2024 IEEE 25th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC) | 979-8-3503-9318-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/SPAWC60668.2024.10694252

SHARE: A Distributed Learning Framework For Multivariate Time-Series Forecasting

Wei Ye†, Prashant Khanduri‡, Jiangweizhi Peng†, Feng Tian†,
Jun Gao*, Jie Ding†, Zhi-Li Zhang†, Mingyi Hong†
†University of Minnesota Twin Cities; ‡Wayne State University; *Meta Platforms, Inc.
ye000094@umn.edu, khanduri.prashant@wayne.edu, peng0504@umn.edu, tianx399@umn.edu,
jungao@meta.com, dingj@umn.edu, zhzhang@cs.umn.edu, mhong@umn.edu

***Abstract*—In this work, we propose a distributed time series forecasting framework for scenarios where multiple computing nodes collaboratively make predictions using only locally available data streams. Our proposed framework utilizes a *hierarchical* architecture, consisting of multiple *local* models and a *global model*, and provides an efficient training algorithm to achieve high prediction accuracy. Key features of the framework are: (i) *Simple collaboration*, where a carefully designed global model enables the system to leverage the correlations among different local streams; (ii) *Flexibility*, where each node can choose its local model from a wide variety of prediction models (e.g., RNNs or Transformers) suitable to its compute resources; (iii) *Privacy and communication efficiency,* where messages communicated between the nodes are low-dimensional embeddings. The proposed approach's effectiveness is demonstrated theoretically and numerically using a number of time series forecasting tasks, with the (surprising) observation that the proposed distributed models can match or even outperform centralized ones.**

***Index terms*—** Time series forecasting, Vertical Learning, Distributed Optimization

I. Introduction

Time series forecasting finds applications in many scientific fields, including healthcare [1], finance [2], meteorology [3], and traffic engineering [4], etc. Traditionally, regression-based approaches have been popular for time series forecasting [5], [6]. However, the recent success of deep learning architectures has shown to outperform simple regression methods [7]–[12]. These recent developments along with large-scale and distributed collection of data demand the development of efficient distributed time series prediction algorithms. However, there is a lack of distributed forecasting modules at scale, especially for the class of problems where each node observes only a subset of the entire collection of data streams. In this work, we consider a time series forecasting problem, where multiple distributed nodes collaborate using their local data streams, to make *both* local (e.g., predict street-level traffic volume) and global predictions (e.g., predict city-level traffic volume).

This research is in part supported by Meta Platforms, Inc. Additionally, Hong's work is partially supported by the USDA National Institute of Food and Agriculture (NIFA) and the National Science Foundation (NSF) National AI Research Institutes Competitive Award under the number 2023-67021-39829, and by NSF EPCN-2311007 and CNS-2003033. Zhang's work is partially supported by the NSF CNS-2212318, 2220286, and 2220292. Ding's work is partially supported by the Army Research Office under the Early Career Program Award with the grant number W911NF-23-10315.

Only a handful of works have addressed the problem, and the proposed learning algorithms are simple extensions of centralized ones [13], [14]. The existing distributed time series forecasting algorithms are unsuitable for meeting data privacy and model flexibility requirements. Key challenges in developing a distributed time series forecasting system are:

- Each node observes only a subset (i.e., partial dimensions) of the time series, making learning challenging since the key to accurate predictions is to utilize correlation among different local data streams [8]–[10].
- In applications such as healthcare [1], data collected at each node (e.g., hospital) may contain sensitive information, so it cannot be directly shared among the collaborating nodes. Therefore, it is highly desirable that a distributed learning algorithm can avoid raw data sharing.
- Finally, the existing time series forecasting models lack theoretically grounded foundations even in centralized settings. A major obstacle in the analysis of distributed time series forecasting models arises from the fact that the training does not follow the standard independent and identically distributed (i.i.d.) gradient assumptions, thereby leading to biased updates.

In this work, we develop a new framework, diStributed HierARchical dEcomposition (SHARE) that addresses the above-mentioned challenges while providing theoretical convergence guarantees. Key to the SHARE is a unique hierarchical architecture, where a number of local models are built to process locally observed data streams, while a global model is designed to fuse the embeddings of the local models. Specifically, SHARE has the following major benefits:

Flexibility: It endows the local computing nodes the flexibility of picking their own model architectures so that each local node can choose from a variety of time series models (e.g., RNN, CNN), depending on their computing resources as well as the complexity of the local task. **Effective information sharing:** A global model is designed to help fuse the information generated from the local models, so as to best leverage the correlations among the data streams. **Theoretical guarantees:** We model the sampling of the time series as a Markov chain and show that the proposed algorithms, when used to train the hierarchical time series forecasting model, converge to the set of stationary solutions of the corresponding training problem.

Finally, we note that the applicability of SHARE is much beyond time series prediction tasks and can be, in fact, used

to solve general distributed learning problems. Specifically, SHARE subsumes a number of popular modern distributed learning systems wherein the feature space of a single data sample is distributed among the distributed nodes, the so-called *vertical* federated learning [15]. Note that SHARE is substantially different from the standard distributed learning, or the so-called *horizontal* federated learning, where each node observes data samples that share the same feature space.

A. Preliminaries

We consider a distributed learning problem with K nodes. Each node $k \in [K]$ observes a time series $\mathbf{x}_k^n \in \mathbb{R}^{d_k}$ where $n \in \mathbb{N}$ indicates the time index. We represent the global time series (observed across all the nodes) as $\mathbf{x}^n \in \mathbb{R}^d$, where $d := \sum_{k=1}^K d_k$ and $\mathbf{x}^n := [(\mathbf{x}_1^n)^\top, \ldots, (\mathbf{x}_K^n)^\top]^\top$. Denote a sequence of multidimensional time series samples as:

$$\bar{\mathbf{x}}^n := [\mathbf{x}^{n+1}, \mathbf{x}^{n+2}, \ldots, \mathbf{x}^{n+D}] \in \mathbb{R}^{d \times D} \quad \text{for } n \in \mathbb{N} \tag{1}$$

where $D > 0$ is *time window*. Note that each node only observes a partial time series; we define the next τ samples as:

$$\bar{\mathbf{y}}^n := [\mathbf{x}^{n+D+1}, \ldots, \mathbf{x}^{n+D+\tau}] \in \mathbb{R}^{d \times \tau} \quad \text{for } n \in \mathbb{N}, \tag{2}$$

where τ is the *time horizon* of the prediction. The *global* task is to use $\bar{\mathbf{x}}^n$ to predict a function of $\bar{\mathbf{y}}^n$, denoted as $g(\cdot) : \mathbb{R}^{d \times \tau} \to \mathbb{R}^e$, which transforms a vector of data points at a given time to a summary statistics. For example, when predicting the total traffic across all agents, we have:

$$g(\mathbf{y}^n) := \left[\sum_{k=1}^K \sum_{i=1}^{d_k} \mathbf{x}_k^{n+D+1}[i], \cdots, \sum_{k=1}^K \sum_{i=1}^{d_k} \mathbf{x}_k^{n+D+\tau}[i] \right].$$

Note here that we have utilized the notation $(\bar{\mathbf{x}}^n, \bar{\mathbf{y}}^n) \sim \Pi$ for $n \in \mathbb{N}$ to denote the feature label pairs and where Π denotes the underlying distribution that generates the time series data. Observe that two samples $\bar{\mathbf{x}}^n$ and $\bar{\mathbf{x}}^\ell$ for $n \neq \ell$ may not be i.i.d. since consecutive training samples are generated using a sliding window. This contrasts the traditional training regimes that assume access to i.i.d. samples for training.

Next, let us move to the distributed setting, where each node has access to only a subset of the dimensions of the global time series $\mathbf{x}_k^n \in \mathbb{R}^{d_k}$. In this case, following (1) – (2) we define the tuple $(\bar{\mathbf{x}}_k^n, \bar{\mathbf{y}}_k^n)$ as:

$$\begin{aligned} \bar{\mathbf{x}}_k^n &:= [\mathbf{x}_k^{n+1}, .., \mathbf{x}_k^{n+D}] \in \mathbb{R}^{d_k \times D}, \\ \bar{\mathbf{y}}_k^n &:= [\mathbf{x}_k^{n+D+1}, .., \mathbf{x}_k^{n+D+\tau}] \in \mathbb{R}^{d_k \times \tau} \end{aligned} \tag{3}$$

for $n \in \mathbb{N}$. Moreover, the *local* task at each node $k \in [K]$ is to forecast the local time series up to the time horizon of τ utilizing a time window D. Since $\bar{\mathbf{y}}_k^n$'s are the local forecasting targets, throughout the paper, we will refer to them as *labels* available at node k.

II. The Proposed Framework

In this section, we begin by describing the hierarchical model adopted by the proposed SHARE framework, as well as its associated training problem.

Local models. Each node $k \in [K]$ maintains a local model (like a neural network, e.g., LSTM or RNN) parameterized by $\theta_k \in \Theta_k \subseteq \mathbb{R}^{P_k}$ that helps with the local prediction task. The input to the local models is the locally observed data streams while their output is referred to as the local embedding and is denoted for all $k \in [K]$ by $f_k(\theta_k; \bar{\mathbf{x}}_k^n)$ for all $n \in \mathbb{N}$, where $\bar{\mathbf{x}}_k^n$ is defined in (3). We note that these embeddings are often low-dimensional and usually have dimensions much smaller than $d_k \times \tau$ for each $k \in [K]$. For brevity of notation, in the following we denote $f_k(\theta_k; \bar{\mathbf{x}}_k^n)$ as $f_k(\theta_k)$.

Global model. In addition to the local models, the server maintains a global model parameterized by $\theta_0 \in \Theta_0 \subseteq \mathbb{R}^{P_0}$, whose goal is to capture the correlations among different time series in order to improve the prediction accuracy. The input to the global model is the local embeddings while its output (referred to as the global embedding) is denoted by $f_0(\theta_0; f_1, \ldots f_K)$. This global embedding will have the same dimension as the label to be predicted, e.g., for time series forecasting, the dimension of the global embedding will be $d \times \tau$ since it is designed to predict the d dimensional global time series for time horizon τ (see (1) and (2)).

Problem. The goal of the forecasting problem is to learn the optimal local parameters at each node while jointly optimizing the global parameters. Defining $\theta \in \Theta$ with $\theta := [\theta_0^\top, \theta_1^\top, \ldots, \theta_K^\top]^\top$, $\Theta := \cup_{i=0}^K \Theta_i \subseteq \mathbb{R}^P$ and $P = \sum_{i=0}^K P_i$. The goal of the forecasting algorithm is to learn the parameters $\theta \in \Theta$ jointly in a distributed fashion such that a given loss function $\mathcal{L} : \mathbb{R}^P \to \mathbb{R}$ is minimized as

$$\min_{\theta \in \Theta} \Big\{ \mathcal{L}(\theta) := \mathbb{E}_{(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \sim \Pi} [\mathcal{L}(f_0(\theta_0; f_1(\theta_1), \ldots, f_K(\theta_K)); (\bar{\mathbf{x}}, \bar{\mathbf{y}}))] \Big\}, \tag{4}$$

A popular choice of $\mathcal{L}(\cdot)$ for time series forecasting problems is the ℓ_2 and ℓ_1 loss [10]. In particular, we assume the following.

Assumption 1. We assume that the loss function $\mathcal{L}(\theta)$ satisfies $\mathcal{L}(\cdot\ ; (\bar{\mathbf{x}}, \bar{\mathbf{y}})) = \sum_{k=1}^K \mathcal{L}(\cdot\ ; (\bar{\mathbf{x}}, \bar{\mathbf{y}}_k))$.

We note that ℓ_2 and ℓ_1 losses popular for time-series forecasting satisfy Assumption 1. Problem (4) is in general non-convex since $\mathcal{L}(\cdot)$ is non-convex in the parameters $\theta \in \Theta$ for many problems of practical interest, e.g., when the local and global models are neural networks [16]. Next, we develop an efficient algorithm to solve problem (4).

A. Algorithm Development

Note that the overall goal is to develop a joint optimization and communication strategy such that the global loss as defined in (4) is minimized. Specifically, each node aims to learn a local model θ_k, while at the same time learning a global model θ_0 that is shared among all nodes. To begin with, let us first compute the stochastic gradients with respect to the local and global models. First, let us sample $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \sim \Pi$ and evaluate the stochastic gradients (SG) as

$$\begin{aligned} \textbf{Local SG: } \nabla_{\theta_k} \mathcal{L}(\theta; (\bar{\mathbf{x}}, \bar{\mathbf{y}})) &= \nabla_{\theta_k} \mathcal{L}(\theta_0, \theta_1, \ldots, \theta_k; (\bar{\mathbf{x}}, \bar{\mathbf{y}})) \\ &= \nabla_{\theta_k} f_k(\theta_k)\ \nabla_{f_k} f_0(\theta_0, f_1(\theta_1), \ldots, f_K(\theta_K)) \\ &\quad \nabla_{f_0} \mathcal{L}(f_0(f_1(\theta_1), \ldots, f_K(\theta_K); \theta_0); \bar{\mathbf{y}}), \end{aligned} \tag{5}$$

which follows from the application of the chain rule. Similar to **Local SG**, we can evaluate the **Global SG** as

$$\textbf{Global SG: } \nabla_{\theta_0}\mathcal{L}(\theta;(\bar{\mathbf{x}},\bar{\mathbf{y}})) = \nabla_{\theta_0}\mathcal{L}(\theta_0,\theta_1,\ldots,\theta_K;(\bar{\mathbf{x}},\bar{\mathbf{y}})) = \nabla_{\theta_0} f_0(f_1(\theta_1),\ldots,f_K(\theta_K);\theta_0) \nabla_{f_0}\mathcal{L}(f_0(f_1(\theta_1),\ldots,f_K(\theta_K);\theta_0);\bar{\mathbf{y}}), \quad (6)$$

To implement an efficient algorithm, one would sample $(\bar{\mathbf{x}},\bar{\mathbf{y}})$ during training and implement (vertical) SG-type algorithm using the SG estimates (5) and (6). However, a major challenge in developing such an algorithm requires the sharing of raw data among nodes since the computation of **Local SG** and **Global SG** depends on the time series collected from all the nodes, i.e., $(\bar{\mathbf{x}},\bar{\mathbf{y}})$, as can be seen from (5)-(6). In the following, we develop an efficient training algorithm that solves the above problem without any raw data sharing.

The detailed steps to solve (4) are listed in Algorithm 1. We note that after the initial sharing of the local embeddings across nodes (Step 5), the local nodes compute the partial gradient $\nabla\mathcal{L}(\cdot;\bar{\mathbf{y}}_k^r)$ using their locally observed data in Step 6. These local partial gradients are then shared among nodes to construct the full partial gradient $\nabla\mathcal{L}(\cdot;\bar{\mathbf{y}}^r)$ using Assumption 1 in Step 7. In Step 8, the rest of the partial gradients are evaluated using the shared local embeddings to construct the **Local SG** and **Global SG** in (5) and (6), respectively. Finally, the global and local models are updated simultaneously using the computed stochastic gradients in Steps 9 and 10, respectively.

We note that the SHARE framework provides flexibility to choose the global model as well as the local models. The local models extract the local time series information, while the global models learn the correlations between the different locally observed time series. Next, we provide the convergence performance of the proposed framework.

B. Convergence Guarantees

Next, we provide the convergence guarantee achieved by Algorithm 1 under the assumption that the training samples $(\bar{\mathbf{x}}^n,\bar{\mathbf{y}}^n)$ for $n\in\mathbb{N}$ utilized for solving (4) are non i.i.d. (see (1) and (2)). To model the non-i.i.d samples, we make the following assumption about the data-generating process.

Assumption 2. Data-generating process $\{\bar{\mathbf{x}}^n,\bar{\mathbf{y}}^n\}_{n\geq 0}$ follows a Markov chain trajectory with M states. The Markov chain is time-homogeneous, irreducible, and aperiodic and has a transition matrix $T\in\mathbb{R}^{M\times M}$ with stationary distribution Π^*.

Algorithm 1 The SHARE Framework

1: **Input**: Rounds $r=\{0,1,\ldots,R-1\}$, local learning rates: $\{\eta_k^r\}_{k=1}^K$, server learning rate: η_0^r
2: **Initialize:** Parameters, $\{\theta_0^0,\ \theta_1^0,\ \ldots,\ \theta_K^0\}$
3: **for** $r=0$ to $R-1$ **do**
4: Sample $(\bar{\mathbf{x}}_k^r,\bar{\mathbf{y}}_k^r)\sim\Pi$ (see (3)) $\forall k\in[K]$
5: `Share` $f_k(\theta_k^r)$ $\forall k\in[K]$ with all nodes via the server
6: `Compute` $\nabla\mathcal{L}_{f_0}(f_0(\theta_0^r;f_1,..,f_K);\bar{\mathbf{y}}_k^r)$ at each node and share with all nodes via the server
7: `Compute` $\nabla\mathcal{L}_{f_0}(\cdot;\bar{\mathbf{y}}^r)=\sum_k\nabla\mathcal{L}_{f_0}(f_0;\bar{\mathbf{y}}_k^r)$ at each node using Assumption 1
8: `Compute` $\nabla_{\theta_k}f_k(\theta_k^r)$, $\nabla_{f_k}f_0(\theta_0^r,f_1,\ldots,f_K)$, and $\nabla_{\theta_0}f_0(f_1,\ldots,f_K;\theta_0^r)$ at each node (see (5) and (6)).
9: `Global Update` $\theta_0^{r+1}=\theta_0^r-\eta_0^r\nabla_{\theta_0}\mathcal{L}(\theta^r;(\bar{\mathbf{x}}^r,\bar{\mathbf{y}}^r))$
10: `Local Update` $\theta_k^{r+1}=\theta_k^r-\eta_k^r\nabla_{\theta_k}\mathcal{L}(\theta^r;(\bar{\mathbf{x}}^r,\bar{\mathbf{y}}^r))$
11: **end for**

Note that Assumption 2 assumes that the samples are generated from a finite state space. This assumption can easily be relaxed with an additional set of assumptions when the data samples $\{\bar{\mathbf{x}}^n,\bar{\mathbf{y}}^n\}_{n\geq 0}$ are generated from a Markov series rather than a Markov chain [17]. Next, we make some assumptions about the loss function $\mathcal{L}(\cdot;(\bar{\mathbf{x}},\bar{\mathbf{y}}))$.

Assumption 3. **Local SG** and **Global SG** derived in (5) and (6) are bounded, i.e., $\|\nabla_{\theta_i}\mathcal{L}(\theta;(\bar{\mathbf{x}},\bar{\mathbf{y}}))\|\leq G$ for $i\in\{0,1,..,K\}$. Also, the loss function is L-Lipschitz smooth.

Assumption 3 is a standard assumption in first-order algorithm analyses and has also been made in earlier works [17]. Next, we state the convergence performance of SHARE.

Theorem 1. *If we choose the step-sizes* $\eta_i^r=c_i/r^q$ *with* $q\in(1/2,1)$ *and* $c_i>0$ *for* $i\in\{0,1,\cdots,K\}$, *then SHARE achieves:* $\mathbb{E}[\min_{1\leq r\leq R}\{\|\nabla_\theta\mathcal{L}(\theta^r)\|^2\}]=\mathcal{O}\left(\frac{1}{R^{1-q}}\right)$.

Note that the above result allows each node to tune its learning rate independently, which is particularly useful since each node can train a different ML model. It also matches the guarantees of a centralized Markov Chain Gradient Descent for $K=1$ [17]. The distributed Markov chain gradient descent under different settings has been considered in the past. Specifically, [18], [19] assume that each node has access to a complete data sample; however, in the SHARE framework, each node only observes a partial time series. Additionally, in [18], [19], each node learns the same model and has the same local step sizes across the network. In contrast, in SHARE framework, each node can flexibly choose the local model. Consequently, each node can perform local learning utilizing different step sizes. Also, the SHARE framework allows each node to maintain a global model to learn the correlations among different time series. These key distinctions make the analysis of SHARE significantly challenging compared to prior works.

III. Numerical Experiments

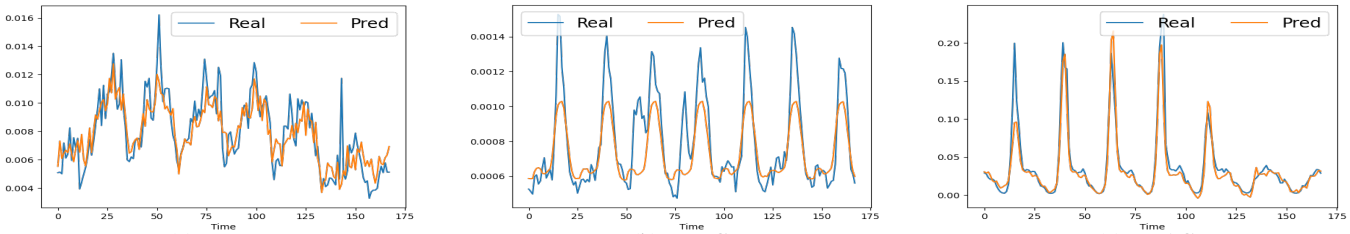
We now evaluate the proposed distributed multivariant time-series framework on three real-world datasets with widely used neural network architectures to showcase its advantages in achieving competitive performance with centralized models while offering flexibility in local model selection.

A. Experiment setup

Datasets: We evaluate the proposed SHAREframework on three real-world datasets, each containing data streams with specific physical meaning and inherent semantic correlations. ***[D1]*** Wiki data (WIKI) [20] extracts hourly user requests of four particular Wiki pages, and each is represented as a time series. ***[D2]*** Traffic data (TRAC) [21] reports hourly road occupancy rates of San Francisco Bay area freeways. The dataset contains 2-year-length records of 861 sensors. ***[D3]*** Electricity data (ELEC) [22] contains hourly electricity consumption of 321 customers from 2012 to 2014.

TABLE I: Comparison between the performance of SHARE with other baselines for mid- and long-term forecasting. The experiments are conducted in both centralized (LSTM, TCN, and LSTNet) and distributed settings (SHARE(LSTM), SHARE(TCN), SHARE(LSTNet)), using MLP as the global model. These models use D=36 hours historical window to predict τ={24, 36, 48, 72, 168} hours future horizons. Performance is assessed using RMSE and DTW, where smaller values denote superior performance. The scaling factor for the values is $\times 10^{-2}$. The best-performing model for each dataset is highlighted in bold.

Models		Prophet		LSTM		SHARE(LSTM)		TCN		SHARE(TCN)		LSTNet		SHARE(LSTNet)	
Metrics		RMSE	DTW	RMSE	DTW	RMSE	DTW	RMSE	DTW	RMSE	DTW	RMSE	DTW	RMSE	DTW
WIKI	24h	1.212	4.658	**1.209**	4.510	**1.209**	**4.495**	1.319	4.914	1.218	4.618	1.245	4.719	1.273	4.738
	36h	1.522	10.18	**1.330**	**5.983**	1.359	6.004	1.389	6.277	1.341	6.112	1.418	6.456	1.375	6.170
	48h	1.596	15.75	**1.415**	**7.178**	1.429	7.260	1.660	8.407	1.504	7.588	1.468	7.567	1.415	7.176
	72h	1.819	25.25	1.617	9.748	**1.557**	**9.579**	1.664	10.12	1.575	9.654	1.612	10.11	1.620	9.691
	168h	2.124	44.70	1.724	15.67	1.726	15.67	1.808	16.60	1.747	16.01	1.723	16.04	**1.721**	**15.56**
ELEC	24h	8.401	43.55	**6.374**	**24.43**	6.555	24.89	6.618	25.11	6.796	26.21	6.741	25.94	6.696	25.56
	36h	8.145	61.01	**6.616**	**30.39**	6.828	30.88	6.972	31.39	6.932	31.68	6.980	31.91	6.918	31.43
	48h	8.803	87.13	**6.862**	36.05	6.990	**35.97**	7.026	36.16	7.104	36.87	7.357	38.25	7.131	36.83
	72h	8.871	113.9	**7.048**	**44.36**	7.365	45.36	7.584	46.17	7.448	46.82	7.614	48.04	7.407	45.35
	168h	9.207	176.4	7.549	68.84	8.113	72.32	**7.530**	**67.78**	7.835	71.16	7.782	71.95	7.864	70.84
TRAC	24h	6.384	**19.33**	5.288	20.40	5.250	20.34	6.272	23.86	5.382	21.08	5.368	21.00	**5.159**	20.08
	36h	6.670	42.35	5.568	25.94	5.739	26.32	6.042	28.21	5.695	26.71	5.528	26.12	**5.419**	**25.38**
	48h	6.891	67.66	5.869	31.63	6.003	31.61	6.634	34.93	5.976	31.93	**5.742**	31.22	5.819	**31.21**
	72h	6.984	95.84	6.514	42.43	6.770	43.18	8.623	53.56	6.341	41.79	**6.103**	**40.53**	6.265	41.07
	168h	7.160	144.0	7.098	71.12	6.915	68.08	7.816	79.16	6.905	70.37	6.807	69.11	**6.768**	**67.82**



(a) WIKI. (b) ELEC. (c) TRAC.

Fig. 1: Forecasting results for SHARE (LSTNet) for mid-term (168 hours) forecasting using the 36-hour window.

Baselines: We consider three widely used neural network architectures: LSTM [7], TCN [11], and LSTNet [10]. We implement them in two ways: (1) as a centralized model and (2) as local models within the SHARE framework. When incorporating them as local models, we use notation like SHARE(LSTNet) to indicate that local nodes employ the LSTNet architecture. Additionally, we also consider Prophet [23], a popular statistics-based time-series forecasting method.

Training schema: The learning-based approach adopts the moving window schema defined in Sec I-A; for each time sequence sample, we predict the future τ timestamps based on the historical D timestamps. In contrast, the Prophet uses the cross-validation schema [24], where we continuously update the training set by incorporating past data samples as time moves forward and keep re-fitting the Prophet model for prediction.

Implementation: For all experiments, we use the following setups unless specified otherwise. We first split the datasets into the training, validation, and test sets with the ratio of $0.4 : 0.1 : 0.5$ and apply the min-max scaler for the data normalization. The TCN and LSTNet adhere to default implementations [10], [11], while other modules are configured with 2 hidden layers and 64 hidden units. For the proposed distributed learning framework SHARE, we emulate 4 local nodes with MLP as the global model. We distribute all data streams to these nodes based on the physical similarity of the data source. For instance, data streams collected at the same or similar geo-location are grouped into one local node. Note that each node contains at least one stream, and the number of streams in each local node may vary. We utilize the Adam optimizer [25] with a batch size of 128 and learning rates of 0.03 for both local and global model training. Lastly, each experiment is run three times with sufficient epochs, and the average performance is reported.

Metrics: We evaluate the performance using two metrics: Root Mean Square Error (RMSE) and Dynamic Time Warping distance (DTW). RMSE calculates the distance between predictions and ground truth in Euclidean space, while DTW further considers trends. For both metrics, the lower the values, the better the prediction performance.

B. Multivariate time series forecasting

Following the above setup, we evaluate the effectiveness of SHARE by comparing its performance to Prophet and other centralized models. Table I summarizes the numerical experiment results, and Fig. 1 visualizes representative results.

SHARE vs. Prophet: The reported results show that SHARE outperforms the Prophet, despite the Prophet benefiting from more samples through continuous expansion of the training set. This could be attributed to the following two reasons: (1) SHARE has stronger learning capabilities than the statistical-based method due to the use of neural networks; (2) The Prophet models each data stream independently, which prevents the local client from leveraging the information from other data streams. In contrast, SHARE is able to utilize the correlation among all the streams.

TABLE II: The performance of SHARE with mixed local models. We use $D = 36$ hours historical window to predict future horizons of $\tau = \{24, 36, 72\}$ hours, evaluated using the RMSE metric. The value scaling factor is $\times 10^{-2}$.

Dataset	WIKI			ELEC			TRAC		
SHARE (mix)	24h	36h	72h	24h	36h	72h	24h	36h	72h
	1.32	1.40	1.63	6.91	7.14	7.26	5.40	5.82	6.31

TABLE III: The performance of SHARE(TCN) on the WIKI dataset with different global and local learning rates. We use $D = 36$ hours historical window to predict future horizons of $\tau = 24$ hours, evaluated using the RMSE metric. The value scaling factor is $\times 10^{-2}$.

η	$\mathbf{L}_{0.001}$	$\mathbf{L}_{0.003}$	$\mathbf{L}_{0.01}$	$\mathbf{L}_{0.03}$	$\mathbf{L}_{0.1}$
$\mathbf{G}_{0.001}$	1.218	1.188	1.193	1.195	1.228
$\mathbf{G}_{0.003}$	1.239	1.216	1.193	1.212	1.190
$\mathbf{G}_{0.01}$	1.226	1.249	1.256	1.324	1.215
$\mathbf{G}_{0.03}$	1.386	1.333	1.264	1.269	1.270
$\mathbf{G}_{0.1}$	2.516	1.686	1.679	2.232	1.893

SHARE vs. Centralized models: We also compare SHARE with the centralized baselines, which have access to all input data streams. As shown in Table I, SHARE delivers competitive performance compared to the centralized baselines even *without* local access to all data streams. Interestingly, in certain cases, SHARE even outperforms the centralized baselines. This may be attributed to its enhanced modeling capabilities, stemming from the composition of multiple local models and its effective learning of correlations among them.

Flexibility of local model selection: As noted earlier, SHARE is not only compatible with different local models but also allows individual nodes to choose it independently. Table I reports the performance of SHARE using one of three different types of neural network models in all local nodes. We see that LSTNet is superior on the traffic dataset, while LSTM is better on the electricity dataset. We conjecture that this is due to the higher correlation among the traffic data streams, leading to better performance when utilizing LSTNet. It's important to emphasize that the number of data streams allocated to each node is heterogeneous, implying that local models at each node will differ, although they may share a similar architecture. Moreover, we also tried a mixed local models configuration, labeled as DIVIDE(mix), where one-third of the local nodes utilize the LSTM model, another third use TCN, and the remaining nodes employ LSTNet. Table II presents results evaluated via the RMSE. We observe that even with the mixed local model, SHARE can still converge, highlighting the flexibility of DIVIDE in local model selection.

Asynchronous learning rate: Table III evaluates the performance of SHARE(TCN) on the WIKI dataset with different learning rates for the local model and global model. Here, η_0 in $\mathbf{G}_{\eta_0}$ represents the learning rate of the global model, while η_k in $\mathbf{L}_{\eta_k}$ denotes the local learning rate of the clients. The result implies our framework's adaptability to different global and local learning rates, confirming its robust performance with a small learning rate.

REFERENCES

[1] Ons Aouedi, Alessio Sacco, Kandaraj Piamrat, and Guido Marchetto, "Handling privacy-sensitive medical data with federated learning: Challenges and future directions," *IEEE Journal of Biomedical and Health Informatics*, pp. 1–14, 2022.

[2] Bilberto Batres-Estrada, "Deep learning for multivariate financial time series," 2015.

[3] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Advances in neural information processing systems*, vol. 28, 2015.

[4] Dong Wang, Wei Cao, Jian Li, and Jieping Ye, "Deepsd: Supply-demand prediction for online car-hailing services using deep neural networks," in *2017 IEEE 33rd international conference on data engineering (ICDE)*. IEEE, 2017, pp. 243–254.

[5] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung, *Time series analysis: forecasting and control*, John Wiley & Sons, 2015.

[6] Roger Frigola, *Bayesian time series learning with Gaussian processes*, Ph.D. thesis, University of Cambridge, 2015.

[7] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, nov 1997.

[8] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 753–763.

[9] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi, "Gman: A graph multi-attention network for traffic prediction," in *Proceedings of the AAAI conference on artificial intelligence*, 2020, vol. 34, pp. 1234–1241.

[10] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st international ACM SIGIR conference on research & development in information retrieval*, 2018, pp. 95–104.

[11] Shaojie Bai, J Zico Kolter, and Vladlen Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[12] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, vol. 35, pp. 11106–11115.

[13] Maneesha Perera, Julian De Hoog, Kasun Bandara, and Saman Halgamuge, "Multi-resolution, multi-horizon distributed solar pv power forecasting with forecast combinations," *Expert Systems with Applications*, vol. 205, pp. 117690, 2022.

[14] Xiaoqian Wang, Yanfei Kang, Rob J Hyndman, and Feng Li, "Distributed arima models for ultra-long time series," *International Journal of Forecasting*, 2022.

[15] Kang Wei, Jun Li, Chuan Ma, Ming Ding, Sha Wei, Fan Wu, Guihai Chen, and Thilina Ranbaduge, "Vertical federated learning: Challenges, methodologies and experiments," *arXiv preprint arXiv:2202.04309*, 2022.

[16] Prateek Jain, Purushottam Kar, et al., "Non-convex optimization for machine learning," *Foundations and Trends® in Machine Learning*, vol. 10, no. 3-4, pp. 142–363, 2017.

[17] Tao Sun, Yuejiao Sun, and Wotao Yin, "On markov chain gradient descent," *Advances in neural information processing systems*, vol. 31, 2018.

[18] Tao Sun and Dongsheng Li, "Decentralized markov chain gradient descent," *arXiv preprint arXiv:1909.10238*, 2019.

[19] Hoi-To Wai, "On the convergence of consensus algorithms with markovian noise and gradient bias," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 4897–4902.

[20] "Page view statistics for wikimedia projects," https://dumps.wikimedia.org/other/pagecounts-raw/, Accessed: 2022-08-23.

[21] "California department of transportation," https://pems.dot.ca.gov/, Accessed: 2022-09-21.

[22] "Electricityloaddiagrams20112014 data set," https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014, Accessed: 2022-09-21.

[23] Sean J Taylor and Benjamin Letham, "Forecasting at scale," *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018.

[24] "Prophet diagnostics," https://facebook.github.io/prophet/docs/diagnostics.html, Accessed: 2024-04-11.

[25] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.