LSTM network unit capable of learning order dependence for time-series prediction problems. GRUs are selected among other types of LSTMs because GRU uses fewer parameters during training and requires less memory. GRUs consists of two gates (reset and update gate), while on the other hand, most of the other types of LSTMs consist of three memory gates (input, output, and forget gate). Hence, GRUs are more efficient than other types of LSTMs. The reset gate R_t of GRU determines how much of the previous information to forget and is given as,

$$R_t = \boldsymbol{\sigma}(\Theta_r \odot [D_i, H_{t-1}] + b_r) \tag{4}$$

where Θ_r is the parameterized tensor, H_{t-1} is the state from the previous time-step, b_r is the bias tensor and $\boldsymbol{\sigma}(.)$ is the sigmoid activation function. The update gate Z_t in GRU determines how much of the past information from the previous time-steps need to be passed to the future:

$$Z_t = \boldsymbol{\sigma}(\Theta_u \odot [D_i, H_{t-1}] + b_u) \tag{5}$$

where Θ_u represents the parameterized tensor, and b_u represents the bias tensor. Further, the hidden state in GRU is represented as, $\tilde{H}_t = tanh(\Theta_C \odot [D_i, (R_t \odot H_{t-1})] + b_c)$, where $tanh(.)$ is the tangential operation, and b_c is the bias tensor for the hidden cell state. Using the output from the reset gate, the update gate, and the hidden state, the final output H_t of the GRU is formulated as,

$$H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot \tilde{H}_t \tag{6}$$

Training Phase: The module is cold started at G_i by agent A_i with a training phase if the predictive model M_{pi} is not available. The learning objective is to reduce the error between the observed data D_i^o and the predicted data $\hat{D}_i$. The GRU model is trained using stochastic gradient descent algorithm to minimize the predictive error quantified as:

$$RMSE = \sqrt{\frac{1}{\gamma}\sum_{i=1}^{\gamma}(D_i^o - \hat{D}_i)^2} \tag{7}$$

where γ is the total number of data points in training batch i. The values of s, and the structure and hyper-parameters of the GRU are heuristically determined (see Section-IV).

Inference Phase: During inference phase, given sliding window data D_i from bad data module as input, the trained GRU model predicts a sliding window $\hat{D}_i$ of size s as output.

As shown in Fig. 2, for each window, agent A_i for region G_i continuously monitors the RMSE of model M_{pi} shown in Eq. (7). When the performance of M_{pi} degrades and the prediction error is greater than guard error ϵ_i (which can be set locally and adaptively), the agent A_i initiates the FI module to improve the local prediction. Note that at this moment, the impact of bad data is mitigated due to the imputation.

D. Federated Inference Module

Each individual region G_i independently, and simultaneously maintains a pre-trained local predictive model M_{pi}. During prediction phase, the model performance may degrade due to the non-i.i.d nature of PMU data caused by changes in system dynamics. The objective of the FI module is to improve the performance of the local model M_{pi} - this is achieved by aggregating model weights and outputs from a set of highly

Algorithm 1: Federated Inference (at each region G_i, executed asynchronously in parallel; time t omitted)

Input: local prediction $\hat{D}_i$, agent A_i, $\mathcal{N}(A_i)$, pre-trained models M_i;
Output: refined prediction $\hat{D}_i^{fd}$.
1: initialize list L_{ki} := $\emptyset$; set $\hat{D}_i^{fd}$:=$\emptyset$; set message W_{ki}:= $\emptyset$;
2: list L_{ki} := topkCN (A_i, D_i, $\mathcal{N}(A_i)$);
3: message W_{ki} := FedAgg ($\mathcal{N}(A_i), L_{ki}$);
4: $\hat{D}_i^{fd}$ = RfnPred ($M_{pi}, \hat{D}_i, W_{ki}$);
5: **return** $\hat{D}_i^{fd}$ (to Proactive stability control module);

Algorithm 2: Procedure topkCN: identify top-k Correlated Neighbors of A_i

Input: validated prediction D_i, A_i, $\mathcal{N}(A_i)$
Output: a list of top-k agents (models) L_{ki};
1: list L_k:= $\emptyset$;
2: A_i requests data ($\bar{\theta}$) from neighboring agents $\mathcal{N}(A_i)$;
3: **for** $j \in [\bar{\theta}_1, ..., \bar{\theta}_n]$ **do**
4: $r[i,j]$ = PearsonCorrelation($\bar{\theta}_i, \bar{\theta}_j$)
5: **end for**
6: L_{ki} = sortAscending(r);
7: **return** L_{ki};

correlated "neighboring" models. FI module leverages a cost-effective federated inference algorithm to refine the output of M_{pi} by a weighted aggregation strategy, as described below.

Asynchronous Federated Inference. We outline the asynchronous FI algorithm, which is executed in parallel at each single region G_i (as illustrated in Algorithm 1). At any time t, it takes as input the local predicted result $\hat{D}_i$, agent A_i and its neighbors $\mathcal{N}(A_i)$, and pre-trained models M_i = (M_{bi}, M_{pi}). Each agent A_i *asynchronously* performs a "superstep" that sequentially performs three primitive procedures at G_i, in two rounds of communications with $\mathcal{N}(A_i)$. This asynchronous process eliminates the need to wait for other agents to complete data transmission, avoiding delays and possibly multiple missed data window.

- Procedure topkCN (line 2, communication round 1): identify a list L_{ki} of top k most correlated neighboring models with M_i (see Algorithm 2);
- Procedure FedAgg (line 3, communication round 2): computes a weighted output of neighbor models by requesting and aggregating the outputs from L_{ki} (see Algorithm 3);
- Procedure RfnPred (line 4, locally at G_i): agent A_i then refines the local predicted results $\hat{D}_i$ with the weighted average of the outputs from neighboring models for downstream proactive stability control module.

We next describe Procedures topkCN and FedAgg (Procedure RfnPred is straightforward and omitted).

Procedure topkCN. The procedure (Algorithm 2) takes as input D_i, A_i and its neighbors $\mathcal{N}(A_i)$, and outputs the top-k correlated models (agents) of A_i. Specifically, it performs the following. (a) A_i requests each agent $A_j \in \mathcal{N}(A_i)$ to share a sketch of the *validated prediction* from M_j that passes the guard error test; Here we make case for bus voltage angle θ that are verified to be accurately predicted. An agent A_j may *refuse* to share, if M_j fails the guarded predictive error ϵ_j. Otherwise, A_j sends the average of voltage angle, $\bar{\theta} = \frac{1}{m}\sum_{i=1}^{m}\theta_m$ from m measurable PMUs in G_j. (b) Let the averaged voltage angles received be $[\bar{\theta}_1, \ldots \bar{\theta}_N]$ (for N neighbors). Pearson correlation coefficient is computed between the average voltage angle $\bar{\theta}_i$ of region G_i, and each $\bar{\theta}_j$ from $A_j \in \mathcal{N}(A_i)$, to derive a correlation score $r[i,j]$. (c) $\mathcal{N}(A_i)$ is then refined (by pruning agents that has no response or refused to share) and sorted according to the correlation scores, and top-k neighbors are chosen as L_{ik}.

Algorithm 3: Procedure FedAgg (at region G_i)

Input: top-k neighboring agents L_{ki}, agent A_i;
Output: aggregated model weights W_k.
1: Initialize model weight tensor W_k:=$\emptyset$;
2: **for each** neighboring agent $A_j \in L_{ki}$ **do**
3: A_i requests weight tensor W_{pj} of M_{pj} from A_j;
4: **if** $W_j \neq \emptyset$ and W_k=$\emptyset$ **then**
5: W_k := W_{pj};
6: **else**
7: **if** $W_j \neq \emptyset$ **then**
8: W_k := Avg(W_k, W_{pj});
9: **end if**
10: **end if**
11: **end for**
12: **return** W_k;

Procedure FedAgg. This procedure (Algorithm 3, at region G_i) takes as input the top-k correlated neighbors L_{ki}. For each neighboring agent $A_j \in L_{ki}$, agent A_i requests an auxiliary structure called *model weight tensor* W_j (simply "model weight") of the local predictive model M_{pj}. We define the term "model weight" [27] as a 3D matrix W (a tensor) of dimension $l*d*k$, which profiles the intermediate d-dimension embeddings of all the l-th layers of each of top k models. The averaged model weight W_k is then returned for the refinement of model outputs.

Procedure RfnPred. Upon receiving the averaged model weights, proposed approach refines the predicted results by (a) "replacing" the intermediate embedding representation of the layers of the local model M_i, and (b) re-applying M_i to infer an updated predicted results $\hat{D}_i^{fd}$.

Intuitively, each agent simulates federated learning to produce a surrogate "global" model, yet only use the intermediate embeddings of pre-trained neighboring (correlated) models which has validated high-quality output, hence avoid retraining of either global or local predictive models. Each agent A_i dynamically refines the local prediction by prioritizing and favoring "useful" layer embeddings from neighboring models L_{ki} in terms of "*how strong we are correlated in the past*" and "*are you a reliable neighbor*", thus leverage their pre-trained intermediate representations to "regenerate" refined local output towards more accurate prediction. This justification, as experimentally verified by our experiments (see Section IV) for proactive stability control, is consistent with recent study on general collaborative inference in [15], [28].

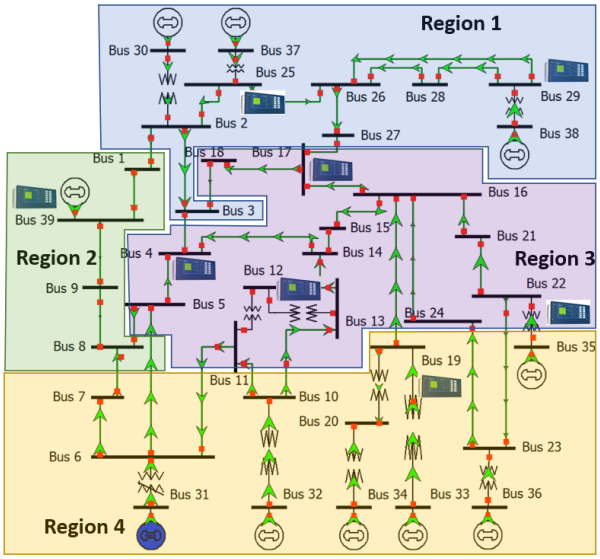

Fig. 3. The IEEE 39 bus system partitioned into 4 regions.

Cost analysis. The overall time cost of a single superstep at each agent consists of (1) at most $\mathcal{N}(A_i) + |\mathcal{N}(A_i)|^2 + |\mathcal{N}(A_i)| \log |\mathcal{N}(A_i)|$ time to obtain correlation scores and top-k neighbors from all $|\mathcal{N}(A_i)|$ neighbors; (2) at most $k \times \omega \times d$ time for aggregation collected model weights, where $\omega \times d$ is the size of the predicted results (with ω PMU packets, each with at most d number of values); and (3) the inference cost of the local model M_{pi}. Such cost is usually small (see Section IV). As the process is asynchronous, each step incurs at most 2 rounds of communication. No synchronization or re-training cost is incurred in this process.

E. Proactive Stability Control Module

We make case of the application of federated inference for cost-effective proactive stability control. These proactive schemes have a distinct advantage over out-of-step generator protection and event-based RAS. When a generator loses synchronism with the grid, out-of-step protection system disconnects the generator. Similarly, event-based RAS mitigate instability by generation rejection or load shedding. Both these control actions result in loss of assets. Generator tripping may subsequently lead to short-term and steady state stability violations. Event-based RAS actions may result in subsequent steady-state violations. Proactive control actions address these drawbacks through rapid during-event actions.

Proactive Action. Proactive control action is defined as during-event action that prevents system wide instability and reduces the magnitude of subsequent RAS. Similar to RAS, these proactive control actions are based on domain knowledge, extensive operator experience and pre-determined offline simulations considering wide range of operating scenarios. As shown in Fig. 2, the output of FI module (predicted PMU measurements $\hat{D}_i^{fd}$) serves as an input to the proactive stability control module.

IV. Simulation Results

The developed FI architecture for short-term proactive stability control is implemented on the IEEE-39 bus system with focus on federated inference and not the system size. A total load of 8903 MW and 2600 MVAR, and a total generation of 9109 MW and 5187 MVAR as well as total of 8 PMUs are placed at buses $4, 12, 17, 19, 22, 25, 29$ and 39, covering four communication sub-regions, as shown in Fig. 3. The details of all stable and unstable scenarios, generated using PowerWorld^{TM}, are given in Table I.

TABLE I
Details of Events Leading to Stable and Unstable Scenarios

ID	Event Type	Event Location	Event Onset Time (seconds)	Event Clear Time (seconds)	Stable/ Unstable
1	Bus Fault	16	5.0	5.1	Stable
			5.0	5.2	Unstable
2	Generator Trip	31	5.0	-	Stable
	Generator Fault	32	5.0	5.1	Unstable
3	3-Phase Line Fault	2 - 3	5.0	5.2	Stable
		5 - 6	5.0	5.1	Stable
		28 - 29	5.0	5.1	Unstable
4	Load Shed (30%)	4	5.0	-	Stable
5	Load Trip	25	5.0	-	Stable
6	1-Phase Line Fault	13 - 14	5.0	5.1	Stable
7	3-Phase Transformer Fault	29 - 38	5.0	5.1	Unstable
8	Generator Trip	35	5.0	-	Unstable

Evaluation metrics. We use five performance metrics - the first three metrics, *accuracy, precision, recall,* are calculated using four key terms: (1) False Positive (FP), (2) True Positive (TP), (3) False Negatives (FN) and (4) True Negative (TN). The F-1 score is evaluated as the harmonic mean of the precision and recall. The RMSE is calculated between D_i and predicted output $\hat{D}_i$, similar to eq (7). The effectiveness of the bad data detection module is evaluated using accuracy, precision, recall, and F-1 score, while the prediction and FI module are evaluated on the RMSE.

Hyper-parameter tuning. We tune the hyper-parameters with a grid search strategy [29]. We describe the tuning for the following important hyper-parameters.

Sliding window size (ω) and prediction window size (s). We aim to identify the "optimal" values for ω and s, under which the local bad data detection and predictive models achieves the best performance. Table II reports the impact of ω and s. On one hand, larger input window size yields lower error rates yet may incur longer processing time. On the other hand, it is more challenging to achieve higher accuracy for further "future" – by enforcing larger s, as expected. An "optimal" performance could be achieved with $\omega = 20$ and $s = 10$, achieving an accuracy of 97.97%, precision of 96.64%, recall of 97.96%, and F-1 score of 97.07% for bad data detection module and RMSE of 3.09×10^{-5} for prediction module, respectively, with a reasonable inference time of 0.22 seconds on average.

Impact of guard error threshold ϵ and k (# of neighbors). Table III reports the performance of FI algorithm with varied ϵ and k. In general, RMSE remains to be relatively higher if ϵ is set with higher values. For example, for $5.5 \times 10^{-3} \leq \epsilon \leq 10.5 \times 10^{-3}$ and $k \in \{1, 2\}$, the RMSE before and after FI was relatively higher, between 4.2×10^{-3} and 10.2×10^{-3}. Results on smaller ϵ e.g., (3.5×10^{-3}) indicate that FI improves RMSE better. Nevertheless, a small ϵ incurs more frequent triggers of FI algorithm, incurring

TABLE II
PERFORMANCE COMPARISON OF BAD DATA DETECTION AND PREDICTION MODULE WITH VARYING VALUES OF ω AND s

Input window size (ω)	**Prediction window size (s)**	**Bad Data Detection (performance averaged across all regions)**				**Prediction (performance averaged across all regions)**	**Inference time**
		Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE ($\times 10^{-3}$)**	**(seconds)**
	10	95.89	95.12	95.95	95.64	7.24	0.13
15	15	59.05	53.02	54.43	53.37	15.95	0.15
	25	44.32	43.35	45.49	44.92	45.95	0.19
	10	**97.97**	**96.64**	**97.96**	**97.07**	**3.09**	**0.22**
20	15	85.65	85.29	85.74	85.48	11.62	0.26
	25	45.32	45.59	46.32	45.88	61.58	0.29
	10	98.32	98.53	98.45	98.50	3.14	0.34
25	15	94.48	94.54	94.48	94.51	14.43	0.37
	25	68.65	65.39	65.12	65.27	61.34	0.41

TABLE III
PERFORMANCE COMPARISON OF FI WITH VARYING VALUE OF GUARD ERROR ϵ AND TOP-KCN

Guard Error ϵ	**RMSE ($\times 10^{-3}$) prior to FI**				**RMSE ($\times 10^{-3}$) after FI**								**Inference Time (seconds)**	
	Region 1	**Region 2**	**Region 3**	**Region 4**	**Region 1**		**Region 2**		**Region 3**		**Region 4**			
					k=1	*k=2*	*k=1*	*k=2*	*k=1*	*k=2*	*k=1*	*k=2*	*k=1*	*k=2*
3.5	4.4	4.3	4.4	4.7	3.3	2.1	3.2	2.4	3.3	2.9	3.4	3.1	0.14	0.20
4.5	4.9	4.6	4.8	4.8	4.2	**2.8**	4.1	**2.4**	4.4	**2.6**	4.2	**2.9**	0.17	**0.21**
5.5	6.4	6.3	6.5	5.9	5.2	4.0	5.3	4.1	5.1	4.9	5.3	4.3	0.13	0.22
6.5	6.8	6.7	6.4	6.8	6.3	5.8	6.2	5.6	6.2	5.2	6.3	5.5	0.15	0.21
7.5	7.9	7.6	7.7	7.9	6.9	5.5	6.9	5.3	6.3	5.8	6.6	5.2	0.13	0.25
8.5	8.7	8.8	8.6	8.6	8.1	7.7	8.4	7.3	8.2	7.9	7.9	7.3	0.14	0.22
9.5	9.8	9.9	9.4	9.6	9.3	8.8	9.1	8.3	8.9	8.7	8.2	8.9	0.13	0.23
10.5	11.1	10.9	10.8	10.8	10.2	9.9	10	9.3	9.6	9.1	10.3	9.9	0.15	0.24

TABLE IV
HYPERPARAMETERS OF LOCAL MODELS

Configuration	**Bad Data Detection Model (M_{bi})**			
	Region 1	**Region 2**	**Region 3**	**Region 4**
Number of layers	8 (4 encoder / 4 decoder)	6 (3 encoder / 3 decoder)	8 (4 encoder / 4 decoder)	8 (4 encoder / 4 decoder)
Number of neurons in respective layers	Encoder : [64, 28, 24, 12] Decoder : [12, 24, 28, 64]	Encoder : [28, 24, 12] Decoder : [24, 28, 64]	Encoder : [64, 28, 24, 12] Decoder : [12, 24, 28, 64]	Encoder : [64, 28, 24, 12] Decoder : [12, 24, 28, 64]
	Prediction Model (M_{pi})			
Number of LSTM layers	5	3	5	5
Number of LSTM cells in respective layers	[64, 64, 28, 24, 12]	[28, 24, 12]	[64, 64, 28, 28, 12]	[64, 64, 28, 28, 12]

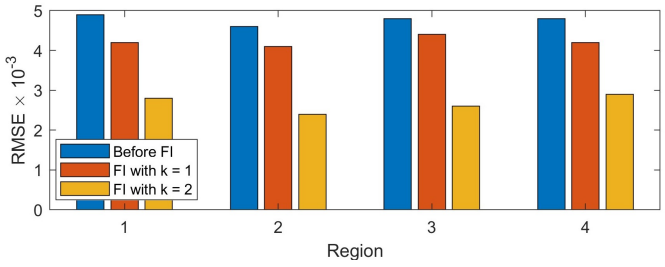

Fig. 4. RMSE before and after Federated Inference with $\epsilon = 4.5 \times 10^{-3}$

larger total overhead. We choose an optimal value of ϵ as 4.5×10^{-3}. Fig. 4 shows the RMSE values with different k; $k = 2$ has the lowest RMSE and is chosen as there are 4 areas, each having 2 neighbors.

Putting these together, in our tests, we set hyper-parameters for all local model training at all regions consistently as: $\omega = 20, s = 10, \epsilon = 4.5 \times 10^{-3}$ and $k = 2$ by default. Table IV shows the setting of hyper-parameters for the local models. Parameters are fine-tuned with grid search to ensure that local models achieved reasonably good performance at local dataset.

A. Ablation Analysis

The performance of the developed architecture with and without the FI module is compared in Table V. For all the six different events in Table V leading to instability, it is observed that the developed architecture with the FI module outperforms cases when FI is not implemented.

Consider the 3-phase bus fault event at bus 16 in region 3 as shown in row 1 of Table V. Without FI, the RMSE for region 3 is 5.8×10^{-3}. Compared to the case with FI module, the RMSE drops down to 3.3×10^{-3}, showing significant improvement. The detailed analysis for model performance improvement is as follows. The FI module is triggered between 5.0167 seconds and 5.0583 seconds when the prediction error $\epsilon \geq 4.5 \times 10^{-3}$. During this time, the FI module requests and receives model weights from highly correlated adjacent neighbors. In this case, the two chosen neighbors are region

TABLE V
ABLATION ANALYSIS - PERFORMANCE ANALYSIS WITHOUT AND WITH FI

Event	Region	RMSE ($\times 10^{-3}$) Without FI and (With FI)			
		Region 1	Region 2	Region 3	Region 4
Bus fault	3	3.2 (3.2)	3.7 (3.7)	5.8 (**3.3**)	3.4 (3.4)
3-phase Line Fault	1	5.7 (**3.1**)	3.4 (3.4)	3.1 (3.1)	2.8 (2.8)
3-phase Transformer Fault	1	4.9 (**2.7**)	2.5 (2.5)	2.2 (2.2)	3.2 (3.2)
Generator Fault	4	2.7 (2.7)	2.8 (2.8)	4.6 (**2.5**)	5.4 (**3.8**)
Line Trip	3	3.1 (3.1)	3.2 (3.2)	5.3 (**3.6**)	3.9 (3.9)
Generator Trip	3	2.7 (2.7)	2.2 (2.2)	5.3 (**3.9**)	4.2 (4.2)

TABLE VI
PERFORMANCE ANALYSIS - BEFORE AND AFTER FI

Event	Region	RMSE ($\times 10^{-3}$) Before FI and (After FI)				
		Region 1	Region 2	Region 3	Region 4	Global Average with FI
Bus fault	3	3.3 (3.3)	3.8 (3.8)	5.5 (**3.5**)	3.2 (3.2)	3.4
3-phase Line Fault	1	4.7 (**3.1**)	3.4 (3.4)	2.2 (2.2)	3.9 (3.9)	3.1
3-phase Transformer Fault	1	5.7 (**3.8**)	2.5 (2.5)	3.4 (3.4)	4.1 (4.1)	3.1
Generator Fault	4	3.4 (3.4)	4.0 (4.0)	3.9 (3.9)	5.3 (**3.3**)	3.6
Line Trip	3	2.5 (2.5)	2.5 (2.5)	5.2 (**3.2**)	3.8 (3.8)	3.0
Generator Trip	3	3.4 (3.4)	4.2 (4.2)	5.4 (**3.9**)	3.3 (3.3)	3.7

1 and region 4 with correlation of 0.92 and 0.75 respectively. The weights are then aggregated to improve the performance.

For the 3-phase transformer fault event in region 1 shown in row 3 of Table V, the RMSE for region 1 with and without FI is 4.9×10^{-3} and 2.7×10^{-3} respectively. In this case, the two chosen neighbors are region 2 and region 3 with correlation of 0.6 and 0.9 respectively. Similar observations are noted for each scenario in Table V. This shows that our developed architecture with the FI module is able to improve inference through weight sharing with highly correlated neighbors when local model performance degrades.

B. Performance of FI-based prediction and proactive control

The performance of the developed FI architecture, consisting of bad data detection module, prediction module, FI module and proactive stability control module, as shown in Fig 2, was evaluated on the IEEE-39 bus system for different events shown in Table I. For each region, sliding window $\omega = 20$, prediction widow $s = 10$, guard error $\epsilon = 4.5 \times 10^{-3}$, and top-kCN $= 2$ was set as per the hyper-parameter tuning analysis discussed in section IV. The results are summarized in Table VI. The performance metric corresponding to Table VI had an average accuracy of 97.97%, precision of 96.64%, recall of 97.96% and F1-score of 97.07%. The results of the

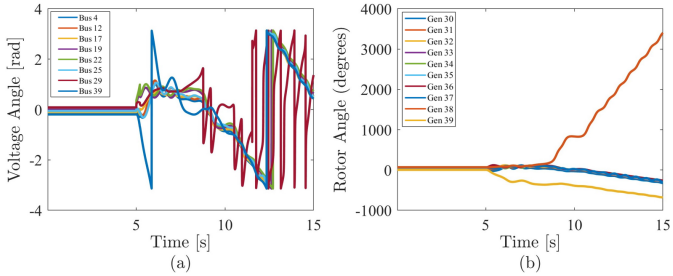

Fig. 5. Unstable scenario - 3-phase fault at bus 16 at $t = 5$ s with delayed clearing at $t = 5.2$ s. (a) Bus voltage angle and (b) generator rotor angle

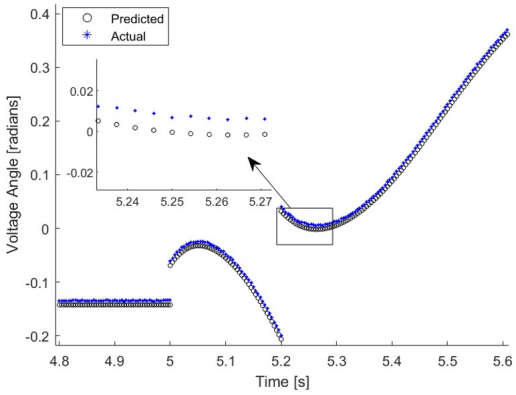

Fig. 6. Predicted voltage angle at PMU bus 17 before Federated Inference - high RMSE $= 5.5 \times 10^{-3}$

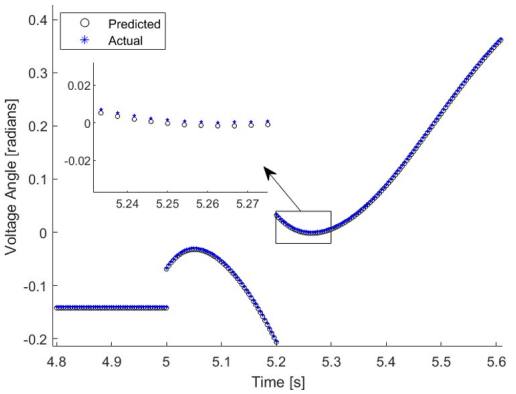

Fig. 7. Predicted voltage angle at PMU bus 17 after Federated Inference - low RMSE $= 3.5 \times 10^{-3}$

case study for the unstable scenario of 3-phase bus fault at bus 16 in row 1 of Table I is discussed in details next. The bus voltage angle and rotor angle corresponding to this unstable event is shown in Fig. 5.

Condition of short-term instability for the 3-phase bus fault event at bus 16 is simulated due to delayed fault clearing (case of stuck breaker or protection system malfunction) at $t = 5.2$ seconds. Measurement data was obtained from PMUs placed at buses $4, 12, 17, 19, 22, 25, 29$ and 39. All bad data was imputed in the bad data detection module, and the correct data was fed to the prediction module. The prediction model M_{pi} for region $N = 3$ predicts PMU data $s = 10$ time-steps ahead. However, due to change in system dynamics, the prediction RMSE $= 5.5 \times 10^{-3}$ was greater than guard error $\epsilon = 4.5 \times 10^{-3}$. Fig 6 shows the predictions with RMSE $= 5.5 \times 10^{-3}$, showing a considerable difference between predicted and actual PMU measurements. In this case, agent A_3 of region $N = 3$ triggered the FI module to improve the performance of M_{pi} model. The FI module identifies top correlated regions 1 and 4 with Pearson correlation coefficient 0.9 and 0.7, respectively. Model weights are then exchanged between A_1, A_4 and A_3. Fig 7 shows the predictions after FI module is invoked. It is observed that the performance of predictions after FI significantly improved with RMSE $= 3.5 \times 10^{-3}$. The average global RMSE for this case is 3.4×10^{-3}. The predictions from the updated model are fed to the proactive stability control module for instability detection

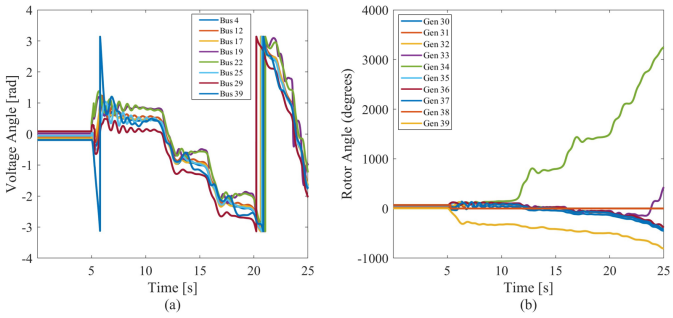

Fig. 8. Unstable scenario after generator 38 out-of-sync trip at $t = 5.28$ s, additional generator 34 and generator 33 goes out-of-step. (a) Bus voltage angle and (b) generator rotor angle

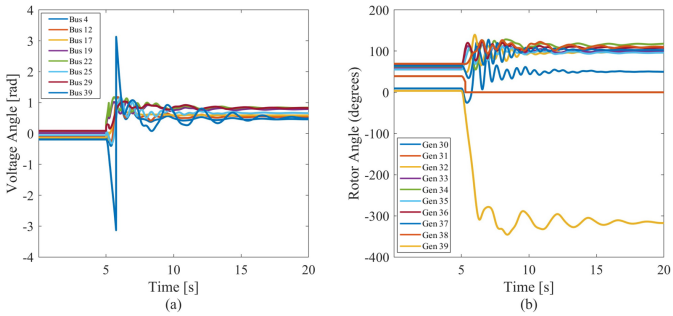

Fig. 9. Stable scenario after RAS action rejects generator 31 at $t = 5.55$ s. (a) Bus voltage angle and (b) generator rotor angle

and proactive control action.

As discussed earlier, the unstable conditions are simulated due to delayed fault clearing at $t = 5.2$ seconds. This results in an out-of-step condition for generator 38, resulting in loss of synchronization with the rest of the system, as shown previously in Fig. 5. First, the results of generator out-of-trip and event-based remedial schemes are discussed. Next, the results of proactive stability control are demonstrated.

At $t = 5.28$ seconds, the phase difference between generator 38 and slack bus 39 is greater than $120°$ and the generator 38 is tripped at $t = 5.28$ seconds with out-of-sync protection. The tripping is done before the phase angle difference between the generator and the system is greater than $120°$ as larger angles induce excessive stress on the circuit breaker due to dynamic recovery voltage [20]. As a result of generator 38 disconnection, it is observed that generator 34 and generator 33 go out of step. The bus voltage angle and rotor angle for this unstable case is shown in Fig. 8. In addition, lines $\{25-26, 17-27, 10-13, 25-37, 2-3\}$ are overloaded beyond 110% when generator 38 is out. The system is both short-term and steady-state unstable, which results in subsequent outages and may further evolve into cascading outages leading to a blackout.

To mitigate, there are multiple solutions: a) corrective control, b) event-based remedial action scheme (RAS) and c) proactive control. Generator tripping will be an example of corrective control. Activating a pre-set event-based RAS as a second possible control will trigger generator 31 rejection at $t = 5.55$ seconds. No further short-term instability occurs as seen from Fig. 9. However, line $2-3, 10-32$ are overloaded above 110% of system operating limits. A total of 572 MW

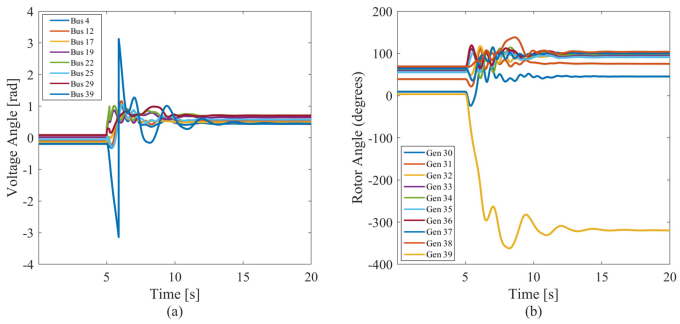

Fig. 10. Stable scenario after proactive control action sheds 2% output of generator 38 at $t = 5.32$ s; (a) Bus voltage angle and (b) generator rotor angle.

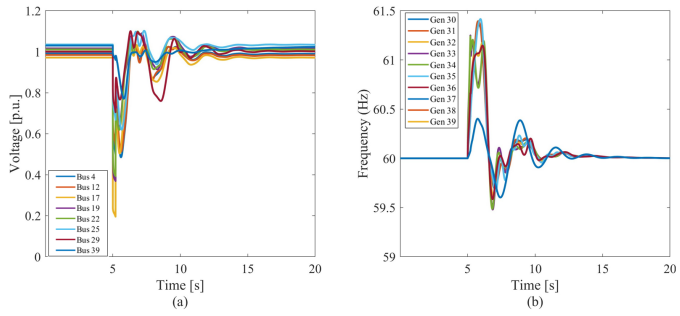

Fig. 11. Stable scenario after proactive control action sheds 2% output of generator 38 at $t = 5.32$ s. (a) Bus voltage and (b) frequency

of generation is lost and frequency stabilizes at 59.96 Hz. The third possible control is using the results of the FI-module for proactive short-term instability control. For proactive control, data prediction indicates impending instability triggering the MW output of generator 38 to proactively ramp down by 2% of its initial value of 1217 MW at $t = 5.32$ seconds. This action is chosen based on domain knowledge and pre-determined offline simulations. No further expected short-term instability occurs. This is evident from the voltage angle and rotor angle plots in Fig. 10, and bus voltage and frequency plots in Fig. 11. A total of only 16 MW of generation is shed and frequency is stabilized at 59.99 Hz. Simulation results clearly demonstrate that the proposed proactive control provides superior performance compared to the corrective control and event-based RAS for instability mitigation.

C. Comparison with Centralized and Federated Learning

We compare the performance of FI-based proactive short-term stability control over (1) the centralized approach and (2) the federated learning (FL) approach [30], in terms of (a) prediction accuracy, and (b) amount of data exchange.

Set up. For centralized training, a single GRU model was trained over a dataset gathered from all the regions in a single computing node. The GRU model consists of 9 layers with $256, 256, 128, 128, 128, 128, 64, 64$ and 64 GRU cells in respective layers. For FL, all regions collabratively train a global GRU model with 5 layers with $64, 64, 28, 28$ and 10 GRU cells in layers respectively, where each learns a local GRU with 4 layers of $28, 28, 24$ and 12 GRU cells respectively. For a fair comparison, all the training are consistently optimizing RMSE

TABLE VII
PREDICTION ERRORS FOR CENTRALIZED, FL AND FI

Event	Region	RMSE$\times 10^{-3}$						
		Centralized Approach	Federated Learning	Federated Inference				
				Region 1	Region 2	Region 3	Region 4	Global
3-phase line fault	1	6.6	5.9	3.1	3.4	3.1	2.8	3.1
3-phase transformer fault	1	8.4	6.3	2.7	2.5	2.2	3.2	2.6
Bus fault	3	9.2	8.1	3.2	3.7	3.3	3.4	3.4
Generator Fault	4	5.5	4.8	2.7	2.8	2.5	3.8	2.9
Line Trip	3	7.4	6.9	3.1	3.2	3.6	3.9	3.4
Generator Trip	3	6.9	6.4	2.7	2.2	3.9	4.2	3.2

TABLE VIII
PERFORMANCE ANALYSIS - BEFORE AND AFTER FI - 6 AREA SYSTEM

Event	Region	RMSE ($\times 10^{-3}$) Before FI and (After FI)						Global Average with FI
		Region 1	Region 2	Region 3	Region 4	Region 5	Region 6	
Bus Fault	1	5.2 (3.3)	3.4(3.4)	(3.1) (3.1)	3.6 (3.6)	3.2 (3.2)	3.4 (3.4)	3.3
3-Phase line fault	1	4.9 (3.2)	3.1 (3.1)	3.4 (3.4)	3.5 (3.5)	3.2 (3.2)	3.1 (3.1)	3.2
3-Phase Transformer Fault	1	5.8 (3.6)	3.5 (3.5)	3.7 (3.7)	3.8 (3.8)	3.1 (3.1)	3.2 (3.2)	3.4
Generator Fault	3	3.5 (3.5)	3.4 (3.4)	4.2 (3.1)	3.6 (3.6)	3.2 (3.2)	3.3 (3.3)	3.3
Line Trip	6	3.1 (3.1)	3.2 (3.2)	3.4 (3.4)	3.2 (3.2)	3.1 (3.1)	5.4 (3.3)	3.2
Generator Trip	2	3.1 (3.1)	5.9 (3.2)	3.2 (3.3)	3.4 (3.4)	3.2 (3.2)	3.1 (3.1)	3.2

loss, with an ADAM optimizer for stochastic gradient descent. The performances are reported in Table VII.

Accuracy and Time cost. It is observed that the RMSE error for the centralized and the FL architecture is higher than the developed FI architecture. The reason is attributed to that both centralized and federated learning aims to learn a global model under the assumption that PMU data is of high quality and follows i.i.d assumption. Under our test cases, PMU data at different regions may not necessarily follow a consistent distribution and can be noisy, thus a proper federated inference scheme over verified high-quality PMU data help local models to achieve their better results. Such observation has also been consistently verified in similar data scenarios [31], [32]. Further, both the centralized and the FL approaches require retraining to maintain their performance. In contrast, the RMSE error for every region and the global average with the FI architecture was the lowest among all three methods. Moreover, the average time taken across all regions by FI architecture was 0.25 seconds, while that of the centralized was 3.5 seconds and FL was 1.5 seconds.

Data shipment. During the inference process, for centralized method, in total 240 MB data has been gathered; in contrast, the FI architecture ships in total 197 KB data, around 0.08% of the amount of data shipment of centralized architecture.We estimate that FI improves the communication cost better over large-scale datasets. Consider a case where 100 PMUs with 20 measurements sampling at 120 messages per second transmits 200 GB of data per day [33]. In our tests, the FI inference architecture transmitting one measurement (i.e the average voltage angle) already achieves desirable accuracy, leading to 0.2 GB data shipment. Moreover, we estimate a much smaller amount of data shipment in practice, as FI incurs data shipment overhead only when triggered and when necessary.

D. Analysis on Larger System

Due to the unavailability of transient stability software for larger systems, the results of the FI approach are validated on a modified IEEE 30 bus system with more areas and a significantly more number of PMUs. This modified system has 6 areas and 16 PMUs placed at buses $1, 2, 4, 6, 8, 10, 14, 17, 19, 20, 22, 23, 26, 29, 30$ and 39, providing a total of 91 measurement streams. The parameters are set as $\omega = 22$ and $s = 10$. The similar experiment was repeated and the performance metric is shown in Table VIII with average accuracy of 96.49 %, precision 97.18 %, recall 95.03 %, and F1-score 97.31 %, effectively validating the proposed FI approach. The overall inference time was 0.23 seconds.

V. CONCLUSIONS

A novel federated inference architecture has been proposed in this work with an asynchronous strategy for data prediction, aggregation and inference with a use case of proactive control for mitigating short-term instability. The FI architecture enables collaborative inference to refine local predictions by adaptively identifying and aggregating the weights and outputs from highly correlated neighboring models only when necessary. The developed architecture incurs at most two rounds of communication for each agent, and avoids model retraining from scratch. We have utilized the FI architecture

to enable an example application for a novel proactive short-term stability control strategy that is experimentally verified to drastically reduce the impacts of traditional event-based remedial action schemes. The results obtained under diverse event scenario show superiority of FI based prediction and control. It is observed that predictions after FI significantly improved with 36% decrease in error, and the proactive control resulted in shedding of 95% less generation output compared to traditional remedial schemes. By implementing federated inference based proactive control to address short-term stability issues, the proposed approach mitigates the risk of equipment damage and the large costs associated with it. A future topic is to extend the architecture for federated causality analysis of events in large-scale power networks. Another topic is to evaluate our methods with more ecnomic indicators, more scenarios in protection systems, as well as extreme events such as natural disasters.

REFERENCES

[1] GIZ. (2013) Analysis of system stability in developing and emerging countries - impact of variable renewable energies on power system reliability and system security, deutsche gesellschaft für internationale zusammenarbeit. [Online]. Available: https://www.ctc-n.org/sites/www.ctc-n.org/files/resources/giz2013-en-power-system-stability-developing-emerging-countries.pdf

[2] X. Wang, T. Wang, H.-D. Chiang, J. Wang, and H. Liu, "A framework for dynamic stability analysis of power systems with volatile wind power," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 7, no. 3, pp. 422–431, 2017.

[3] N. Hatziargyriou, J. Milanovic, C. Rahmann, V. Ajjarapu, C. Canizares, I. Erlich, D. Hill, I. Hiskens, I. Kamwa, B. Pal *et al.*, "Definition and classification of power system stability–revisited & extended," *IEEE Transactions on Power Systems*, vol. 36, no. 4, pp. 3271–3281, 2020.

[4] M. Adibi, P. Hirsch, and J. Jordan, "Solution methods for transient and dynamic stability," *Proceedings of the IEEE*, vol. 62, no. 7, pp. 951–958, 1974.

[5] A. Sepehr, O. Gomis-Bellmunt, and E. Pouresmaeil, "Employing machine learning for enhancing transient stability of power synchronization control during fault conditions in weak grids," *IEEE Transactions on Smart Grid*, vol. 13, no. 3, pp. 2121–2131, 2022.

[6] B. Wang, B. Fang, Y. Wang, H. Liu, and Y. Liu, "Power system transient stability assessment based on big data and the core vector machine," *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 2561–2570, 2016.

[7] S. Naderi, M. Javadi, M. Mazhari, and C. Chung, "A machine learning-based framework for fast prediction of wide-area remedial control actions in interconnected power systems," *IEEE Transactions on Power Systems*, 2022.

[8] T. Zhao, M. Yue, and J. Wang, "Structure-informed graph learning of networked dependencies for online prediction of power system transient dynamics," *IEEE Transactions on Power Systems*, 2022.

[9] S. P. Nandanoori, S. Guan, S. Kundu, S. Pal, K. Agarwal, Y. Wu, and S. Choudhury, "Graph neural network and koopman models for learning networked dynamics: A comparative study on power grid transients prediction," *IEEE Access*, vol. 10, pp. 32 337–32 349, 2022.

[10] R. Yan, G. Geng, Q. Jiang, and Y. Li, "Fast transient stability batch assessment using cascaded convolutional neural networks," *IEEE Transactions on Power Systems*, vol. 34, no. 4, pp. 2802–2813, 2019.

[11] A. Gupta, G. Gurrala, and P. Sastry, "An online power system stability monitoring system using convolutional neural networks," *IEEE Transactions on Power Systems*, vol. 34, no. 2, pp. 864–872, 2018.

[12] L. Zhu, D. J. Hill, and C. Lu, "Hierarchical deep learning machine for power system online transient stability prediction," *IEEE Transactions on Power Systems*, vol. 35, no. 3, pp. 2399–2411, 2019.

[13] J. James, D. J. Hill, A. Y. Lam, J. Gu, and V. O. Li, "Intelligent time-adaptive transient stability assessment system," *IEEE Transactions on Power Systems*, vol. 33, no. 1, pp. 1049–1058, 2017.

[14] S. Nuthalapati, *Power system grid operation using synchrophasor technology*. Springer, 2019.

[15] M. Tang, X. Ning, Y. Wang, J. Sun, Y. Wang, H. Li, and Y. Chen, "Fedcor: Correlation-based active client selection strategy for heterogeneous federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 102–10 111.

[16] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 13, no. 3, pp. 1–207, 2019.

[17] A. Poudyal, U. Tamrakar, R. D. Trevizan, R. Fourney, R. Tonkoski, and T. M. Hansen, "Multiarea inertia estimation using convolutional neural networks and federated learning," *IEEE Systems Journal*, 2021.

[18] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[19] Z. Su, Y. Wang, T. H. Luan, N. Zhang, F. Li, T. Chen, and H. Cao, "Secure and efficient federated learning for smart grid with edge-cloud collaboration," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1333–1344, 2021.

[20] NERC, "Protection system response to power swings," 2013.

[21] H. Li, K. S. Shetye, S. Hossain-McKenzie, K. Davis, and T. J. Overbye, "Investigation of automated corrective actions for special protection schemes," Sandia National Lab.(SNL-NM), Albuquerque, NM (United States); Texas A & M, Tech. Rep., 2020.

[22] NASPI, "PMU data quality: A framework for the attributes of PMU data quality and quality impacts to synchrophasor applications," 2017.

[23] A. Gholami, A. K. Srivastava, and S. Pandey, "Data-driven failure diagnosis in transmission protection system with multiple events and data anomalies," *Journal of Modern Power Systems and Clean Energy*, vol. 7, no. 4, pp. 767–778, 2019.

[24] S. Pandey, A. K. Srivastava, and B. G. Amidan, "A real time event detection, classification and localization using synchrophasor data," *IEEE Transactions on Power Systems*, vol. 35, no. 6, pp. 4421–4431, 2020.

[25] M. Wang, J. H. Chow, D. Osipov, S. Konstantinopoulos, S. Zhang, E. Farantatos, and M. Patel, "Review of low-rank data-driven methods applied to synchrophasor measurement," *IEEE Open Access Journal of Power and Energy*, vol. 8, pp. 532–542, 2021.

[26] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[27] 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.

[28] Z. Hao, G. Xu, Y. Luo, H. Hu, J. An, and S. Mao, "Multi-agent collaborative inference via dnn decoupling: Intermediate feature compression and edge learning," *arXiv preprint arXiv:2205.11854*, 2022.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[30] H. Liu, X. Zhang, X. Shen, and H. Sun, "A federated learning framework for smart grids: Securing power traces in collaborative learning," *arXiv preprint arXiv:2103.11870*, 2021.

[31] N. Shlezinger, E. Farhan, H. Morgenstern, and Y. C. Eldar, "Collaborative inference via ensembles on the edge," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 8478–8482.

[32] J. Liu, C. Xie, O. O. Koyejo, and B. Li, "Copur: Certifiably robust collaborative inference via feature purification," in *Advances in Neural Information Processing Systems*.

[33] P. H. Gadde, M. Biswal, S. Brahma, and H. Cao, "Efficient compression of pmu data in wams," *IEEE Transactions on Smart Grid*, vol. 7, no. 5, pp. 2406–2413, 2016.