A Solar Forecasting Framework based on Federated Learning and Distributed Computing

Haoran Wen[a], Yang Du[b,*], Eng Gee Lim[a], Huiqing Wen[a], Ke Yan[c], Xingshuo Li[d], Lin Jiang[e]

[a]*School of Advanced Technology, Xi'an Jiaotong-Liverpool University, China*
[b]*College of Science and Engineering, James Cook University, Australia*
[c]*Department of the Built Environment, National University of Singapore, Singapore*
[d]*School of Electrical and Automation Engineering, Nanjing Normal University, China*
[e]*Department of Electrical and Electronics, University of Liverpool, UK*

Abstract

Solar forecasting is a crucial and cost-effective tool for better utilization of solar energy for smart environment design. Artificial intelligence (AI) technologies, such as machine learning (ML) and deep learning (DL), have gained great popularity and widely applied in solar forecasting in recent years. However, conventional AI-based forecasting methods suffer from high variability and stochasticity of solar irradiation, making unreliable predictions due to heterogeneous solar resources. Moreover, the training process of DL models is less flexible and requires immense data. Even for a well-trained model, it can still yield deteriorated performances on other datasets of varying data distributions. To tackle the deficiencies of AI forecasting models, we present a flexible distributed solar forecasting framework based on a novel spatial and temporal attention-based neural network (STANN) with federated learning (FL) technique, considering multi-horizon forecasting scenario from 5–30 min. The STANN model consists of a feature extractor and a forecaster, which can be respectively trained on various local datasets for better localization, and updated to further improve forecasting accuracy through global parameter aggregation under the proposed framework without data gathering. We evaluate effectiveness of the proposed method by conducting extensive experiments on real-world datasets and compare it to other popular forecasting models. The results demonstrate that our approach outperforms the other benchmarks with higher forecasting accuracy for all forecast horizons and better generalization on various datasets, achieving the highest forecast skill of 28.83% at 30-min horizon and an improvement of 11.2% compared with the centralized, localized, and conventional FL training methods.

Keywords: solar forecasting, deep learning, federated learning, distributed computing, sustainable environment

1. Introduction

Solar energy has sparked worldwide interest and acceptance as a viable alternative to fossil fuels due to its advantages of being environmentally friendly and inexhaustible [1]. In a symbiotic relationship to the development of solar energy, solar forecasting becomes one of the popular topics in the field of photovoltaic (PV) power generation [2], sustainable smart environment design [3], and smart cities [4]. For example, a weather forecasting platform is incorporated into energy management systems (EMSs) to anticipate onsite energy generation for better electrical and thermal consumption [5]. And the weighted ensembles of supervised learning models are used to predict thermal comfort exceedance under a possible climate change scenario [6]. However, solar forecasting remains a challenging task due to the highly variable and stochastic nature of solar irradiance. It hinders optimal solar energy harvesting and poses challenges to power grids, e.g., intermittent PV power generation caused by passing clouds can create power fluctuations, impair grid frequency balance, and finally lead to system instability and even blackouts [7]. Therefore, reliable solar forecasting is required for solar grid operation as well as passive solar architecture design for better solar energy utilization [8].

As a sub-domain of energy meteorology, solar forecasting refers to the prediction of both PV power generation and solar irradiation. Both of them can be achieved by using similar methods, however, power data is usually unavailable as it is private to PV power facilities. Thus, we only focus on the forecasting of global horizontal irradiance (GHI) because solar generation is directly affected by incident irradiation, and the irradiation-to-power conversion is well understood.

The existing solar irradiance forecasting approaches can be classified into three categories by predictive models: (1) physical models, (2) statistical models, and (3) hybrid approaches [9]. The physical models simulate coupled physical equations of the atmosphere to generate realistic predictions of solar irradiation based on numeric weather

*Corresponding author.
Email address: Yang.du@jcu.edu.au (Yang Du)

prediction (NWP) systems. For very short-term forecasting or nowcasting at minute to sub-minute levels, it tends to make poor predictions due to the difficulty in cloud motion/height estimation and the limited temporal and geographic resolutions [10].

Alternatively, statistical models make empirical predictions using historical irradiance observations. They can be applied to arbitrary forecast horizons from intra-hour to day-ahead depending on the temporal resolution of data. The established statistical methods, such as regressive models and machine learning models, usually adopt shallow networks with only a few hidden layers, thus having limited capability to handle complex nonlinearity [11]. To overcome this problem, deep learning (DL) models have been adopted to predict solar irradiation or power generation because of their sufficient ability for information extraction and nonlinearity representation. In [12], the authors utilize long-short term memory (LSTM) and recurrent neural networks (RNN) to forecast day-ahead PV power production with time correlation principles under a partial daily pattern prediction. Besides, a convolutional neural network (CNN) is applied to extract the underlying relationship between sky images and solar irradiance and make predictions for PV power ramp rate control [13]. A comprehensive review of DL approaches for renewable energy forecasting is presented in [11].

Hybrid methods integrate the merits of different forecasting techniques to improve forecasting accuracy by either combining multiple statistical models (hybrid-statistical) or applying statistical techniques to the physical modeling process (hybrid-physical) [14]. For example, a hybrid-statistical method proposed in [16] combines both CNN and LSTM networks and presents improved forecasting performance than the standalone CNN and LSTM. However, this stacked model requires immense training data and leads to increased training complexity and more computational resources. In [15], the authors compare physical deterministic models based on weather forecasts and a hybrid physical artificial neural network (ANN) for day-ahead PV power prediction. The results show that the hybrid physical-ANN combined achieves the best forecasting results than the other physical model.

In addition to model architectures, practical solar forecasting is always associated with a variety of data sources as well. Unlike the synthesized and well-elaborated datasets for certain tasks, solar data naturally differs in probability distribution due to the heterogeneity of solar resources. This paradigm violates the identically and independently distributed (IID) assumption commonly applied to neural networks and would increase the likelihood of errors and the complexity of modeling [16]. Moreover, there is little prior knowledge on how different input variables interact, including endogenous features derived from the observations, and exogenous features (e.g., wind speed, humidity, turbidity, and zenith angle). Thus, the heterogeneity of solar data makes solar forecasting particularly challenging, even for a sophisticated model.

In this work, we present a distributed multi-horizon solar forecasting framework based on a spatial-temporal attention-based neural network (STANN) and federated learning (FL) to leverage the above issues. The STANN model is built in the encoder-decoder structure consisting of a spatial feature extractor and a forecaster. The spatial feature extractor integrates attention mechanism to select the relevant input variables by assigning weights in the feature space. The forecaster employs attention layers in time space to provide insight into long-term temporal dependencies, thus improving forecasting accuracy. To make our model more robust and generalized on various data sources, we incorporate the STANN model with the FL technique into a distributed two-stage training strategy: (1) the STANN model is trained on each dataset to learn the local characteristics of each dataset; (2) after local training, the decoder parameters are aggregated by performing the FedAvg algorithm to further improve forecasting performance, while preserving the encoder parameters locally.

The contributions of this paper are summarized below:

1. A spatial-temporal attention-based STANN model is designed for multi-horizon solar forecasting. It contains a spatial feature encoder and a temporal decoder, which separately selects relevant features and provides insight into temporal dynamics, enabling skillful multi-horizon solar forecasting for a wide range of input features.

2. We proposed a distributed forecasting framework using FL in favor of a customized feature extractor to fit divergent data sources, while collaboratively training the forecaster without data sharing. To investigate the effectiveness of the proposed approach, the results from the conventional centralized and localized training methods are also compared and discussed.

3. We carry out extensive experiments on various real-world datasets and input features. The results demonstrate improved accuracy and better generalization of various data sources, making it more applicable to practical forecasting scenarios.

The remainder of the paper is organized into the following four sections. In Section II, a brief literature study is presented. Section III introduces the data used in simulations. In Section IV, we introduce the STANN model with the FL separable training method. Section V presents the simulation results with discussions and is followed by a conclusion with potential future works in section VI.

2. Related work

2.1. AI-based multi-step time series forecasting

From a statistical perspective, multi-step time series forecasting can be considered as a variation of the one-step ahead forecasting problem. It can be further divided

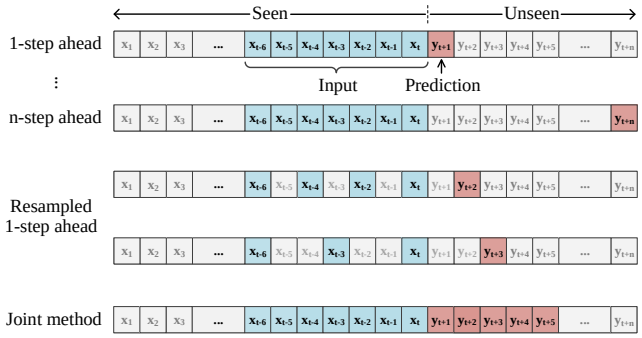

Figure 1: Comparison of direct and joint multi-step ahead forecasting methods.

into *iterative*, *direct* and *joint* approaches [17]. Iterative methods typically make use of auto-regressive models by recursively feeding one-step ahead predictions into future input data samples, e.g., related works in [18, 19, 20]. However, the main limitation of the iterative methods is that the recursive structure could result in large error accumulations over long forecast horizons when small forecasting errors are produced at each time step.

In contrast, direct approaches (e.g., [21, 22]) require a collection of models that are trained to predict each N-step ahead. In addition to training multiple models, the main drawback is the deterioration in forecasting accuracy at longer horizons due to a weakened temporal correlation. Alternatively, studies in [13, 23] utilize resampled time series for training. The resampled data have the same temporal resolution for a specific forecast horizon, such that the predictions are always one-step ahead regardless of how long the forecast horizon is. Nonetheless, it results in a reduced amount of data by a factor of $\frac{1}{N}$ with sparse temporal information in the time series as the forecast horizon increases. For example, a weighted gaussian process regression model is proposed in [24] for 10 steps-head daily global and direct solar radiation forecasting. The authors compared both direct and iterative approaches by using paralleled and cascaded architectures, respectively. The results showed that the cascaded model achieved better accuracy than the direct method by considering the temporal correlation.

In the joint strategy, also known as multiple input multiple outputs (MIMO), the objective is to train a capable model (e.g., recurrent and Seq2Seq models) to directly generate multiple forecasts in a one-shot manner. A comparison between the direct approaches and the joint method is shown in Fig. 1. Recently, Transformer [25] architectures based on attention mechanisms have achieved great success in natural language processing and other Seq-2-Seq tasks. The attention mechanism is also applied to other architectures for multivariate time series forecasting tasks [26, 27]. For example, in [26], the authors predict multi-step citywide passenger demand based on a graph with a hierarchical graph convolutional structure, where the attention-based module is to model the dynamic temporal information. In [28], a self-attention-based Transformer model has been utilized for multivariate solar time-series forecasting. The method achieves evident improvements compare to the LSTM model and its attention-based variants on two different datasets. However, these often fail to consider the different types of inputs commonly present in multivariate time series forecasting, and either assume that the exogenous inputs are known into the future [29].

2.2. FL-based forecasting

Federated learning (FL) is a distributed stochastic gradient descent (SGD) procedure that allows a selection of devices to train on local data and contribute updates to a shared model [30]. This learning technique has shown promising prospects in many fields, such as Internet of Things (IoT)-based energy control in smart buildings [31, 32], public health [33], traffic prediction [34], and load forecasting [35, 36]. The work of [37] applies FL to predict the socio-demographic characteristics for energy utilities to offer diversified services to their electricity consumers. The authors in [35] provide a simple study of FL for electrical load forecasting using a client cluster method and compare it with centralized and localized forecasting. The results suggest that FL can be used only for individual load forecast problems in cases where access to training data is not possible because FL outperforms centralized forecasting, but is worse than localized forecasting. A recent study in [38] proposes an innovative federated deep generative learning framework for renewable scenario generation. The model outperforms the state-of-the-art centralized methods. Besides, different federated learning settings are designed and performed that demonstrate better robustness of the method. In [39], an FL-based Bayesian neural network (FL-BNN) is proposed to preserve the privacy of utilities in the behind-the-meters (BTMs) estimation with a layerwise parameter aggregation strategy enabling a customized model for each client. The FL-BNN model is slightly better than that of the centralized BNN model and is much better than the other benchmarks. To the best of our knowledge, the only existing work related to FL-based solar forecasting is introduced in [40]. The authors propose a federated probabilistic solar forecasting scheme based on variational Bayesian. This method presents slightly improved accuracy in comparison to the centralized model on the data collected from a limited area of 64 square km. However, the work lacks an in-depth analysis of the impact of different input variables regardless of the fact the time series used for training are highly correlated.

Although, the decoupled FL training process keeps raw data decentralized, the learning objective is to optimize a shared model which can be vulnerable to non-IID data [41]. Therefore, the initiative of the proposed forecasting scheme is to train multiple customized models fitting various real-world data sources instead of a global model.

Table 1: NREL datasets information

Site	State	Length	Location	Elevation [m]	TZ
SRRL BMS	Colorado	46,342	(39.742, -105.182)	1828.8	PST
ECSU	North Carolina	46,584	(36.282, -76.216)	26	EST
UO SRML	Oregon	45,813	(44.046, -123.074)	133.8	PST
UNLV	Nevada	46,706	(36.107, -115.143)	615	PST
La Ola Lanai	Hawaii	47,428	(20.766, -156.922)	381	HST
ARM RCS	Kansas	45,245	(36.606, -97.486)	320	CST
SW Solar	Arizona	46,876	(33.416, -123.108)	347	MST
HSU SoRMS	California	46,219	(40.875, -124.081)	36	PST

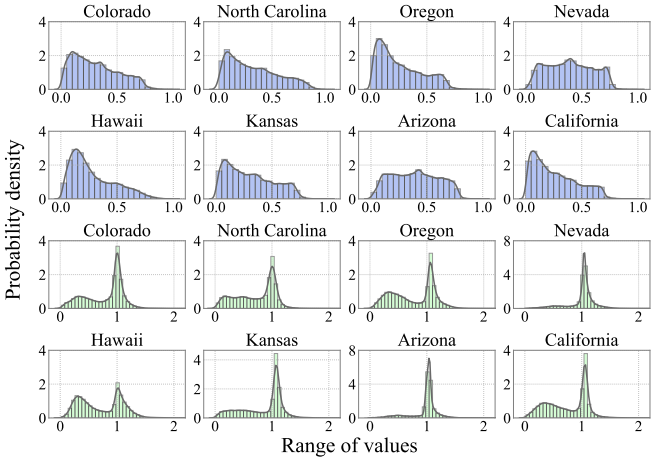

Figure 2: Probability density of the GHI (blue) and the CSI (green) time series of the NREL datasets. The GHI values are scaled in a range of [0, 1] by a *MaxMinScaler* for better visualization.

3. Data description

3.1. Datasets

Folsom dataset: This comprehensive dataset is published in [42] for benchmark solar forecasting models. It contains 3-year (2014-2016) solar irradiation data in 5-min resolution with well-elaborated endogenous and exogenous features at Folsom, California. We use 2014 and 2015 as training set and 2016 as out-of-sample testing set to compare the STANN with other benchmark models. The dataset contains multiple features and covers annual solar patterns under different weather such that model performance can be fully evaluated.

NREL datasets: We utilize solar irradiance measured by the baseline measurement system [43] of the National Renewable Energy Laboratory (NREL). We select 8 measurement sites at different locations in the US that cover various weather patterns. The data from each site contains 1-year observations of global horizontal irradiance (GHI) at an interval of 1 min. The data distribution of each dataset is shown in Fig. 2. The data is then averaged to 5-min resolution and the nighttime values are removed according to solar zenith angle ($\theta_z > 85°$ during nighttime) since they have no impact on PV power production. We divided each dataset into 70% of the training set and 30% of the out-of-sample test set for evaluating the proposed

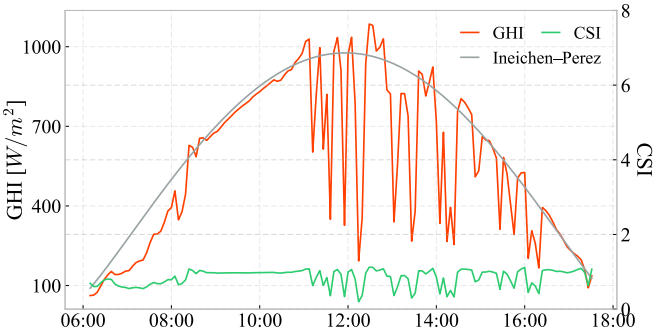

Figure 3: Diurnal variation of GHI and CSI on 30 July, 2012 in Arizona USA.

distributed learning framework. The detailed information is presented in Table 1.

3.2. Data processing

3.2.1. Clear-sky index

We first calculate clear-sky index (CSI) as input time series for all forecasting models instead of directly predicting GHI values. It is defined as the ratio between the measured irradiance and its expectation under the cloud-free atmosphere, which is written as:

$$k^{(\mathrm{cs})} = I/I^{(\mathrm{cs})} \tag{1}$$

where $k^{(\mathrm{cs})}$ is CSI, I is actual GHI and $I^{(\mathrm{cs})}$ is corresponding clear-sky irradiance. Using CSI is a common practice to remove seasonality and to improve accuracy in solar forecasting, which is equivalent to data normalization in other time series forecasting tasks. The predicted CSI can be converted back to GHI by multiplying corresponding clear-sky values as a reversed process in equation 1. In this study, Ineichen–Perez clear-sky model [44] based on Linke turbidity is used to retrieve CSI using pvlib [45] packaged in Python. An example of daily GHI variation and its corresponding CSI values are illustrated in Fig. 3.

3.2.2. Feature extraction

Feature extraction is the process of extracting features from raw data using data mining techniques to improve the performance of machine learning algorithms. Following [42], we featurize the derived CSI time series into three variables, namely backward average values ($\mathcal{B}$), lagged average values ($\mathcal{L}$), and variability ($\mathcal{V}$):

$$\mathcal{B}_i(t) = \frac{1}{i\delta}\sum_{t-i\delta}^{t} k^{(\mathrm{cs})}(t), \quad i = \{1, 2, \ldots, N\} \tag{2}$$

$$\mathcal{L}_i(t) = \frac{1}{\delta}\sum_{t-(i+1)\delta}^{t-i\delta} k^{(\mathrm{cs})}(t), \quad i = \{1, 2, \ldots, N\} \tag{3}$$

$$\mathcal{V}_i(t) = \sqrt{\frac{1}{i\delta}\sum_{t-i\delta}^{t}\Delta k^{(\mathrm{cs})}(t)^2}, \quad i = \{1, 2, \ldots, N\} \tag{4}$$

where δ denotes the temporal resolution of data and N is the number of horizons. In this research, we perform intra-hour forecasts (i.e., 5–30-min at a step of 5-min) so that $\delta = 5$ and $N = 6$ for 6 forecast horizons. As for other features, such as weather-related variables, their effectiveness depends on one's understanding of the predictors, and how they affect the predictand. Adding irrelevant predictors might result in overall prediction variance [46]. Therefore, we do not include any other meteorological variables.

3.2.3. Data transformation

After the feature extraction, the data is in a form of a 2-dimensional multivariate matrix:

$$\mathbb{D} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^m \\ x_2^1 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \ddots & \vdots \\ x_n^1 & x_n^2 & \cdots & x_n^m \end{bmatrix} \in \mathbb{R}^{n\times m}$$

where n is the length of the data and m is the input size. It is then rearranged into sequences over a look-back window of length s for DL time series forecasting models:

$$\mathbb{D} = [\mathbf{X}_s, \mathbf{X}_{s+1}, \ldots, \mathbf{X}_t \ldots, \mathbf{X}_n]^\top \in \mathbb{R}^{(n-s)\times s\times m}$$

where $\mathbf{X}_t = [\mathbf{x}_{t-s}, \mathbf{x}_{t-s+1}, \ldots, \mathbf{x}_t] \in \mathbb{R}^{s\times m}$ is input sequence corresponding to a target vector $\widehat{\mathbf{y}}_{t+\tau} = [y_{t+1}, y_{t+2}, \ldots, y_{t+\tau_{\max}}]$ for multi-horizon forecasting, where $\tau \in [1, 2, \ldots, \tau_{\max}]$ is a discrete forecast horizon.

4. Methodology

This section introduces the proposed distributed solar forecasting framework based on federated learning. First, a spatial-temporal attention-based neural network (STANN) is specialized to incorporate into the framework as the forecasting model. Then, we distinguish our framework from the conventional FL setting by introducing a novel two-stage training strategy to fit diverse data distribution. An overall schematic view of this framework is illustrated in Fig. 4.

4.1. STANN forecasting model

In time series forecasting, the encoder-decoder architecture has exhibited cutting-edge results by encoding the temporal information underlying the historical observations into a latent vector $\mathbf{z}_t$ and generates the final forecast using the vector as follows:

$$\mathbf{z}_t = \mathrm{f}_{\mathrm{enc}}(\mathbf{X}_t, \mathbf{y}_{t+\tau})$$

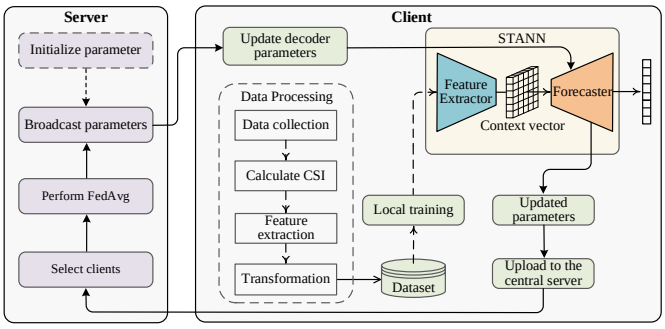

Figure 4: The proposed distributed solar forecasting framework. The dashed line means the step is only executed once at the beginning of the updating rounds.

$$\widehat{\mathbf{y}}_{t+\tau} = \mathrm{f}_{\mathrm{dec}}(\mathbf{z}_t)$$

where $\mathrm{f}_{\mathrm{enc}}(\cdot)$ and $\mathrm{f}_{\mathrm{dec}}(\cdot)$ are encoder and decoder functions respectively. By taking advantage of this architecture, we propose a STANN model that consists of a spatial feature extractor and a forecaster shown in Fig. 5. The details of both the feature extractor and forecaster will be presented in the following sections.

4.1.1. Spatial feature extractor

The spatial feature extractor is used to aggregate spatial information among different input features using dynamically generated weights, further enabling the recurrent layers directly focus on the significant time dependencies – even far back time steps in the sequence. In this case, we apply additive attention [47] function that uses a single hidden layer with two linear transformations and a tanh activation in between. Particularly, given an input sequence $\mathbf{X}_t = [\mathbf{x}_{t-s}, \mathbf{x}_{t-s+1}, \ldots, \mathbf{x}_t] \in \mathbb{R}^{s\times m}$ at position t, we can first compute attention scores $\boldsymbol{\varepsilon}_t$ over the feature space, as a function of the previous hidden states $\boldsymbol{h}_{t-1}$ and the input sequence $\mathbf{X}_t$:

$$\begin{aligned} \boldsymbol{\varepsilon}_t &= \mathrm{a}(\boldsymbol{h}_{t-1}, \mathbf{X}_t) \\ &= \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_a \boldsymbol{h}_{t-1} + \boldsymbol{U}_a \mathbf{X}_t\right) \end{aligned} \tag{5}$$

where $\mathrm{a}(\cdot)$ is the alignment function that evaluates how well the input data match with the previous hidden state and determines which input values x_t^i in i-th feature at time t should focus on. $\boldsymbol{W}_a \in \mathbb{R}^{s\times p}$, $\boldsymbol{U}_a \in \mathbb{R}^{s\times s}$ and $\boldsymbol{v}_a \in \mathbb{R}^s$ are learnable parameters that can be jointly optimized by any back-propagation algorithms with the recurrent layers. And $\boldsymbol{h}_{t-1} \in \mathbb{R}^p$ is the previous hidden state vector and p is the size of the encoder hidden state vector.

The attention weights $\boldsymbol{\alpha}_t$ are then obtained by applying a softmax function over the attention scores $\boldsymbol{\varepsilon}_t = [\varepsilon_t^1, \varepsilon_t^2, \ldots, \varepsilon_t^m] \in \mathbb{R}^m$:

$$\boldsymbol{\alpha}_t = \mathrm{softmax}(\boldsymbol{\varepsilon}_t) = \frac{\exp\left(\boldsymbol{\varepsilon}_t\right)}{\sum_{j=1}^{m}\exp\left(\varepsilon_t^j\right)} \tag{6}$$

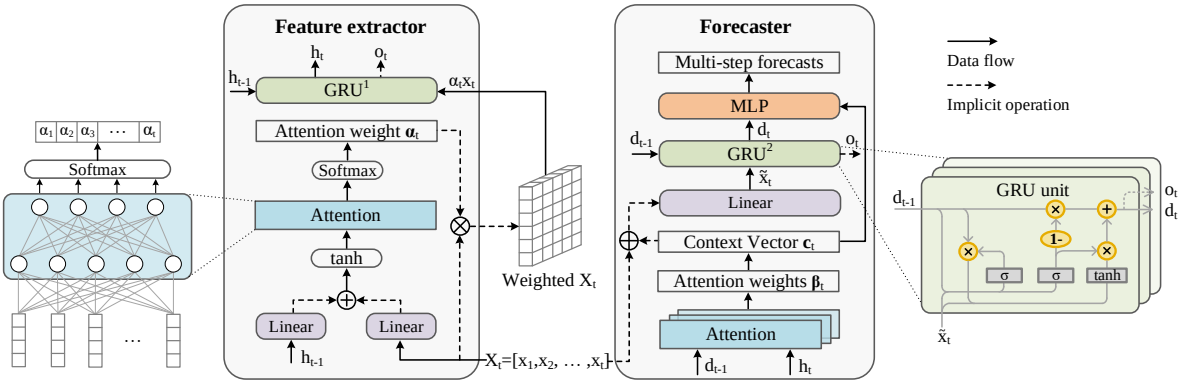

Figure 5: The forecasting model architecture based on attention layers and GRU network.

where $\boldsymbol{\alpha}_t = [\alpha_t^1, \alpha_t^2, \dots, \alpha_t^m] \in \mathbb{R}^m$ is a vector of attention weights representing the probability distribution over the feature space and the sum of all the weights $\sum_{j=1}^{m} \alpha_t^j = 1$.

By performing the above operations at each time step, we can obtain the weighted input sequence by multiplying the spatial attention weights with corresponding input data and represent the attended input sequence as:

$$\mathbf{X}'_s = \begin{bmatrix} \mathbf{x}'_1 \\ \mathbf{x}'_2 \\ \vdots \\ \mathbf{x}'_s \end{bmatrix} = \begin{bmatrix} \alpha_1^1 x_1^1 & \alpha_1^2 x_1^2 & \cdots & \alpha_1^m x_1^m \\ \alpha_2^1 x_2^1 & \alpha_2^2 x_2^2 & \cdots & \alpha_2^m x_2^m \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_s^1 x_s^1 & \alpha_s^2 x_s^2 & \cdots & \alpha_s^m x_s^m \end{bmatrix} \tag{7}$$

Subsequently, we use a GRU layer to encode the time dependencies of the weighted input sequences. The GRU layer contains multiple GRU units, each of which processes a data point at a one-time step by referring to the memory from its previous hidden states $\boldsymbol{h}_{t-1}$ [48]. The update gate $\mathrm{z}(\cdot)$ and the reset gate $\mathrm{r}(\cdot)$ then decide which information to be passed or forgotten. The currently hidden state vector is computed as follows:

$$\mathbf{z}_t = \sigma(\boldsymbol{W}_z \mathbf{x}'_t + \boldsymbol{U}_z \boldsymbol{h}_{t-1}) \tag{8}$$

$$\mathbf{r}_t = \sigma(\boldsymbol{W}_r \mathbf{x}'_t + \boldsymbol{U}_r \boldsymbol{h}_{t-1}) \tag{9}$$

$$\boldsymbol{h}'_t = \tanh(\boldsymbol{W} \mathbf{x}'_t + \mathbf{r}_t \odot \boldsymbol{h}_{t-1}) \tag{10}$$

$$\boldsymbol{h}_t = \mathbf{z}_t \odot \boldsymbol{h}_{t-1} + (1 - \mathbf{z}_t) \odot \boldsymbol{h}'_t \tag{11}$$

where $\sigma(\cdot)$ is sigmoid activation function and $\odot$ is element-wise Hadamard product and $\boldsymbol{W}_z$, $\boldsymbol{U}_z$, $\boldsymbol{W}_r$ and $\boldsymbol{U}_r$ are GRU parameters. The current hidden state $\boldsymbol{h}_t \in \mathbb{R}^p$ contains all the encoded information of the weighted input data $\mathbf{x}'_t$ and will be forwarded to the decoder for further processing.

4.1.2. Forecaster

The structure of the forecaster is similar to the encoder, except for the additional linear transformation and a multi-layer perceptron (MLP) layer to make multi-step forecasts. To further enhance the forecasting performance, we use temporal-attention layers to capture temporal dynamics of the encoder hidden state $\boldsymbol{h}_t$ based on the previous decoder hidden state $\boldsymbol{d}_{t-1}$. The temporal attention weights are calculated by the following equations:

$$\boldsymbol{\vartheta}_t = \boldsymbol{v}_a'^{\top} \tanh\left(\boldsymbol{W}'_a * \boldsymbol{d}_{t-1} + \boldsymbol{U}'_a * \boldsymbol{H}_t\right) \tag{12}$$

$$\boldsymbol{\beta}_t = \mathrm{softmax}(\boldsymbol{\vartheta}_t) = \frac{\exp\left(\boldsymbol{\vartheta}_t\right)}{\sum_{j=1}^{s} \exp\left(\vartheta_t^j\right)} \tag{13}$$

where $\boldsymbol{W}'_a \in \mathbb{R}^{p \times q}$, $\boldsymbol{U}'_a \in \mathbb{R}^{p \times p}$ and $\boldsymbol{v}_a'^{\top} \in \mathbb{R}^q$ are parameters of the alignment function to learn, and q is the size of the decoder hidden vector. $\boldsymbol{\beta}_t = [\beta_{t-s}, \beta_{t-s+1}, \dots, \beta_t] \in \mathbb{R}^s$ is temporal attention weight vector that scores the importance of each hidden state in $\boldsymbol{H}_t = [\boldsymbol{h}_{t-s}, \boldsymbol{h}_{t-s+1}, \dots, \boldsymbol{h}_t]^{\top}$ corresponding to the attended input sequence $\mathbf{X}'_t$. Afterward, the attention layer computes a context vector as a weighted sum of all encoder hidden states as represented below:

$$\boldsymbol{c}_t = \boldsymbol{\beta}_t \mathbf{H}_t = \sum_{i=t-s}^{t} \beta_i \boldsymbol{h}_i \tag{14}$$

where $\boldsymbol{c}_t \in \mathbb{R}^q$ is the context vector, which is then concatenated with input data point $\mathbf{x}_t \in \mathbb{R}^m$. The concatenated vector $[\boldsymbol{c}_t; \mathbf{x}_t] \in \mathbb{R}^{q+m}$ is transformed by a linear layer:

$$\tilde{\mathbf{x}}_t = \mathbf{w}^{\top}(\boldsymbol{c}_t \oplus \mathbf{x}_t) + b \tag{15}$$

where $\oplus$ is concatenation operation and $\tilde{\mathbf{x}}_t \in \mathbb{R}^{q+s}$ is the newly transformed input data based on which the current hidden state $\boldsymbol{d}_t \in \mathbb{R}^q$ of the decoder GRU layer is produced.

Finally, the MLP layer takes the final decoder hidden state $\boldsymbol{d}_t$ and the context vector $\boldsymbol{c}_t$ as inputs, and summarize all the information to make forecasts by:

$$\hat{\mathbf{y}}_{t+\tau} = \hat{\mathbf{w}}^{\top}(\boldsymbol{c}_t \oplus \boldsymbol{d}_t) + \hat{b} \tag{16}$$

where $\hat{\mathbf{w}}$ and $\hat{b}$ are learnable parameters of the MLP layer and $\hat{\mathbf{y}}_{t+\tau} = [y_{t+1}, y_{t+2}, \dots, y_{t+k}] \in \mathbb{R}^k$ are the predictions

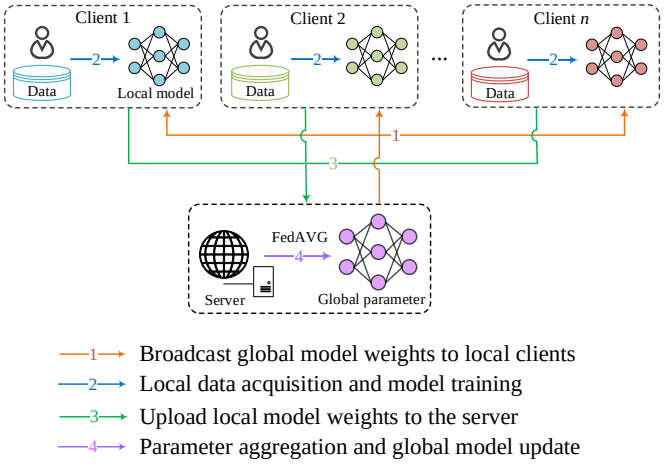

Figure 6: A schematic view of a typical FL approach. The illustrated is one communication round between local clients and a central server.

of the forecasting model. The final forecasting results $\widehat{\mathbf{Y}}$ for all forecast horizons at $\{5, 10, \ldots, 30\}$-min are:

$$\widehat{\mathbf{Y}} = \begin{array}{c} \begin{matrix} \text{5-min} & \text{10-min} & \cdots & \text{30-min} \\ \downarrow & \downarrow & & \downarrow \end{matrix} \\ \begin{bmatrix} \hat{y}_{t+1} & \hat{y}_{t+2} & \cdots & \hat{y}_{t+6} \\ \hat{y}_{t+2} & \hat{y}_{t+3} & \cdots & \hat{y}_{t+7} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{y}_{n+1} & \hat{y}_{n+2} & \cdots & \hat{y}_{n+6} \end{bmatrix} \end{array} \begin{array}{l} \leftarrow \mathbf{X}_t \\ \leftarrow \mathbf{X}_{t+1} \\ \vdots \\ \leftarrow \mathbf{X}_n \end{array}$$

After prediction, the predicted CSI are converted back to GHI by multiplying the corresponding clear-sky irradiance as a inverse process in (1) for further evaluation.

4.2. Distributed solar forecasting framework using FL

Based on the abovementioned theory, we propose a distributed solar forecasting framework that incorporates the STANN model with federated learning technique (namely FL-STANN) to further improve forecasting accuracy and generalization of various data.

4.2.1. General FL setting

In conventional FL settings, the learning task is to optimize a *global* model parameter Ψ_S via a group of distributed datasets under the coordination of a central server S as shown in Fig. 6. We can regard the group of datasets as a federation of N clients denoted as $\mathbb{C} = \{C_i\}|_{i=1}^{N}$, each holding a local dataset $\mathbb{D}_i$. Nevertheless, the clients are prohibited from uploading the data to their central server due to privacy concerns, but the central server can access the local network parameters. Therefore, the global parameter is exclusively optimized on each local dataset with a loss function $\ell(\mathbb{D}_i; \psi_i)$, and then aggregated on the central server. The local optimization is defined as:

$$\min_{\tilde{\psi}_i} \ell(\mathbb{D}_i; \psi_i) \overset{\text{def}}{=} \frac{1}{n_i} \sum\nolimits_{t \in n_i} \ell(\mathbb{D}_i; \psi_i) \tag{17}$$

where $\tilde{\psi}_i$ is the updated parameters of client C_i trained on dataset $\mathbb{D}_i$. It is obtained by minimizing average loss over all n_i data samples in the dataset.

After training, the updated parameters are uploaded to the central server and aggregated using FedAvg algorithm as follows:

$$\tilde{\Psi}_S = \Psi_S + \frac{1}{N} \sum_{i=1}^{N} \Delta\psi_i \tag{18}$$

where $\tilde{\Psi}$ is aggregated global parameter, Ψ is global model parameter from previous updating round and n_i is length of the data. $\Delta\psi_i = \tilde{\psi}_i - \psi_i$ is the difference between parameters from current round and previous round of client C_i.

4.2.2. FL-STANN framework

In solar forecasting applications, learning objective is to train multiple *local* models for each client C_i to make predictions based on their datasets $\mathbb{D}_i \sim P_i$. Given an input sequence $\mathbf{X}_t \in \mathbb{R}^{s \times m}$ from dataset $\mathbb{D}_i$, the feature extractor weights $\psi_i^{(\text{enc})}$ and the forecaster weights $\psi_i^{(\text{dec})}$ can be separately optimized by minimizing MSE loss using backward propagation algorithm with two AdamW optimizers and a MSE loss function $\ell(\cdot)$ on each dataset. The optimization of both extractor and forecaster can be formulated as followings:

$$\tilde{\psi}^{(\text{enc})} \leftarrow \arg\min_{\psi^{(\text{enc})}} \sum_{t=1}^{s} \sum_{i=1}^{m} \ell\left(\mathbf{x}_t^i, \psi_i^{(\text{enc})}\right) \tag{19}$$

$$\tilde{\psi}^{(\text{dec})} \leftarrow \arg\min_{\psi^{(\text{dec})}} \sum_{t=1}^{s} \sum_{j=1}^{q} \ell\left(\boldsymbol{h}_t^j, \psi_j^{(\text{dec})}\right) \tag{20}$$

where the encoder weight is training on dataset $\mathbb{D}_i$ to follow its data distribution $\psi_i^{(\text{enc})} \sim P|_{\mathbb{D}_i}$ in (19). The encoded the hidden state vector $\boldsymbol{h}_t$ is further used as the input together with the previous decoder hidden states $\boldsymbol{d}_{t-1}$ to make multi-horizon predictions $\hat{\mathbf{y}}_{t+\tau}$ in (20). The localized optimization procedure is described in the following steps:

(i) Collecting data and making datasets $\mathbb{D}_i$ before training. This step is only executed once before local training;

(ii) Updating decoder parameters $\psi_i^{(\text{dec})}|_{S \to C_i}$ from the globally aggregated updates;

(iii) Training both encoder and decoder based on the local datasets $\mathbb{D}_i$ for a certain number of local iterations;

(iv) Uploading the updated the decoder parameters $\Delta\psi_i^{(\text{dec})}|_{C_i \to S}$ to the central server S, in the meanwhile keeping the optimized encoder weights $\tilde{\psi}_i^{(\text{enc})}$ locally.

Table 2: Implementation details of FL-STANN model.

Parameters	No. of layers	Hidden layer size	Sequence length	Batch size	Learning rate	No. of clients	Global epochs	Local epochs	Loss function	Optimizer
Values	1	$p = q = 128$	$s = 20$	$bs = 64$	$\phi = 1e{-}3$	$N = 8$	$T = 5$	$e = 10$	MSE	AdamW

On the central server, a global decoder $\Psi_S^{(\text{dec})}$ is constructed by aggregating the updated parameters from local clients. Nonetheless, this will lead to the asynchronous issue of updates, for example, each client's update is immediately applied to the local decoder before any aggregation with updates from other clients as discussed in [49]. Since there is no actual communication between clients and the server in our simulations, we assume that all clients upload and aggregate the parameters in a synchronized way as follows:

(i) Initializing decoder parameters at the beginning of the FL updating round;

(ii) The server selects a set of clients meeting the eligibility requirement (e.g., the gradient is converged or training loss is continuously decreasing);

(iii) Performing FedAvg algorithm to aggregate parameters of the selected clients:

$$\tilde{\Psi}_S^{(\text{dec})} = \Psi_S^{(\text{dec})} + \frac{1}{\eta \times N} \sum_{i=1}^{\eta \times N} \Delta\psi_i^{(\text{dec})} \tag{21}$$

(iv) Broadcasting the aggregated parameter to the selected clients.

where $\eta \in [0, 1]$ is ratio of the clients participated in each FL round. However, the case of $\eta = 0$ is very unlikely, especially in cross-device FL applications that involve large number of clients. The overall procedure of the proposed FL-STANN approach is described in Algorithm 1.

Algorithm 1 FL-STANN

input: Initial local encoder $\psi_i^{(\text{enc})}$, global decoder Ψ_S, optimizer $\mathcal{O}(\cdot)$, loss function $\ell(\cdot)$, and learning rate ϕ.
output: Updated local encoder $\tilde{\psi}_i^{(\text{enc})}$ and decoder $\tilde{\psi}_i^{(\text{dec})}$.

init: clients $\mathbb{C} = \{C_i\}|_{i=1}^N$ holds local encoder parameter $\psi_i^{(\text{enc})}$ and initialize the decoder parameter $\psi_i^{(\text{dec})} \leftarrow \Psi_S$, and a local dataset $\mathbb{D}_i$.
for each round $r = 1, 2, \ldots; r \in T$ **do**
  Client **executes**:
  for each client $C_i \in \mathbb{C}$ ***in parallel*** **do**
    $\cdot\ \tilde{\psi}_i^{(\text{enc})}, \tilde{\psi}_i^{(\text{dec})} \leftarrow \text{Training}(\mathbb{D}_i, \phi, \psi_i^{(\text{enc})}, \psi_i^{(\text{dec})})$
    $\cdot$ update local encoder $\psi_i^{(\text{enc})} \leftarrow \tilde{\psi}_i^{(\text{enc})}$
    if $\ell^{(r)} - \ell^{(r-1)} < 0$ **then**
      $\cdot\ \Delta\psi_i^{(\text{dec})} = \tilde{\psi}_i^{(\text{dec})} - \psi_i^{(\text{dec})}$
      $\cdot\ \text{upload}_{C_i \to S}(\Delta\psi_i^{(\text{dec})})$
    else
      $\cdot$ keep the optimal decoder weights $\psi_i^{(\text{dec})}$
    end if
  end for
  Server **executes**:
  $\cdot$ select $\eta \times N$ clients $C_i \subseteq \mathbb{C}$
  $\cdot\ \text{aggregate } \tilde{\Psi}_S \leftarrow \text{FedAvg}(\Delta\psi_i^{(\text{dec})})$
  $\cdot\ \text{broadcast}_{S \to C_i}(\tilde{\Psi}_S \to \psi_i^{(\text{dec})})$
end for
Return $\tilde{\psi}_i^{(\text{enc})}$, $\tilde{\psi}_i^{(\text{dec})}$

5. Experiments

To fully assess the performance of the proposed approach, we carry out extensive experiments on real-world solar datasets. First, we compare the multi-horizon forecasting performance of the STANN model with different benchmarks on various input features with the Folsom dataset. Then, we verify the effectiveness of the FL-STANN training strategy on different NREL datasets. Finally, we demonstrate the performance of FL-STANN at different participation rates.

5.1. Experiment setup and statistics

In this work, all the simulations are implemented with *Python* 3.8 and *PyTorch* deep learning library and trained on a server using an NVIDIA 1080Ti GPU. For the parameters of the STANN (i.e., p, q, and s), they are determined by conducting grid search on $p = q \in \{32, 64, 128, 256, 512\}$ and $s \in \{5, 10, 15, 20, 25, 30\}$. For example, we initially set $p = q = 32$ and conducted training on the 2014-2015 data of the Folsom dataset and tested on 2016 data. Then, we repeated the above procedure by using various hidden layer sizes {64, 128, 256, 512}. We finally choose $p = q = 128$ and $s = 20$ that achieve the best performance considering the trade-off between accuracy and computation cost. The hyperparameters of the framework are shown in Table 2.

Besides, we consider three commonly used error metrics for time series prediction, namely, mean absolute error (MAE), mean bias error (MBE), and root mean square error (RMSE). The normalized versions are used to compare the performance across different dataset, which are defined as: $\text{nMAE} = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t|/\bar{y}$, $\text{nMBE} = \frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)/\bar{y}$ and $\text{nRMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}/\bar{y}$ where $\bar{y}$ is the mean of target values. Finally, we use forecast skill to measure the improvement of a forecasting model over the reference model:

$$\text{skill} = 1 - \frac{\text{nRMSE}}{\text{nRMSE}_\text{p}} \tag{22}$$

where nRMSE_p is the nRMSE of the smart persistence.

Table 3: Multi-horizon forecasting results of the STANN and the benchmarks. The numbers in the parenthesis indicate the changes caused by introducing the endogenous and the exogenous features compared with clear-sky index.

		5-min	**10-min**	**15-min**	**20-min**	**25-min**	**30-min**
		CSI(+Endo.)(+Exo.)	CSI(+Endo.)(+Exo.)	CSI(+Endo.)(+Exo.)	CSI(+Endo.)(+Exo.)	CSI(+Endo.)(+Exo.)	CSI(+Endo.)(+Exo.)
nMAE [%]	Baseline	5.65	8.74	11.12	13.33	15.35	17.30
	Lasso	6.24(-0.01)(+1.36)	8.45(+0.01)(+1.37)	10.83(-0.05)(+1.83)	12.84(-0.01)(+1.89)	14.24(-0.01)(+2.76)	16.85(-0.04)(+3.01)
	ElasticNet	6.25(-0.01)(+1.29)	8.44(-0.02)(+1.35)	10.81(-0.03)(+1.77)	12.81(-0.01)(+1.84)	14.22(-0.02)(+2.74)	16.83(-0.03)(+2.98)
	GRU	5.52(+0.03)(+1.32)	7.64(+0.08)(+2.59)	9.38(+0.07)(+2.64)	10.95(+0.16)(+3.19)	11.22(-0.05)(+2.63)	13.34(+0.11)(+2.38)
	LSTMT	5.31(+0.02)(+1.48)	7.41(+0.10)(+2.45)	9.17(+0.11)(+2.29)	10.82(-0.05)(+2.74)	10.95(+0.07)(+3.19)	12.75(+0.13)(+2.42)
	A-LSTM	4.92(-0.02)(+1.22)	6.91(-0.09)(+1.28)	8.77(+0.06)(+1.79)	8.61(+0.18)(+2.41)	10.12(-0.08)(+2.71)	10.83(-0.05)(+2.54)
	Transformer	**4.68**(-0.14)(+0.92)	6.53(-0.15)(+0.95)	8.68(+0.03)(+1.21)	**7.62**(-0.31)(+1.37)	9.86(-0.15)(+1.58)	10.61(-0.04)(+1.87)
	STANN	4.73(**-0.11**)(**+0.84**)	**6.51**(**-0.17**)(**+0.82**)	**8.47**(**-0.18**)(**+0.93**)	7.91(**-0.23**)(**+1.29**)	**9.32**(**-0.27**)(**+1.42**)	**10.03**(**-0.14**)(**+1.62**)
nMBE [%]	Baseline	0.0	0.0	0.01	0.0	0.01	0.02
	Lasso	1.18(-0.01)(+1.42)	1.33(-0.01)(+1.78)	1.35(+0.02)(+1.79)	1.44(-0.01)(+2.02)	1.58(-0.02)(+2.31)	1.89(-0.01)(+2.55)
	ElasticNet	1.16(-0.00)(+1.40)	1.31(-0.01)(+1.79)	1.33(-0.01)(+1.76)	1.43(-0.00)(+1.98)	1.59(-0.03)(+2.26)	1.88(+0.01)(+2.53)
	GRU	2.92(-0.03)(+4.21)	3.41(-0.02)(+5.42)	2.32(+0.01)(+8.74)	2.24(-0.05)(+6.51)	2.79(-0.02)(+5.18)	3.70(-0.03)(+4.86)
	LSTM	3.08(-0.12)(+3.75)	3.42(-0.05)(+4.82)	2.43(-0.04)(+4.13)	2.11(-0.03)(+3.81)	2.76(-0.03)(+4.32)	3.44(-0.03)(+3.41)
	A-LSTM	2.54(-0.09)(+2.25)	2.61(-0.12)(+2.31)	**-1.19**(+0.08)(-0.21)	1.67(-0.04)(+2.47)	-3.15(+0.32)(-1.25)	2.91(-0.58)(+3.21)
	Transformer	-1.92(+0.41)(-1.42)	1.84(-0.42)(+2.26)	-1.91(+0.42)(-1.30)	-1.60(+0.63)(-1.85)	**1.27**(-0.11)(+1.85)	**1.25**(**-1.02**)(+2.33)
	STANN	**-0.63**(**-0.31**)(**-1.24**)	1.48(**-0.91**)(**+1.42**)	1.45(**-0.47**)(**+1.27**)	**1.19**(**-1.02**)(**+1.84**)	-1.76(**+0.43**)(**-1.37**)	1.49(-0.83)(**+1.59**)
nRMSE [%]	Baseline	11.84	15.62	17.83	19.88	21.73	23.48
	Lasso	11.72(-0.01)(+0.32)	14.92(-0.02)(+0.44)	16.62(-0.02)(+0.84)	17.90(-0.02)(+1.02)	18.91(-0.02)(+1.33)	19.86(-0.03)(+2.01)
	ElasticNet	11.70(-0.03)(+0.31)	14.87(-0.01)(+0.42)	16.59(-0.03)(+0.85)	17.86(-0.01)(+0.99)	18.87(-0.02)(+1.37)	19.83(-0.02)(+2.03)
	GRU	11.38(-0.03)(+0.53)	14.11(-0.02)(+0.68)	15.92(-0.05)(+0.94)	17.51(-0.05)(+1.36)	18.09(-0.11)(+1.94)	19.26(-0.10)(+2.24)
	LSTM	11.43(-0.02)(+0.49)	14.25(+0.01)(+0.53)	16.04(-0.04)(+1.02)	17.26(-0.03)(+1.24)	17.94(-0.03)(+2.15)	19.17(-0.05)(+1.79)
	A-LSTM	11.31(-0.05)(+0.34)	13.96(-0.05)(+0.41)	15.32(-0.08)(+0.83)	16.48(-0.06)(+1.45)	17.73(-0.09)(+1.92)	18.89(-0.07)(+1.94)
	Transformer	11.14(-0.04)(+0.27)	13.71(-0.07)(+0.25)	15.01(-0.12)(+0.68)	15.92(-0.12)(+1.13)	17.29(-0.21)(+1.36)	17.65(-0.13)(+1.46)
	STANN	**10.94**(**-0.06**)(**+0.24**)	**13.04**(**-0.08**)(**+0.22**)	**14.72**(**-0.15**)(**+0.47**)	**15.39**(**-0.17**)(**+0.97**)	**16.72**(**-0.19**)(**+1.12**)	**16.87**(**-0.16**)(**+1.35**)
Skill [%]	Baseline	0.0	0.0	0.0	0.0	0.0	0.0
	Lasso	1.01(+0.08)(-2.87)	4.48(+0.13)(-2.81)	6.79(+1.12)(-4.71)	9.96(+1.01)(-5.13)	12.98(+0.92)(-6.12)	15.41(+0.12)(-8.56)
	ElasticNet	1.18(+0.25)(-2.70)	4.80(+0.06)(-2.69)	6.95(+1.68)(-4.76)	10.16(+0.05)(-5.48)	13.16(+0.92)(-6.30)	15.54(+0.08)(-8.64)
	GRU	3.89(+0.25)(-4.48)	7.54(+0.13)(-4.46)	10.71(+0.28)(-5.27)	11.92(+0.25)(-6.84)	16.75(+0.51)(-8.93)	17.97(+0.43)(-9.54)
	LSTM	3.46(+0.17)(-4.14)	6.62(-0.07)(-3.47)	10.04(+0.22)(-5.71)	13.18(+0.15)(-6.24)	17.44(+0.14)(-7.59)	18.36(+0.21)(-7.63)
	A-LSTM	4.48(+0.33)(-2.88)	10.63(+0.32)(-2.63)	14.08(+0.45)(-4.66)	17.10(+0.30)(-7.29)	18.41(+0.41)(-8.84)	19.55(+0.17)(-8.26)
	Transformer	5.91(+0.34)(-2.28)	12.23(+0.45)(-1.61)	15.82(+0.67)(-3.82)	19.92(+0.61)(-5.68)	20.43(+0.97)(-6.26)	24.83(+0.55)(-6.30)
	STANN	**7.76**(**+0.51**)(**-2.03**)	**16.52**(**+0.51**)(**-1.41**)	**17.44**(**+0.84**)(**-2.63**)	**22.59**(**+0.85**)(**-4.88**)	**23.06**(**+0.87**)(**-5.16**)	**28.15**(**+0.68**)(**-5.75**)

Table 4: Comparison of inference time of each model [sec.].

	Lasso	ElasticNet	GRU	LSTM	A-LSTM	Transformer	STANN
CSI	2.6	2.7	10.2	13.6	16.8	21.9	17.8
CSI+Endo.	4.8	5.1	17.5	21.9	29.3	45.7	40.6
CSI+Exo.	8.1	8.9	31.7	36.9	40.4	72.5	64.1

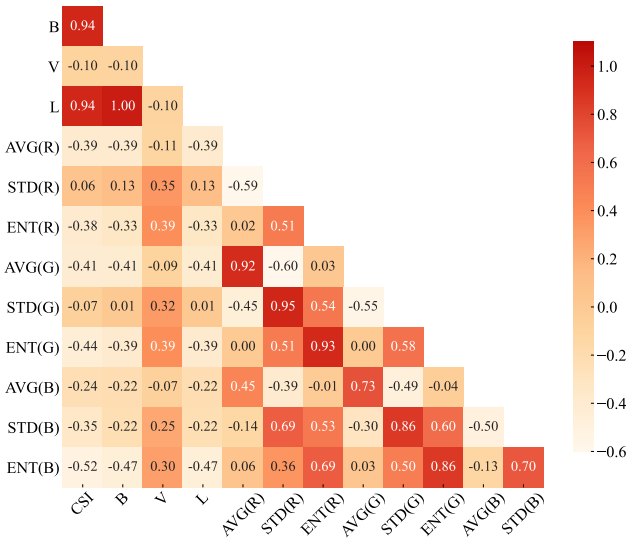

Figure 7: Correlation matrix of different input features.

5.2. Multi-horizon forecasting performance

The multi-horizon forecasting results are presented in Table 3 from 5-min to 30-min ahead. In this experiment, we compare the STANN model compared with some commonly used time series forecasting models, including Lasso regression, ElasticNet regularized regression, GRU, LSTM, attention-based LSTM (A-LSTM) [50] and a deep Transformer model for time series forecasting [51]. The model parameters are the same as those in the original papers. For the Lasso and ElasticNet, the multitask versions in *scikit-learn* machine learning library are used to generate multi-step predictions.

To have better intuition about the results, predictions of four typical weather types are visualized in Fig. 8, which are overcast, cloudy, partially cloudy, and sunny, respectively. Since the predictions from the ElasticNet is very close to that of the Lasso model, its results are omitted for a more clear presentation. According to the figure, it is obvious that the forecasts of the STANN model is closer to the actual values under all weathers. In contrast, the other models suffer from deteriorated forecasting accuracy more or less during large ramp events.

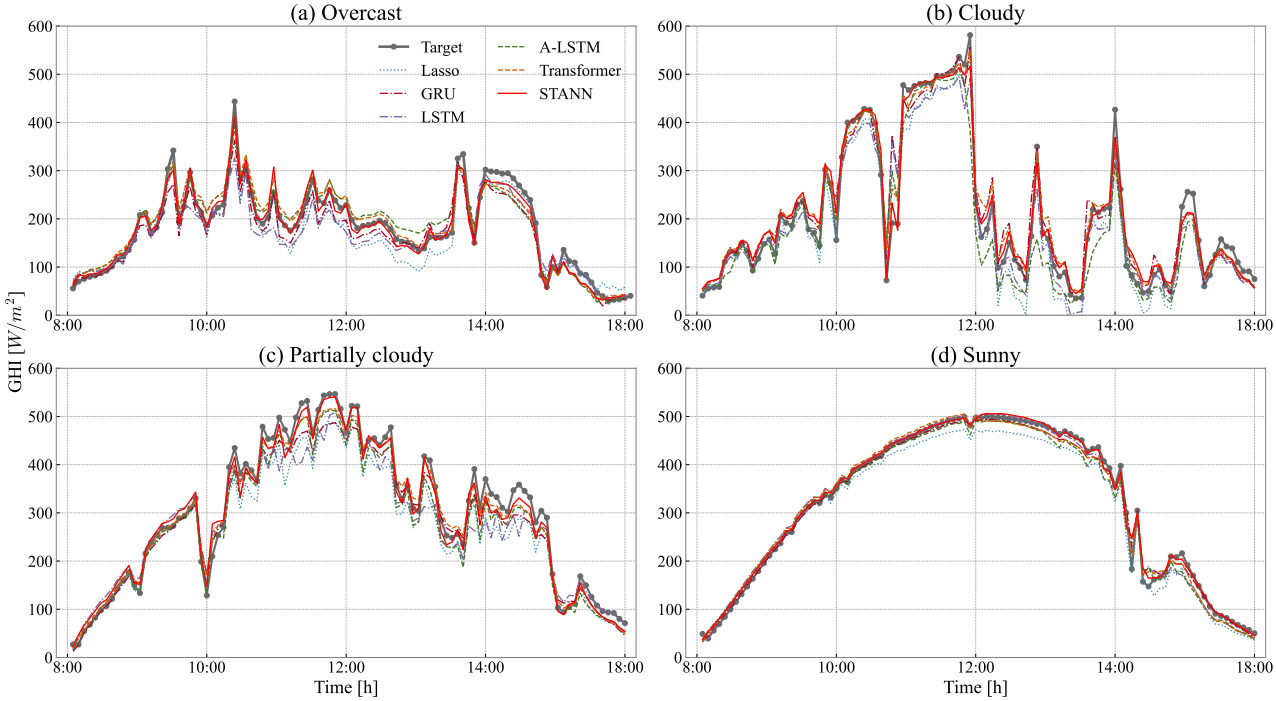

Figure 8: Visualized prediction curves of GHI at 5-min ahead under different weathers: (a) overcast, (b) cloudy, (c) partially cloudy, and (d) sunny days.

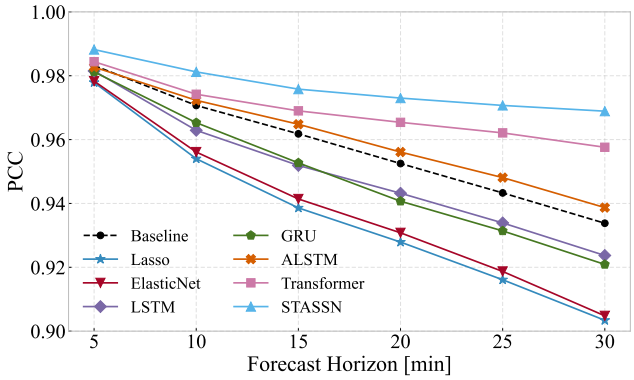

Figure 9: PCC between predictions and measurements at each forecast horizon.

Moreover, we also investigate the model performance by using three input combinations: (1) CSI only, (2) CSI with endogenous features, and (3) CSI with exogenous features.Besides the endogenous features in Section 3.2.2, exogenous variables (i.e., average, standard deviation, and entropy derived for each red, green, and blue channels of selected sky images) are also included. Although the negative impact of using exogenous features has been reported in [42], we still use them as noisy signals to test the robustness of the forecasting models in this simulation. To better compare model performance on different input features, the results directly present improvement or decrement compared to the CSI forecasting results, for example, the numbers in the parenthesis are the corresponding changes resulting from the endogenous and exogenous features.

According to the results, the regression approaches, i.e., the Lasso and the ElasticNet regressions have the worse performance in forecasting errors and skills, which indicates the incapability of dealing with complex nonlinearity. As for the DL-based neural networks, i.e., GRU, and LSTM, outperform the regression approaches concerning nMAE, nRMSE, and skill. However, it is worth mentioning that the predictions of GRU and LSTM models are generated by using the n-step ahead forecasting approach since they failed to generate comparable predictions in a joint way. As for the attention-based approaches, i.e., A-LSTM, Transformer, and STANN perform much better than the vanilla LSTM and GRU in terms of all metrics. Particularly, the proposed STANN model achieves the best results on nRMSE and forecast skills for all forecast horizons, which are respectively 7.76%, 16.52%, 17.44%, 22.59%, 23.06%, and 28.15%. To evaluate the computational efficiency, we compare runtime for the model inference. The results are presented in Table 4. It is seen that as the model complexity increases, the time used for testing increases as well. The Lasso and ElasticNet only take a few seconds for three input combinations. However, the transformer has the longest runtime during inference and the runtime for STANN is slightly shorter.

Fig. 7 shows the correlations between different input features. By using the three endogenous features, the forecasting performance of each model can be further improved to different extents due to the strong correlation between

Table 5: The forecasting results generated by the STANN trained with different training strategies.

Site	Input	Model	nMAE [%]	nMBE [%]	nRMSE [%]	Skill [%]	Runtime [h]
HI	CSI	Persistence	29.1±7.1	0±0	45.4±8.1	-	-
		Localized	26.7±5.0	2.9±1.2	39.0±5.2	14.1±1.2	0.048
		Centralized	25.4±4.8	-1.5±1.3	37.6±4.2	17.2±1.4	0.231
		FL-Both*	25.2±4.6	-1.9±1.1	37.4±4.3	17.6±1.4	0.127
		FL-STANN	**23.8±4.2**	**1.3±0.8**	**35.6±4.5**	**21.6±1.5**	0.108
CA	CSI $+\mathcal{B}$	Persistence	15.3±5.0	0±0	23.7±5.9	-	-
		Localized	12.5±2.9	**-0.5±1.0**	19.9±3.2	16.0±2.7	0.070
		Centralized	14.2±2.8	-1.9±0.9	20.2±3.2	14.8±2.5	0.506
		FL-Both*	12.7±2.5	-1.0±1.1	19.2±3.3	19.0±2.1	0.127
		FL-STANN	**11.9±2.4**	-1.2±0.5	**19.1±3.3**	**19.4±2.2**	0.146
CO	CSI $+\mathcal{BV}$	Persistence	13.5±4.5	-0.1±0	20.9±4.8	-	-
		Localized	11.4±2.7	-6.6±0.4	18.5±3.0	11.5±2.5	0.116
		Centralized	10.7±2.4	-4.2±0.5	18.1±2.7	13.4±1.8	0.958
		FL-Both*	10.9±1.9	-2.9±0.8	18.3±2.3	12.4±1.2	0.127
		FL-STANN	**9.8±1.2**	**-0.3±0.6**	**17.2±2.6**	**17.7±1.6**	0.206
OR	CSI $+\mathcal{BVL}$	Persistence	16.8±3.2	-0.1±0	27.1±6.2	-	-
		Localized	14.8±3.2	1.2±0.8	23.5±4.2	13.3±1.8	0.164
		Centralized	15.4±2.8	-3.2±1.3	24.2±2.2	10.7±1.1	1.417
		FL-Both*	14.6±2.5	-1.8±1.4	24.0±1.8	11.4±1.2	0.127
		FL-STANN	**14.5±3.2**	**-0.7±0.6**	**23.2±4.1**	**14.4±1.6**	0.367

Site	Input	Model	nMAE [%]	nMBE [%]	nRMSE [%]	Skill [%]	Runtime [h]
NV	CSI	Persistence	9.6±3.6	0±0	13.8±3.7	-	-
		Localized	5.8±0.3	2.7±1.4	10.6±0.8	23.2±3.9	0.048
		Centralized	6.2±0.4	-1.9±1.0	10.8±0.7	21.7±4.1	0.231
		FL-Both*	6.7±1.2	-5.4±1.2	11.3±1.4	18.1±3.8	0.127
		FL-STANN	**5.3±1.2**	**-1.3±0.6**	**10.1±1.2**	**26.8±2.6**	0.108
ARI	CSI $+\mathcal{B}$	Persistence	10.8±3.8	0±0	15.8±3.9	-	
		Localized	**6.1±0.6**	**0.1±1.7**	12.0±1.6	24.1±2.2	0.071
		Centralized	8.6±0.8	-2.4±1.1	12.1±1.7	23.4±2.0	0.506
		FL-Both*	7.8±1.4	-3.5±0.7	12.7±1.9	19.6±2.7	0.127
		FL-STANN	7.2±1.3	-0.8±1.2	**11.7±1.7**	**25.8±2.1**	0.146
NC	CSI $+\mathcal{BV}$	Persistence	19.5±4.6	0±0	31.3±4.5	-	
		Localized	19.1±1.7	8.9±0.6	29.4±2.2	6.1±2.4	0.116
		Centralized	17.3±1.6	-3.9±1.1	27.8±1.7	11.2±2.1	0.958
		FL-Both*	16.7±1.9	-1.8±1.5	26.4±2.7	15.6±1.1	0.127
		FL-STANN	**16.0±2.6**	**-0.7±1.0**	**25.7±2.5**	**17.3±1.3**	0.206
KS	CSI $+\mathcal{BVL}$	Persistence	15.6±4.6	0±0	24.2±4.8	-	-
		Localized	13.8±3.6	-6.0±1.1	21.5±3.6	11.2±1.2	0.163
		Centralized	13.2±2.4	-2.7±1.4	20.7±2.7	14.5±1.3	1.417
		FL-Both*	14.4±3.2	-4.3±0.6	21.1±2.6	12.8±1.4	0.127
		FL-STANN	**12.8±2.6**	**-1.5±0.4**	**19.7±2.9**	**18.6±1.5**	0.367

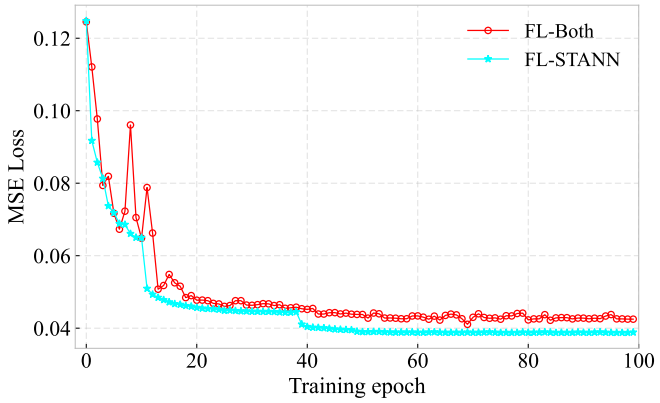

Figure 10: Training losses of FL-STANN vs. conventional FL setting.

CSI and endogenous variables $\mathcal{B}$ and $\mathcal{L}$. Generally, the results in the middle parenthesis show that the nMAEs and nRMSE are reduced and the nMBEs are more towards 0, which results in more accurate predictions and improved forecasting skills. Particularly, the STANN model has the most significant improvements than the others, for example, forecast skill at 5-min ahead is increased by 0.51% higher than the 0.34% of the Transformer model. In contrast, the forecasting performances deteriorated by introducing the 9 exogenous features (see in Section II. B) because of the weak correlations with the CSI as well as other endogenous features. Notably, the GRU and LSTM models are most vulnerable to the exogenous features in terms of all metrics. A worse case can be observed where the forecast skill of the GRU has decreased by 9.54% at 30-min ahead. Nevertheless, the deterioration in STANN is much smaller than the others, which suggests that the STANN model is more robust to unexpected input features.

We also compute the Pearson correlation coefficient (PCC) for each horizon below:

$$\gamma = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})}}$$

which reflects the linear correlation between the predictions and the targets. A larger value indicates a stronger relationship between two variables. Fig. 9 shows the variations of the PCC of each forecasting model as the horizon increases. As we can see, STANN has the highest PCCs and the trend of decreasing is slight as the horizon increases, which demonstrates a strong correlation between the predictions and the target for all forecast horizons. In contrast, the PCCs of the Lasso, ElasticNet, GRU, and LSTM decrease fast and are less than that of the baseline model. In the meanwhile, the A-LSTM and the Transformer have intermediate performance between the baseline and STANN. The results demonstrate the better capability of the STANN in multi-horizon solar forecasting.

5.3. FL forecasting performance

In this section, we compare four training methods using NREL datasets: STANN trained with data gathered from all local datasets (centralized), STANN exclusively trained with each local dataset (localized), and conventional FL approach where both feature extractor and forecaster are updated through central server (FL-Both), and the proposed FL-STANN. Each local dataset is divided into 70% of the training set and 30% of the out-of-sample test set. Considering the availability of exogenous data at each site, we only use endogenous features generated without other exogenous features.

We firstly carry out a preliminary experiment in the FL-STANN framework and in a conventional FL setting using CSI without other features. The models are trained for 100 FL rounds and their training loss curves are shown in Fig 10. It can be observed that the losses decrease rapidly at the beginning, and eventually approach a steady state

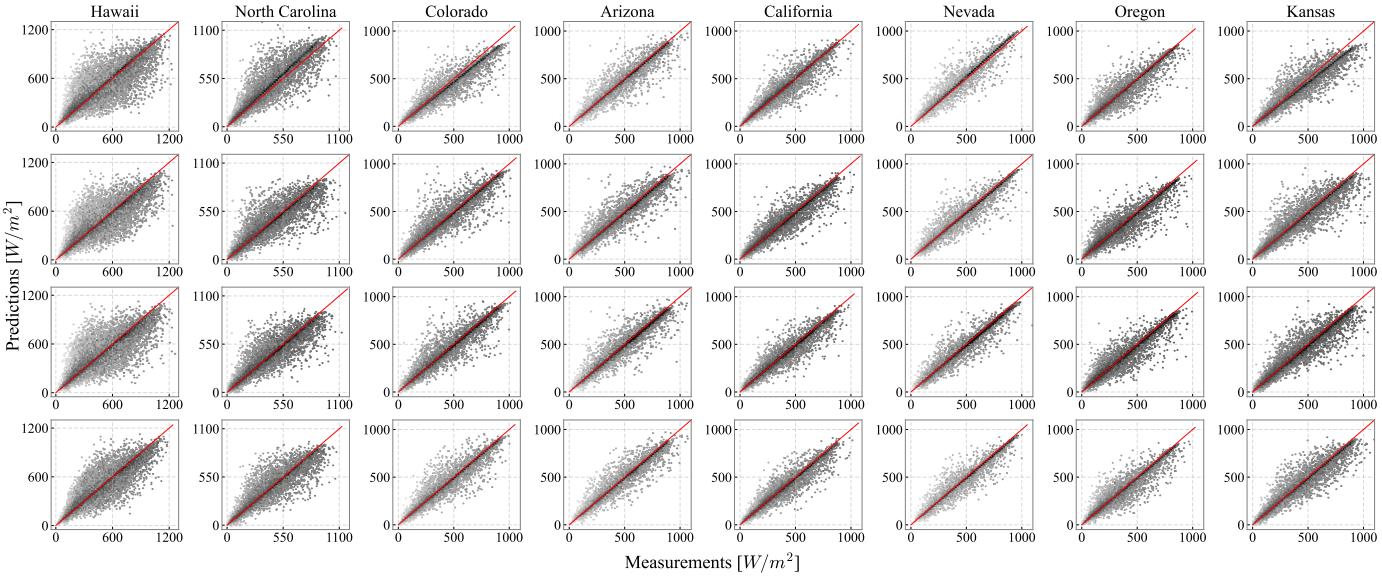

Figure 11: The scatter plots of 5-min forecasting results produced by the four training methods: (1) localized, (2) centralized, (3) conventional FL, and (4) the FL-STANN framework from top to bottom rows, respectively. Darker color indicates denser data points.

after about 40 rounds. However, the FL-STANN loss is smaller and more stable, which indicates better convergence in training and higher forecasting accuracy compared to the conventional FL setting.

Then, we compare the FL-STANN with centralized and localized methods. In this test, we use different features for each dataset to account for the capability of FL-STANN to handle various input data. For example, Hawaii and North Carolina only utilize CSI time series as an input, but Oregon and Kansas employ CSI with three endogenous variables $\mathcal{B}$, $\mathcal{L}$, and $\mathcal{V}$. It is worth mentioning that the conventional FL setting is not able to deal with data of various dimensions because the feature extractors can not be aggregated through the FedAvg algorithm due to different input layer sizes. Thus, we only use CSI as input without other features. The forecasting results generated by the STANN using the three training methods are summarized in Table 5. The numerical results are represented in the form of "mean±std" for all 6 forecast horizons, and the asterisk * symbol means the model is trained with CSI only.

According to the results, three observations can be made from the results. Firstly, model performance differs largely among clients. The substantial variations in model performance might be explained by the skewed data distribution of each dataset as shown in Fig.2. For instance, the localized model trained on the Nevada dataset has a forecast skill of $6.1 \pm 2.4\%$, but it achieves a high forecast skill of $23.2 \pm 3.9\%$ on the Nevada dataset as it has more evenly distributed data.

Secondly, the FL-STANN strategy outperforms the others on all local datasets, and the performance of localized and centralized models can be improved by using FL-STANN. For example, in North Carolina, nMBE of localized STANN is $8.9 \pm 0.6\%$, which implies the predictions are larger than the actual values and results in a deteriorated accuracy. In contrast, the centralized FL-STANN is learned from the less skewed data that is gathered from all the training sets, thus the nMBE is reduced to $-3.9 \pm 1.1\%$ and the forecasting skill is improved to $11.2 \pm 2.1\%$. By using FL-STANN, the nMBE can be further reduced to $-0.7 \pm 1.0\%$ with the highest skill of $17.3 \pm 1.3\%$.

Thirdly, opposite to the localized and centralized models, the model trained in conventional FL setting tends to have better performance on highly skewed data and degraded performance on more balanced distributed data. Noticeably, the forecast skills are $18.1 \pm 3.8\%$ and 19.6 ± 2.7 on the Nevada and Arizona dataset, which are lower than that of the other models. This is because the feature extractor parameters are aggregated with the others during FL rounds in this setting so that its capability of extracting local features is weakened and influenced by the other extractor trained on skewed data. The visualized forecasting results are shown in Fig. 11. The predictions of STANN learned by localized, centralized, and conventional FL methods are widely spread with more outliers, whereas FL-STANN significantly reduced biases, especially in the NC, CO, and KS datasets, resulting in improved forecasting accuracy.

In terms of runtime for training, on the one hand, the localized STANN spent the least times of 0.048, 0.07, 0.116, and 0.164 hours for the four input variable combinations on each dataset. On the other hand, the centralized model took the longest time which is about 8 times that of the localized one since the amount of training data is proportionally increased. As for the FL-STANN, the training time is increased compared with the localized model because of the additional time for the parameter aggregation process at each updating round. However, the increased runtime of the FL-STANN can be acceptable and compro-

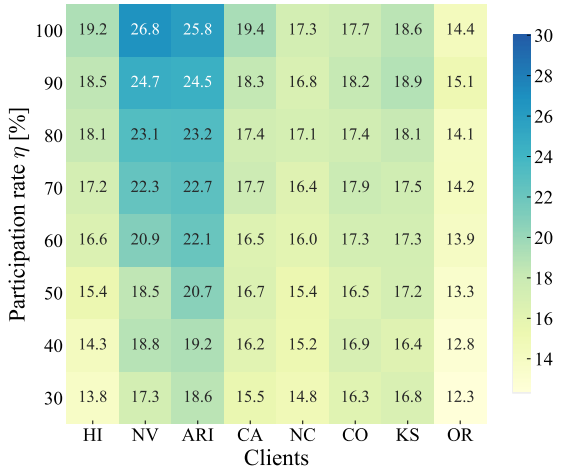

Figure 12: Forecast skills (%) produced by the FL-STANN on different data sources at various participation rates η.

mised with enhanced robustness and model performance.

Finally, we investigate the impact of the number of participants on the FL-STANN framework. The participation rate is referring to the ratio of clients participating in every FL round. It depends on either FL settings or clients themselves. For example, with a huge population of devices, server samples only part of them every update due to communication overhead. Fig. 12 illustrates the forecast skills under different participation rates η and the results are mean skills for all forecast horizons. We assume that there will be at least 3 participants in every updating round to guarantee a comparable model performance, thus the rate is changed from a minimal rate of 30% to 100%. For example, when $\eta = 0.3$, we randomly choose 3 clients from all datasets to train the FL-STANN model at each updating round. As we can see, the forecast skill of each client is promoted as η increases, which suggests that more clients are encouraged to participate in FL training for better performance. In practice, clients can make informed choices about how and whether to participate at all based on convergence states or if there is new data collected.

6. Conclusion

In this work, we propose a STANN model for multi-horizon solar forecasting, which integrates both spatial and temporal attention with a detachable encoder-decoder architecture. The model can not only handle complex temporal dynamics but provide insight into feature space for relevant feature selection using its context spatial encoder. To have a better generalization of different data sources, a novel distributed forecasting strategy is developed for the STANN model based on FL. It allows the model to preserve the local data characteristics while improving the forecasting ability of the temporal decoder through FL aggregation without revealing the local data. We verify the effectiveness of our method by conducting extensive experiments on different real-world datasets and features. The results demonstrate better forecasting accuracy and robustness to a variety of input features than the benchmarks. The STANN performance can be further enhanced via the FL separable training on various data sources, with significant improvement in forecast skills up to 11.2%.

In the future study, we will further develop the framework for real-world smart environment design applications considering finer-grained FL aggregation algorithms, communication bottlenecks, and privacy issues.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

AI University Research Centre (AI-URC) through XJTLU Key Programme Special Fund (KSF-P-02) and research development fund of XJTLU (RDF-17-01-28). This work is also partly supported by Singapore MOE AcRF Tier 1 fundings with grant numbers A-0008299-00-00 and A-0008552-01-00.

References

[1] C. Breyer, D. Bogdanov, A. Gulagi, A. Aghahosseini, L. S. Barbosa, O. Koskinen, M. Barasa, U. Caldera, S. Afanasyeva, M. Child, et al., On the role of solar photovoltaics in global energy transition scenarios, Progress in Photovoltaics: Research and Applications 25 (8) (2017) 727–745.

[2] L. Bird, M. Milligan, D. Lew, Integrating variable renewable energy: challenges and solutions, Tech. Rep. NREL/TP-6A20-60451, National Renewable Energy Laboratory (NREL), Golden, CO, USA (2013).

[3] S. S. Govada, T. Rodgers, L. Cheng, H. Chung, Smart environment for smart and sustainable hong kong, in: Smart Environment for Smart Cities, Springer, 2020, pp. 57–90.

[4] G. Notton, G. A. Faggianelli, C. Voyant, S. Ouedraogo, G. Pigelet, J.-L. Duchaud, Solar radiation forecasting for smart building applications, Computational Intelligence Techniques for Green Smart Cities (2022) 229.

[5] D. Lazos, A. B. Sproul, M. Kay, Development of hybrid numerical and statistical short term horizon weather prediction models for building energy management optimisation, Building and Environment 90 (2015) 82–95.

[6] A. Rysanek, R. Nuttall, J. McCarty, Forecasting the impact of climate change on thermal comfort using a weighted ensemble of supervised learning models, Building and Environment 190 (2021) 107522.

[7] M. Diagne, M. David, P. Lauret, J. Boland, N. Schmutz, Review of solar irradiance forecasting methods and a proposition for small-scale insular grids, Renewable and Sustainable Energy Reviews 27 (2013) 65–76.

[8] A. Naveen Chakkaravarthy, M. Subathra, P. Jerin Pradeep, N. Manoj Kumar, Solar irradiance forecasting and energy optimization for achieving nearly net zero energy building, Journal of Renewable and Sustainable Energy 10 (3) (2018) 035103.

[9] J. Antonanzas, N. Osorio, R. Escobar, R.Urraca, F. J. M. de Pison, F. Antonanzas-Torres, Review of photovoltaic power forecasting, Solar Energy 136 (2016) 78–111.

[10] H. Yang, B. Kurtz, D. Nguyen, B. Urquhart, C. W. Chow, M. Ghonima, J. Kleissl, Solar irradiance forecasting using a ground-based sky imager developed at UC San Diego, Solar Energy 103 (2014) 502–524.
[11] H. Wang, Z. Lei, X. Zhang, B. Zhou, J. Peng, A review of deep learning for renewable energy forecasting, Energy Conversion and Management 198 (2019) 111799.
[12] F. Wang, Z. Xuan, Z. Zhen, K. Li, T. Wang, M. Shi, A day-ahead PV power forecasting method based on LSTM-RNN model and time correlation modification under partial daily pattern prediction framework, Energy Conversion and Management 212 (2020) 112766.
[13] H. Wen, Y. Du, X. Chen, E. Lim, H. Wen, L. Jiang, W. Xiang, Deep learning-based multi-step solar forecasting for PV ramp-rate control using sky images, IEEE Trans. Ind. Inform. 17 (2021) 1397–1406.
[14] X. Zhang, S. L. Y. Li and, H. F. Hamann, B. Hodge, B. Lehman, A solar time based analog ensemble method for regional solar power forecasting, IEEE Trans. Sustain. Energy 10 (2019) 268–279.
[15] E. Ogliari, A. Dolara, G. Manzolini, S. Leva, Physical and hybrid methods comparison for the day ahead pv output power forecast, Renewable energy 113 (2017) 11–21.
[16] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, Proceedings of Machine Learning and Systems 2 (2020) 429–450.
[17] S. B. Taieb, A. F. Atiya, A bias and variance analysis for multistep-ahead time series forecasting, IEEE transactions on neural networks and learning systems 27 (1) (2015) 62–76.
[18] M. Rana, I. Koprinska, V. G. Agelidis, Forecasting solar power generated by grid connected PV systems using ensembles of neural networks, in: 2015 International Joint Conference on Neural Networks (IJCNN), 2015, pp. 1–8.
[19] Y. Zhang, M. Beaudin, H. Zareipour, D. Wood, Forecasting solar photovoltaic power production at the aggregated system level, in: 2014 North American Power Symposium (NAPS), 2014, pp. 1–6.
[20] D. Salinas, V. Flunkert, J. Gasthaus, T. Januschowski, Deepar: Probabilistic forecasting with autoregressive recurrent networks, International Journal of Forecasting 36 (3) (2020) 1181–1191.
[21] H. T. Pedro, C. F. Coimbra, Assessment of forecasting techniques for solar power production with no exogenous inputs, Solar Energy 86 (7) (2012) 2017–2028.
[22] Z. Liu, C. K. Loo, K. Pasupa, A novel error-output recurrent two-layer extreme learning machine for multi-step time series prediction, Sustainable Cities and Society 66 (2021) 102613.
[23] M. Rana, A. Rahman, Multiple steps ahead solar photovoltaic power forecasting based on univariate machine learning models and data re-sampling, Sustainable Energy, Grids and Networks 21 (2020) 100286.
[24] M. Guermoui, F. Melgani, C. Danilo, Multi-step ahead forecasting of daily global and direct solar radiation: A review and case study of Ghardaia region, Journal of Cleaner Production 201 (2018) 716–734.
[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).
[26] L. Bai, L. Yao, S. S. Kanhere, X. Wang, Q. Sheng, STG2Seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting, ArXiv abs/1905.10069 (2019).
[27] Z. Chen, E. Jiaze, X. Zhang, H. Sheng, X. Cheng, Multi-task time series forecasting with shared attention, 2020 International Conference on Data Mining Workshops (ICDMW) (2020) 917–925.
[28] S. Sharda, M. Singh, K. Sharma, RSAM: Robust self-attention based multi-horizon model for solar irradiance forecasting, IEEE Transactions on Sustainable Energy 12 (2) (2020) 1394–1405.
[29] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, T. Januschowski, Deep state space models for time series forecasting, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 31, 2018.
[30] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial intelligence and statistics, PMLR, 2017, pp. 1273–1282.
[31] S. V. Dasari, K. Mittal, G. Sasirekha, J. Bapat, D. Das, Privacy enhanced energy prediction in smart building using federated learning, in: 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), IEEE, 2021, pp. 1–6.
[32] M. Khalil, M. Esseghir, L. Merghem-Boulahia, Federated learning for energy-efficient thermal comfort control service in smart buildings, in: 2021 IEEE Global Communications Conference (GLOBECOM), IEEE, 2021, pp. 1–6.
[33] R. Kumar, A. A. Khan, J. Kumar, Zakria, N. A. Golilarz, S. Zhang, Y. Ting, C. Zheng, W. Wang, Blockchain-federated-learning and deep learning models for covid-19 detection using CT imaging, IEEE Sensors Journal 21 (14) (2021) 16301–16314.
[34] Y. Liu, J. J. Q. Yu, J. Kang, D. Niyato, S. Zhang, Privacy-preserving traffic flow prediction: A federated learning approach, IEEE Internet of Things Journal 7 (2020) 7751–7763.
[35] N. Gholizadeh, P. Musilek, Federated learning with hyperparameter-based clustering for electrical load forecasting, Internet of Things 17 (2022) 100470.
[36] M. N. Fekri, K. Grolinger, S. Mir, Distributed load forecasting using smart meter data: Federated learning with Recurrent Neural Networks, International Journal of Electrical Power & Energy Systems 137 (2022) 107669.
[37] Y. Wang, I. L. Bennani, X. Liu, M. Sun, Y. Zhou, Electricity consumer characteristics identification: A federated learning approach, IEEE Transactions on Smart Grid 12 (4) (2021) 3637–3647.
[38] Y. Li, J. Li, Y. Wang, Privacy-preserving spatiotemporal scenario generation of renewable energies: A federated deep generative learning approach, IEEE Transactions on Industrial Informatics 18 (4) (2021) 2310–2320.
[39] J. Lin, J. Ma, J. Zhu, A privacy-preserving federated learning method for probabilistic community-level behind-the-meter solar generation disaggregation, IEEE Transactions on Smart Grid 13 (1) (2021) 268–279.
[40] X. Zhang, F. Fang, J. Wang, Probabilistic solar irradiation forecasting based on variational bayesian inference with secure federated learning, IEEE Transactions on Industrial Informatics 17 (11) (2020) 7849–7859.
[41] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, Foundations and Trends® in Machine Learning 14 (1–2) (2021) 1–210.
[42] H. T. C. Pedro, D. P. Larson, C. F. M. Coimbra, A comprehensive dataset for the accelerated development and benchmarking of solar forecasting methods, Renewable Sustainable Energy 11 (Jun. 2019).
[43] A. Andreas, T. Stoffel, Baseline measurement system (BMS), Tech. Rep. DA-5550-56488, National Renewable Energy Laboratory (NREL), Golden, CO, USA (1981).
[44] P. Ineichen, R. Perez, A new airmass independent formulation for the linke turbidity coefficient, Solar Energy 73 (3) (2002) 151–157.
[45] W. Holmgren, C. Hansen, M. Mikofski, Pvlib python: a python package for modeling solar energy systems, J. Open Source Softw. 3 (2018) 884.
[46] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, Least angle regression, The Annals of statistics 32 (2) (2004) 407–499.
[47] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, CoRR abs/1409.0473 (2015).
[48] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase represen-

tations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078 (2014).
[49] F. Sattler, S. Wiedemann, K.-R. Müller, , W. Samek, Robust and communication-efficient federated learning from non-i.i.d. data, IEEE Trans. Neural Netw. Learn. Syst. 31 (2020) 3400–3413.
[50] H. Zhou, Y. Zhang, L. Yang, Q. Liu, K. Yan, Y. Du, Short-term photovoltaic power forecasting based on long short term memory neural network and attention mechanism, IEEE Access 7 (2019) 78063–78074.
[51] N. Wu, B. Green, X. Ben, S. O'Banion, Deep transformer models for time series forecasting: The influenza prevalence case, arXiv preprint arXiv:2001.08317 (2020).