FEDERATED LEARNING BASED WORKLOAD PREDICTION IN CLOUD COMPUTING

Nguyen Quoc Khanh*, Tran Quang Duc*, Nguyen Van Toan†, Tong Van Van*
[*] Hanoi University of Science and Technology, Hanoi, Vietnam
[+] Salesforce.com, Inc., California, USA

Abstract: Predicting CPU demand is a major challenge in cloud computing due to the volatile nature of CPU utilization. Moreover, gathering CPU utilization data from multiple virtual machines to develop a prediction method raises concerns about data privacy, transmission costs, and system scalability. To address these challenges, this paper introduces FL-LSTM, a novel workload prediction technique that combines long short-term memory networks (LSTM) with federated learning (FL). In the FL-LSTM approach, each client uses LSTM along with its local CPU utilization data to create a local model. These local models are then aggregated to form a global model using the standard Federated Averaging (FedAvg) algorithm. We conducted a thorough evaluation of FL-LSTM using eight clusters of Google cluster traces and eight clusters of the Azure Public Dataset. Our results demonstrate that FedAvg outperforms alternative FL strategies, while FL-LSTM meets or surpasses the performance of other state-of-the-art methods for cloud workload prediction. Notably, FL-LSTM achieved a Mean Squared Error of 0.00438, representing improvements of 74.7% and 9.4% compared to ARIMA and HBNN, respectively. These findings highlight the potential of FL-LSTM as an effective solution for predicting CPU demand in cloud computing environments.

Keywords: CPU usage prediction, Cloud computing, Federated Learning, LSTM.

I. INTRODUCTION

Cloud computing has seen remarkable growth in recent years, with more enterprises leveraging services and applications on platforms like Amazon AWS, Google Cloud, and Alibaba. These providers enable businesses to autonomously procure cloud resources, deploy scalable solutions, and improve cost efficiency. Cloud computing providers aim to preconfigure servers in advance to ensure enhanced Quality of Service (QoS), characterized by minimal latency, high availability, and reliable performance. Accurate forecasting of CPU demand allows providers to anticipate improved QoS indicators and optimize resource utilization by reducing the number of preconfigured idle machines or allocated but unused resources. However, predicting cloud workload demands is challenging due to their variability, massive scale, diverse nature, and fluctuations.

The research community has demonstrated considerable interest in predicting CPU consumption. Given that CPU consumption rates are recorded as time-series data, their forecasting can be regarded as a time-series prediction challenge. Forecasting CPU utilization employs techniques such as Auto Regression (AR) [1], Moving Average (MA) [2], Autoregressive Integrated Moving Average (ARIMA) [3], Support Vector Regression (SVR) [4], Bayesian classifiers [5], efficient supervised learning-based Deep Neural Networks (esDNN) [6], and Long Short-Term Memory Networks (LSTMs) [7], each exhibiting varying degrees of efficacy in generating precise predictions. Traditionally, these techniques can be executed using two distinct methodologies. The initial approach involves utilizing data from a singular virtual machine (VM) to develop the predictive models. It has encountered the issue of data scarcity, particularly when implemented in a new VM. The second approach is executed in a centralized fashion, with training conducted on a central server utilizing CPU data aggregated from multiple VMs or clusters. Centralized prediction techniques have problems with system scalability and data protection. The scalability challenge pertains to the escalating difficulty of capturing, storing, and analyzing all data at a centralized server. For example, Google cluster traces comprised data from eight distinct Borg cells, with a compressed size of approximately 2.4 TB. The data privacy issue pertains to the sensitive nature of CPU utilization data, which is considered a vital metric for evaluating the performance of services and applications, especially in VMs and servers. Safeguarding CPU consumption data is crucial for all businesses utilizing cloud computing services.

In this paper, we address the challenges of cloud workload prediction with limit data by applying LSTM and Federated Learning (FL). Introduced by Google in 2016, FL is a framework where multiple clients collaboratively train a machine learning model while maintaining the privacy of their data under the coordination of a central aggregator. FL has the potential to transform critical domains, including CPU consumption forecasting. Our paper makes the following contributions:

- We introduce FL-LSTM, a combination of LSTM and Federated Averaging [8]. To our knowledge, this is the first attempt to estimate future CPU demands in cloud

Contact author: Duc Tran,
Email: ductq@soict.hust.edu.vn
Manuscript received: 03/12/2024, revised: 13/3/2025, accepted: 22/3/2025.

computing systems using FL. Previous studies have focused on optimizing power consumption or the number of virtual machines in the system. Given the sensitive nature and volume of workload training data in cloud systems, our approach is crucial for addressing privacy and scalability issues

- We conducted a comprehensive evaluation of FL-LSTM using CPU traces from eight Google clusters [9]. Our results demonstrate that Federated Averaging outperforms other FL strategies, including FaultTolerantFedAvg, FedAdagrad, FedMedian, FedTrimmedAvg, and Krum. Moreover, FL-LSTM has shown superiority over well-established techniques, with MSE improvements of 74.7% and 9.4% compared to ARIMA and HBNN, respectively.

The remainder of this paper is structured as follows: Section II reviews relevant studies on workload prediction using statistical methods and machine learning techniques, and provides an overview of FL. Section III presents a detailed description of the proposed workload prediction mechanism utilizing the LSTM model and FL. Section IV discusses the experimental setup and findings. Finally, Section V concludes the paper and suggests potential directions for future research.

II. BACKGROUND AND RELATED WORK

A. *CPU usage prediction methods*

CPU utilization forecasting techniques can be broadly categorized into statistical and machine learning-based approaches. Among statistical methods, Wu et al. [1] combined Kalman filter and Auto Regression to enhance CPU usage predictions while mitigating measurement inaccuracies. Fu and Zhou [3] employed ARIMA to forecast CPU values, addressing VM deployment issues in data centers and determining VM affinities. Khan et al. [10] explored Hidden Markov Models (HMM) to detect temporal relationships between VMs and predict CPU pattern variations. However, AR, ARIMA, and HMM have shown limitations in handling non-linearity in CPU time series and tend to underperform on time series with temporal variability, complex characteristics, or sudden shifts [11], [12].

Machine learning models offer advantages in learning non-linear correlations among data samples and have demonstrated effectiveness across various time series data. Di et al. [5] extracted features characterizing host load fluctuation and utilized Bayes classifier to improve long-term load predictions. Minxian et al. [6] introduced an efficient supervised learning-based deep neural network (esDNN) that transforms multivariate data into time series using a sliding window, modifying gate recurrent units to address CPU rate fluctuations and gradient vanishing problems.

Long Short-Term Memory (LSTM) networks, a type of recurrent neural network, have shown effectiveness in many comparative studies [13], [14]. Owing to its gating mechanism, the Long Short-Term Memory (LSTM) model effectively captures and associates patterns from distant past sequences, making it particularly well-suited for time series data such as server load, which often exhibits complex cyclical or trending behaviors. Unlike traditional models, LSTM does not rely on assumptions of linearity or stationary states, enabling it to adapt flexibly to nonlinear and heterogeneous datasets. Furthermore, LSTM requires relatively modest data volumes and computational resources, rendering it practical for a wide range of realworld applications. Rossi et al. [12] further refined this approach with Probabilistic LSTM (LSTMD) to reduce uncertainty stemming from statistical characteristics of observations.

While LSTM, esDNN, and LSTMD have shown promise, their training typically occurs on a central server using data collected from various VMs—a centralized learning approach. This method faces challenges in managing the vast data volumes generated by continuous cloud computing operations and raises concerns about data anonymity. Federated Learning (FL) emerges as a potential solution to these challenges.

B. *Federated Learning*

Federated Learning (FL), proposed by McMahan et al. [15], is a framework where a central server coordinates the training of a shared global model across a federation of participating client devices. The foundational algorithm of FL, Federated Averaging (FedAvg), was introduced by McMahan et al. [8]. In each FedAvg iteration, a subset of clients is selected, typically randomly, and the server distributes its global model to each client. Clients then perform Stochastic Gradient Descent (SGD) on their local loss functions and transmit the trained models back to the server. The server updates its global model by averaging these local models. FaultTolerantFedAvg extends this approach by incorporating fault-tolerant methods to handle device dropouts, enhancing FL’s resilience in heterogeneous and variable network environments.

Yin et al. [16] developed two robust distributed gradient descent algorithms: median-based gradient descent (FedMedian) and trimmed-mean-based gradient descent (FedTrimmedAvg). Both techniques achieve order-optimal statistical error rates for strongly convex losses, with FedTrimmedAvg showing superior rates when local sample sizes are limited.

Reddi et al. [17] introduced adaptivity through adaptive optimizers for both client and server optimization - FedOptFedOpt. Their federated variants—FedAdagrad, FedYogi, and FedAdam—demonstrate improved convergence in nonconvex settings with heterogeneous data, enhancing overall FL performance.

FedProx [18] is a federated optimization framework designed to address both statistical and systems heterogeneity in distributed networks. By introducing a proximal term to the local objective function, FedProx restricts local updates to remain closer to the global model, mitigating divergence caused by nonIID data. Additionally, it allows clients to perform variable amounts of work (partial solutions) based on their resource constraints, thereby improving robustness against stragglers.

Addressing Byzantine resilience, Blanchard et al. [19] presented Krum, which guarantees convergence in the presence of f Byzantine adversaries among n total clients. Krum combines squared-distance and majority-based methodologies, selecting the vector that minimizes the sum of squared distances to its $n - f$ nearest vectors. This approach exhibits a time complexity of $O(n^2 \cdot d)$, where d represents the parameter vector dimension.

III. METHODOLOGY

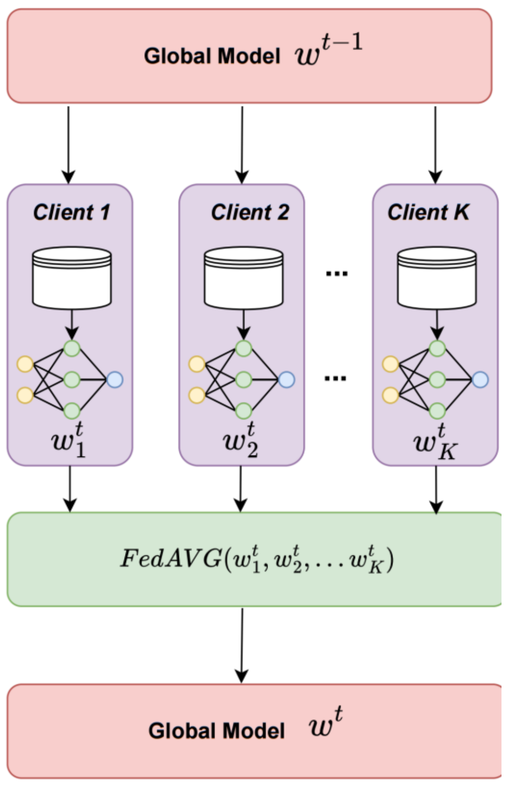
Conventional centralized CPU prediction methodologies typically involve three sequential stages: data processing, compilation, and model building. These approaches require data acquisition from numerous distinct VMs, leading to potential drawbacks such as data privacy concerns, data transfer costs, and scalability challenges. Moreover, the centralized server represents a single point of failure, risking data loss in the event of a crash.

To address these challenges, we propose FL-LSTM, a novel approach combining Long Short-Term Memory (LSTM) networks with Federated Learning (FL). In FL-LSTM, each VM or cluster is treated as a client. Let K represent the total number of clients and D_k denote the local CPU utilization dataset for client k. FL-LSTM achieves high scalability by allowing each client to construct a local training model on its CPU load dataset. Figure 1(a) illustrates the comprehensive design of FL-LSTM, while Algorithm 1 provides its pseudocode.

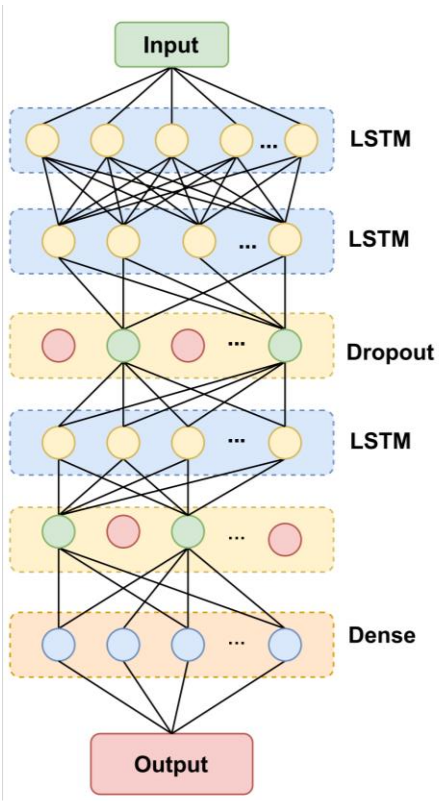
In each iteration, the algorithm chooses a subset S_t from the total K clients. The selection of the client subset S_t plays a pivotal role in simulating real-world scenarios, where not all clients are consistently available due to resource or connectivity constraints, particularly in systems with a large number of clients. As established in the foundational work by McMahan [15], the standard approach involves randomly selecting a fraction C of clients, with the number of selected clients determined by $m = MAX(C * K, 1)$, ensuring that at least one client participates in each round. However, in this study, S_t is chosen to include all available clients to minimize the impact of random selection on the evaluation of model performance. Subsequent sections will explore experimental configurations where only a subset of clients is active in each round, providing a more comprehensive analysis of the algorithm's performance under varying participation levels.

Each client k receives the global model $\mathbf{w}^{t-1}$ and trains its local model on the dataset D_k. FL-LSTM employs LSTM for local training, which possesses inherent benefits in cloud workload forecasting and surpasses alternative methods in numerous comparative analyses (see to Section II for further details).

Figure 1(b) depicts the LSTM architecture implemented within our FL-LSTM framework. The proposed model conceptualizes CPU prediction as a time series forecasting challenge, leveraging sequential CPU utilization measurements to forecast future resource consumption. The architecture comprises a precisely calibrated six-layer configuration: three LSTM layers for



(a)



(b)

Figure 1. (a) The architecture of FL-LSTM, (b) LSTM-based workload predict

Algorithm 1: FL-LSTM

```
1: Input:
    - $T$ is the number of FL rounds
    - $K$ is the total number of clients that are indexed by $k$
    - $D_k$ is the local dataset of client $k$
2: Initialize global model parameters $\mathbf{w}^0$
3: For each round $t$ do
4:     Select a subset $S_t$ of $K$ clients
5:     Send $\mathbf{w}^{t-1}$ to the clients that are elements of $S_t$
6:     For each client $k \in S_t$ in parallel do:
7:             $\mathbf{w}_k^t \leftarrow \text{LSTM}(\mathbf{w}_k^{t-1}, D_k)$
8: Determine the global model updates, i.e.,
             $\mathbf{w}^t \leftarrow \frac{1}{|S_t|}\sum_{k\in S_t}\mathbf{w}_k^t$
9: Output: The global model $\mathbf{w}^T$
```

hierarchical temporal feature extraction, two dropout layers for model regularization, and a terminal dense layer for prediction synthesis. The configuration of the three LSTM layers establishes a hierarchical feature extraction mechanism capable of capturing multi-scale temporal patterns inherent in workload data. Specifically, the initial layer extracts short-term fluctuations (e.g., hourly variations), the intermediate layer identifies medium-term trends (i.e., trends characterized by elevated usage during business hours), and the final layer considers long-term patterns (e.g., weekly periodicity). This multi-scale approach proves particularly useful in characterizing complex temporal dynamics across various time horizons. To fortify the model's generalization capabilities, we integrate two dropout layers parameterized by dropout rate p. These dropout layers stochastically deactivate input units during the training phase. Such regularization methodology is critical within our federated learning paradigm, wherein the model must effectively aggregate heterogeneous features from distributed clients exhibiting potentially divergent load patterns. The dropout mechanism mitigates the model's propensity to overfit specific training data patterns, instead facilitating the acquisition of robust features that generalize effectively across dynamic server load conditions. The architectural configuration culminates in a dense layer employing linear activation, which maps the extracted temporal features to a continuous CPU utilization prediction. This terminal configuration is well-suited for the regression problem at hand and has demonstrated exceptional predictive accuracy through extensive experimental evaluations. We have diligently evaluated and optimized critical LSTM parameters (such as dropout rate, batch size, and input shape) to enhance the model's predictive accuracy.

During the training process, local LSTM model parameter updates, denoted as $\mathbf{w}_k^t$, are transmitted to the central server. The server aggregates these updates to update the global model $\mathbf{w}^t$ using FL strategies. In FL-LSTM, we employ FedAvg to combine client models by averaging their parameters:

$$\mathbf{w}^t = \frac{1}{|S_t|}\sum_{k\in S_t}\mathbf{w}_k^t \tag{1}$$

FedAvg offers reduced privacy risks compared to central server storage, as weight updates are managed in memory and removed following aggregation. This approach has been successfully applied to various time series forecasting challenges, including 5G base station traffic prediction [20], vehicle count forecasting [21], and short-term energy usage prediction [22]. The present work demonstrates the advantages of FedAvg over alternative aggregating functions for workload predictions. A comprehensive examination of these benefits will be presented in the subsequent section.

The stopping criteria of the FL-LSTM algorithm consist of two main parameters: the number of communication rounds T and the number of epochs executed by each client. In real-world cloud workload prediction systems, servers typically operate continuously, with clients being dynamically initialized or released as needed. However, for our implementation of FL-LSTM, we set T to a specific value based on experimental results that demonstrated optimal performance-efficiency tradeoffs. Regarding the number of epochs on each client, we implement an early stopping mechanism based on validation loss. This approach ensures that when the global model reaches sufficient quality, the local LSTM training on each client will terminate once the validation loss stops improving, preventing overfitting and reducing unnecessary computational costs while maintaining prediction accuracy.

IV. EXPERIMENTS AND RESULTS

A. *Dataset specification*

This study utilizes Google cluster traces [9] and Azure Public Dataset V1 [23] to investigate FL-LSTM and other algorithms. We acquired cloud cluster traces from the Google Cloud Platform , which provide resource utilization information for eight Borg cells throughout May 2019. The dataset focuses on resource requests and usage, excluding details about end users, their data, or access patterns to storage systems and other services. We specifically use the CPU usage table, which contains information histograms for every 5-minute interval. Due to its substantial size (approximately 2.4TB compressed), the trace data is only accessible through Google BigQuery (https://cloud.google.com/bigquery). The Azure Public Dataset V1 comprises CPU traces for 2,013,767 virtual machines monitored over a three-month duration from November 2016 to February 2017. Each virtual machine has a minimum of 28 consecutive days of CPU load data, recorded at five-minute intervals. We randomly selected 8 clusters to assess our proposed technique.

We generated datasets encompassing the average CPU loads of all VMs at 5-minute intervals for each cluster. To expedite the training process and enhance convergence, we applied Max-Min scaling to normalize the data within the [0,1] range. This resulted in eight datasets, each containing approximately 8,000 data points of CPU utilization.

Table I. MSE and MAE of the standard LSTM, centralized LSTM and FL-LSTM

Horizon		**MSE**				**MAE**			
	Train-test ratio	1:2	2:2	4:2	8:2	1:2	2:2	4:2	8:2
10-minute-ahead prediction	LSTM	0.0062	0.0049	0.0054	0.0048	0.0564	0.0517	0.0552	0.0518
	Centralized LSTM	0.0054	0.0055	**0.0048**	0.0045	**0.0532**	0.0555	**0.0498**	**0.0477**
	FL-LSTM	**0.0053**	**0.0048**	0.0050	**0.0044**	0.0534	**0.0500**	0.0520	0.0484
30-minute-ahead prediction	LSTM	0.0087	0.0069	0.0072	**0.0067**	0.0688	0.0613	0.0635	**0.0606**
	Centralized LSTM	0.0101	0.0089	0.0072	0.0077	0.0768	0.0722	**0.0607**	0.0673
	FL-LSTM	**0.0076**	**0.0068**	**0.0070**	0.0071	**0.0661**	**0.0597**	0.0625	0.0640

Our experiments produce forecasts for two distinct time frames: 10 and 30 minutes in advance, representing 2- and 6-step trajectories of Google cluster traces, respectively. We employ multiple horizons to assess the ability of FL-LSTM and other methods to manage fluctuating CPU loads over time, given that an effective regressor for one horizon may not produce precise predictions for others. For each horizon, the last 20% of the data from each cluster was employed as the testing set for predictive outcomes. Additionally, we utilized 10%, 20%, 40%, and 80% of the data immediately before the evaluation set for model training, yielding train-test ratios of 1:2, 2:2, 4:2, and 8:2, respectively. By keeping the testing set the same while incrementally augmenting the training data, we seek to illustrate the efficacy of the FL-LSTM algorithm in producing accurate predictions, even with limited data for each cluster. The evaluation was performed on a machine equipped with a 20-core Intel i7 processor and 256 GB of RAM. The experiments in this study were simulated by running both the server and clients in parallel on a single workstation with the specified configuration. For a client equipped with a dual-core Intel i3 CPU and 4GB of RAM, training an LSTM model over 20 epochs required approximately 30 seconds per iteration.

B. Performance metrics

To evaluate the precision of FL-LSTM and several other methods, we employ Mean Squared Error (MSE) and Mean Absolute

Error (MAE), consistent with previous studies [3], [6], [12]. These metrics are calculated as follows:

$$\text{MSE} = \frac{\sum_{t=1}^{T}(x(t) - \hat{x}(t))^2}{T} \tag{2}$$

$$\text{MAE} = \frac{\sum_{t=1}^{T} | x(t) - \hat{x}(t))|}{T} \tag{3}$$

where T represents the number of samples, $x(t)$ denotes the actual value of the t-th sample, and $\hat{x}(t)$ represents the associated predicted value. MSE calculates the squared difference between $x(t)$ and $\hat{x}(t)$, while MAE denotes the absolute difference. Lower MSE and MAE values indicate better performance.

In the following subsections, we present the experimental results of our proposed methods and compare them with state-of-the-art studies

C. Comparison with the standard LSTM and Centralized LSTM

This section measures the performance of the FL architecture in enhancing the accuracy compare to the traditional LSTM model and the centralized LSTM architecture for prediction with constrained input data. The traditional LSTM model operates independently on each cluster, training on local data and making predictions accordingly. Otherwise, the centralized LSTM is designed to collecting all training data from clusters on a server and training one global model to generate predictions for all clusters. Table I show the MSE and MAE of the standard LSTM, centralized LSTM and FL-LSTM in different setups.

In 10-minute-ahead prediction, FL-LSTM consistently achieves the lowest MSE values across all train-test ratios, indicating superior performance in error minimization compared to LSTM and the centralized LSTM. For instance, at the 1:2 ratio, FL-LSTM achieves an MSE of 0.0053 compared to 0.0062 of LSTM. Similarly, for MAE, FL-LSTM generally outperforms, particularly at the 1:2 ratio. However, the centralized LSTM shows competitive performance with FL-LSTM at ratios of 4:2 and 8:2.

In 30-minute-ahead prediction, FL-LSTM exhibits the lowest predictive errors at 1:2, 2:2, and 4:2 ratios with MSEs of 0.0076, 0.0068, and 0.0070, respectively. For example, at a ratio of 1:2, FL-LSTM achieves MSE, which is 12.64% better compared to the MSE of LSTM (0.0087). Regarding MAE, FL-LSTM maintains an advantage across most ratios, particularly with the smallest value (0.0597) observed at 2:2.

Increasing the training dataset size reduces the MSE for all models in most cases, showing that more training data enhances performance. For example, FL-LSTM's MSE drops from 0.0053 at 1:2 to 0.0044 at 8:2 for the 10-minute-ahead prediction. Similarly, MAE decreases with larger train-test ratios, highlighting the generalization capability improvement with more training data. FL-LSTM consistently outperforms other models across both short (10-minute) and long (30-minute) prediction horizons in the limited data scenarios, where the train-test ratios are 1:2 or 2:2, confirming its superior performance and opening up the potential for deploying this solution in practice.

D. Comparison with other FL strategies

FL-LSTM employs FedAvg for federated aggregation. We compare it with various other aggregation algorithms reported in the literature, including FaultTolerantFedAvg, FedYogi, FedMedian, FedOpt, and Krum (refer to Section II for complete details). We assess these methodologies for forecasting CPU rates at 2 and 6 steps in advance. To ensure a fair comparison between FL-LSTM and alternative benchmarks, we utilize 20% of the total number of data samples for testing purposes for all scenarios while increasing training data.

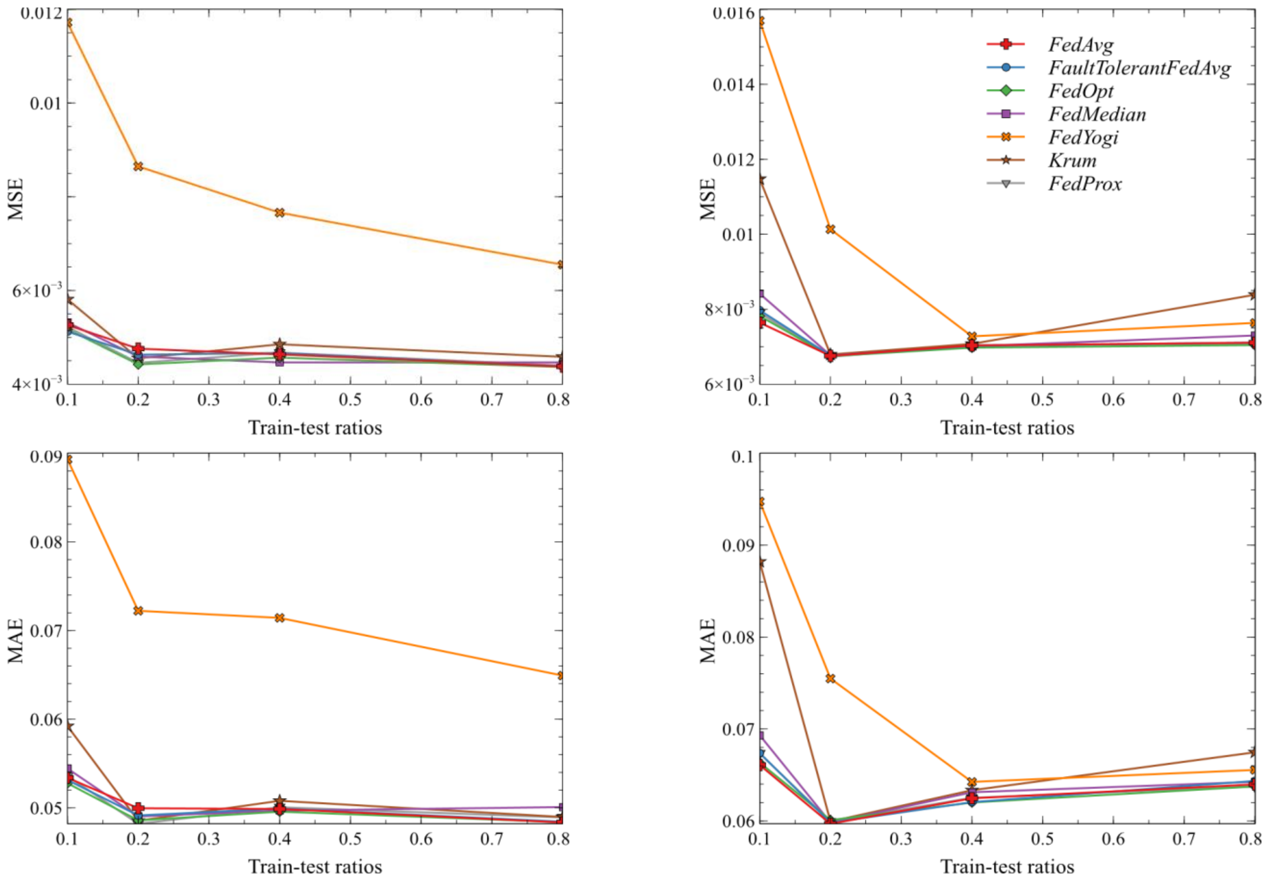


Figure 2. MSE and MAE of different FL strategies when used to render predictions. (a) and (c) represent the 2-step ahead predictions. (b) and (d) represent the 6-step ahead predictions. We use 20% of the total number of samples for testing.

Figure 2 illustrates the MSE and MAE of several Federated Learning (FL) procedures across the different train-test ratios. In summary, FedAvg produces superior prediction outcomes across both time horizons. It is important to note that while most FL algorithms are designed to address non-independent and identically distributed data issues, this may not be a primary concern for CPU utilization prediction in our context.

E. Comparison with other state-of-the-art methods

The aim of this section is to compare FLLSTM to other state-of-the-art methods, such as ARIMA [3], esDNN [6], [3], HBNN [12], and LSTMD [12]. ARIMA is a statistical analysis model that integrates autoregression and moving averages to forecast future trends using time series data. esDNN offers a supervised learning-based Deep Neural Network for predicting short-term CPU workload. HBNN uses Bayesian Neural Networks to forecast future workloads, whereas LSTMD applies the probabilistic LSTM for estimating future workload values.

In addition to the above methods, this paper also considers the Transformer model, which is a specialized type of neural network architecture. In this experiment, the Transformer model uses four stacked encoder blocks with multi-head self-attention (4 heads, key dimension 256), layer normalization, and a feed-forward network with convolutional and dropout layers. It applies global average pooling, followed by a dense layer (128 units, ReLU) and a single-unit output layer.

Table II clarifies the performance metrics of FL-LSTM, ARIMA, esDNN, HBNN, Transformer, and LSTMD utilizing Google cluster traces. Throughout all experiments, ARIMA consistently exhibited the worst performance relative to the other methods, and its accuracy does not change while increasing the training data. On the other hand, the other methods show improvement with more training data.

For the 10-minute-ahead prediction task, FL-LSTM demonstrates remarkable superiority across all train-test ratios. At a traintest ratio of 1:2, FL-LSTM achieves an MSE of 0.00526, representing significant improvements of 69.57%, 80.25%, 26.18%, and 13.35% compared to ARIMA, HBNN, LSTMD, and ESDNN, respectively. The performance of FL-LSTM shows consistent enhancement with increased training data, with its MSE decreasing from 0.00526 at a traintest ratio of 1:2 to an impressive 0.00438 at a ratio of 8:2. Notably, FL-LSTM outperforms all other methods across all evaluated ratios in terms of MSE. For the MAE metric, there is a minor exception at a train-test ratio of 4:2, where the deviation between FL-LSTM and

LSTMD is negligible (0.0520 and 0.0519), indicating comparable performance in this specific scenario. Nevertheless, FL-LSTM regains its lead at the 8:2 ratio with the lowest MAE of 0.0484. When comparing with Transformer, FL-LSTM demonstrates particularly impressive results. The Transformer model shows poor performance at lower training data volumes. Its MSE (0.03933 at 1:2 ratio) is 86.6% higher than that of FL-LSTM. Even at the highest train-test ratio of 8:2, where Transformer improves significantly to an MSE of 0.00513,

Table II. MSE and MAE of FL-LSTM and other state-of-the-art methods on Google Trace Dataset

Horizon		**MSE**				**MAE**			
	Train-test ratio	1:2	2:2	4:2	8:2	1:2	2:2	4:2	8:2
10-minute-ahead prediction	ARIMA	0.01730	0.01730	0.01729	0.01730	0.1031	0.1031	0.1031	0.1031
	HBNN	0.02664	0.01180	0.01248	0.00483	0.1270	0.0846	0.0904	0.0504
	LSTMD	0.00713	0.00580	0.00519	0.00489	0.0612	0.0552	**0.0518**	0.0508
	ESDNN	0.00607	0.00554	0.00579	0.00566	0.0577	0.0555	0.0571	0.0572
	Transformer	0.03933	0.01125	0.01185	0.00513	0.1517	0.0805	0.0797	0.0547
	FL-LSTM	**0.00526**	**0.00476**	**0.00502**	**0.00438**	**0.0534**	**0.0500**	0.0520	**0.0484**
30-minute-ahead prediction	ARIMA	0.01752	0.01752	0.01752	0.01752	0.1038	0.1038	0.1038	0.1038
	HBNN	0.03321	0.01725	0.00750	**0.00681**	0.1346	0.0940	0.0642	**0.0602**
	LSTMD	0.01389	0.00913	0.00954	0.00759	0.0892	0.0713	0.0722	0.0648
	ESDNN	0.00788	0.00816	0.00874	0.00775	**0.0661**	0.0687	0.0705	0.0670
	Transformer	0.02756	0.02376	0.02738	0.00806	0.1318	0.1170	0.1251	0.0694
	FL-LSTM	**0.00764**	**0.00676**	**0.00703**	0.00711	**0.0661**	**0.0597**	**0.0625**	0.0640

FL-LSTM still outperforms it with an MSE of 0.00438, representing a 14.6% improvement. The HBNN model exhibits the most dramatic improvement as training data increases, with MSE reducing from 0.02664 at a 1:2 ratio to 0.00483 at an 8:2 ratio, suggesting high data dependency. Applying the exact same configuration as in [12] (traintest ratio of 8:2), the MSE of FL-LSTM is 0.00438, showing substantial improvements of 74.7% and 9.4% compared to ARIMA and HBNN, respectively.

For the more challenging 30-minute-ahead prediction task, FL-LSTM maintains its superiority at lower train-test ratios ranging from 1:2 to 4:2, consistently achieving the lowest MSE and MAE values compared to other methods. At the 1:2 ratio, FL-LSTM's MSE of 0.00764 outperforms Transformer by 72.3% (Transformer's MSE: 0.02756) and HBNN by 77.0% (HBNN's MSE: 0.03321). As the train-test ratio increases to 8:2, HBNN slightly edges ahead with an MSE of 0.00681 compared to FL-LSTM's 0.00711, marking a 4.2% difference. However, considering FLLSTM's consistently superior performance across multiple ratios and metrics, this minor difference does not diminish its overall effectiveness. Furthermore, even at the 8:2 ratio, FL-LSTM still outperforms Transformer (MSE: 0.00806) by 11.8%. The Transformer model continues to show high sensitivity to training data volume in the 30-minuteahead prediction, with inconsistent improvement patterns when increasing from the 1:2 to 4:2 ratios. Its MSE marginally improves to 0.02376 at 2:2 but then deteriorates to 0.02738 at 4:2, suggesting instability in learning longer-term dependencies with varying data sizes.

Table III displays performance measures that compare FL-LSTM with various stateof-the-art methodologies on the Azure Public dataset, thereby substantiating FL-LSTM's exceptional efficacy in forecasting CPU consumption. For predictions made 10 minutes in advance, FL-LSTM consistently attains the minimal MSE values across all traintest ratios (0.1104 at 1:2, 0.1138 at 2:2, 0.1283 at 4:2, and 0.1105 at 8:2). This indicates substantial enhancements compared to conventional approaches such

Table III. MSE and MAE of FL-LSTM and other state-of-the-art methods on Azure Public dataset

Horizon		**MSE**				**MAE**			
	Train-test ratio	1:2	2:2	4:2	8:2	1:2	2:2	4:2	8:2
10-minute-ahead prediction	ARIMA	0.1983	0.1983	0.1983	0.1983	0.2229	0.2229	0.2229	0.2229
	HBNN	2.2497	9.5529	4.2168	2.6634	1.1695	2.0742	1.3283	1.0825
	LSTMD	0.2288	0.2117	0.2045	0.2166	0.2648	0.2136	0.2034	0.2514
	ESDNN	0.1966	0.1506	0.1500	0.1116	0.1930	**0.1624**	**0.1716**	**0.1545**
	Transformer	0.1970	0.1535	0.1900	0.1484	0.1932	0.1665	0.1876	0.1736
	FL-LSTM	**0.1104**	**0.1138**	**0.1283**	**0.1105**	**0.1738**	0.1806	0.1823	0.1748
30-minute-ahead prediction	ARIMA	0.1984	0.1984	0.1984	0.1984	0.2226	0.2226	0.2226	0.2226
	HBNN	6.5712	2.5329	2.6571	2.9125	1.7347	1.3602	1.0863	1.4502
	LSTMD	0.2391	0.2087	0.1939	0.2083	0.2679	0.2373	0.2023	0.2064
	ESDNN	0.1986	0.1508	0.1504	**0.0935**	0.1928	0.1650	0.1680	**0.1389**
	Transformer	0.1261	0.1195	0.1739	0.1824	**0.1557**	**0.1544**	**0.1841**	0.1850
	FL-LSTM	**0.1123**	**0.1078**	**0.1096**	0.1115	0.1779	0.1704	0.1661	0.1721

as ARIMA, which demonstrates consistently elevated error rates (0.1983 across all ratios). FL-LSTM surpasses more sophisticated methods such as HBNN, LSTMD, ESDNN, and Transformer models. For 30-minute forecasts, FLLSTM consistently exhibits superior performance, achieving the lowest MSE values across most train-test ratios. While ESDNN has marginally superior performance at the 8:2 ratio (0.0935 compared to FL-LSTM's 0.1115), FL-LSTM exhibits enhanced accuracy in situations with constrained training data (1:2, 2:2, and 4:2 ratios). This is especially beneficial for newly deployed cloud services where previous data may be limited.

The test results clearly show that FL-LSTM is better at making predictions than other methods, no matter how far into the future or how much training data is used. Particularly noteworthy is FL-LSTM's remarkable performance when operating with limited training data (train-test ratios of 1:2 or 2:2) on each client, where it consistently outperforms traditional statistical methods like ARIMA by significant margins and maintains an edge over sophisticated deep learning approaches such as HBNN, LSTMD, Transformer, and ESDNN. The approach simultaneously preserves data privacy by keeping sensitive CPU utilization metrics local to each client while delivering superior prediction accuracy. This advantage comes from two key benefits of FL-LSTM: the FL mechanism effectively gathers and uses information from various clients, and the improved LSTM layer design accurately identifies time-related trends in CPU usage data. Through multiple iterations of local training and global aggregation, FL-LSTM enhances model robustness and reduces prediction errors, especially in tasks for predicting server load involving distributed virtual machines with diverse load patterns. This makes FL-LSTM particularly valuable in realworld cloud environments where new virtual machines are continuously created with short operational periods, resulting in limited input data for prediction methods. In such scenarios, the system can combine initially weak models from individual VMs into a more accurate global model without compromising data sensitivity or prediction quality.

Table IV presents the execution times for ARIMA, esDNN, HBNN, LSTMD, Transformer, and FL-LSTM. FL-LSTM forecasts the future value of the CPU in 0.12 seconds, slightly above the execution times for HBNN and LSTMD, which are 0.09 and 0.10 seconds, respectively. Furthermore, esDNN requires an average of 0.50 seconds to predict a workload sample. ARIMA forecasts future values by averaging historical data and necessitates 6.94 seconds, the longest execution time among the evaluated methods, to generate a prediction. Overall, FL-LSTM can accurately predict future CPU values while requiring an acceptable execution duration.

Table IV. The execution time (in seconds) of FL-LSTM and other state-of-the-art methods.

Method	Execution time
ARIMA	6.94
esDNN	0.50
HBNN	0.09
LSTMD	0.10
Transformer	0.13
FL-LSTM	0.12

FL-LSTM offers significant advantages beyond execution time. By eliminating the need to gather and process all training data at the server, it conserves both network resources and processing time. Traditional methods require encryption, decryption, and validation steps during the transfer of data from clients to servers, which FL-LSTM avoids. Moreover, in conventional approaches, the server's capabilities often determine the overall scalability of the system. FL-LSTM mitigates this limitation by distributing the computational load across clients, potentially improving system scalability.

F. Ablation studies

This section examines the performance and behavior of various federated learning algorithms for time series forecasting under different operational conditions. We focus on evaluating algorithm convergence across multiple federation rounds and measuring performance resilience when faced with client disconnections, providing essential insights for real-world deployments.

Figure 3 evaluates the FL algorithms through each iteration to determine the number of iterations that can objectively assess the results of experimental FL algorithms. Here, we use MSE as the loss indicator for early stopping, Figures 3(a) and 3(b) represent the MSE of each FL strategy in each round for the 2-step and 6-step ahead predictions.

For the 2-step horizon prediction, most algorithms (FaultTolerantFedAvg, FedAvg, FedMedian, FedProx, and FedTrimmedAvg) show significant convergence by round 3, with their MSE values stabilizing around 0.0048-0.0052. After this point, the fluctuations in MSE values become minimal (typically less than 5-10% change), indicating that the algorithms have reached a relatively stable state. The Krum algorithm demonstrates a similar pattern, stabilizing around round 5 with an MSE of approximately 0.0048. However, FedYogi exhibits exceptional instability throughout all rounds, with dramatic fluctuations in MSE values even after multiple iterations (ranging from 0.006772 to 0.055009 between rounds 6-7).

The 6-step horizon prediction follows similar convergence patterns, with most algorithms stabilizing around round 3-4, albeit with generally higher MSE values due to the increased prediction difficulty. Based on these observations, we can conclude that approximately 5 rounds of federation are sufficient for most algorithms to reach stable performance in this time series forecasting task, making it a reasonable threshold for comparative evaluation of different federated learning strategies.

Table V presents the performance evaluation of the FL-LSTM algorithm under different client disconnection scenarios for both 10-minute and 30-minute prediction horizons. Here, each iteration has n (Disconnect rate * K) clients that disconnect from the centralized server, and the accuracy of each client takes the value from the last round in which it was still able to connect to the server. For the

10-minute-ahead forecasting, we observe that the model maintains relatively stable performance with disconnect rates up to 37.5%, with MSE increasing gradually from 0.00476 (no disconnections) to 0.00567. However, when the disconnect rate reaches 50%, the MSE significantly jumps to 0.00955, representing a 100.6% increase compared to the baseline, while MAE increases from 0.0500 to 0.0778. Similarly, for the 30-minute-ahead predictions, the performance degradation follows a similar pattern but starts from a higher baseline error (MSE of 0.00676 with no disconnections). As the disconnect rate increases, the MSE steadily rises to 0.01151 at 50% disconnection rate, representing a 70.3% increase in error.

Table V. MSE and MAE of FL-LSTM while disconnect clients.

Horizon	Disconnect rate	MSE	MAE
10-minute-ahead	0.00%	0.00476	0.0500
	12.50%	0.00493	0.0497
	25.00%	0.00538	0.0530
	37.50%	0.00567	0.0541
	50.00%	0.00955	0.0778
30-minute-ahead	0.00%	0.00676	0.0597
	12.50%	0.00806	0.0694
	25.00%	0.00944	0.0745
	37.50%	0.00995	0.0754
	50.00%	0.01151	0.0828

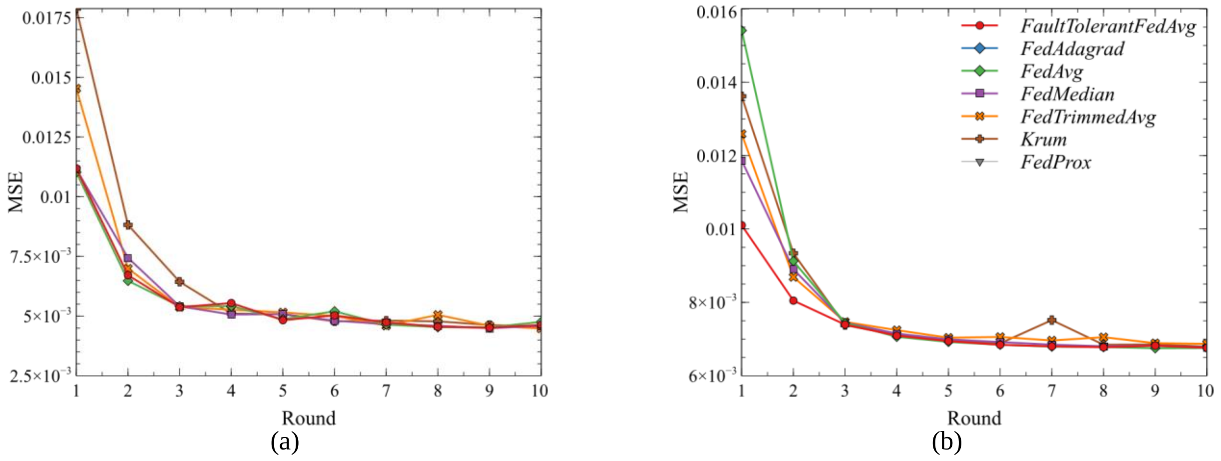


Figure 3. MSE of different FL strategies in each rounds. (a) and (b) represent the 2-step and 6-step ahead predictions. We use 1:1 as training-testing rate.

These results demonstrate that the FL-LSTM algorithm exhibits resilience to client disconnections up to approximately one-third of participants, beyond which prediction quality deteriorates substantially, particularly for shorter-term forecasts.

V. CONCLUSIONS

In this paper, we introduced FL-LSTM, a novel CPU utilization prediction method that combines Federated Learning (FL) and Long Short-Term Memory (LSTM) networks. FLLSTM offers inherent advantages by mitigating communication expenses and enhancing system scalability. Through comprehensive evaluation using Google cluster traces and Azure Public Dataset V1, we demonstrated that FL-LSTM outperforms most benchmarks in terms of both Mean Squared Error (MSE) and Mean Absolute Error (MAE).

While FL-LSTM shows slight inferiority to the standard LSTM in some scenarios, its key strengths lie in preserving data privacy and maintaining scalability. This is achieved by storing training data locally and conducting the training process exclusively on the client side.

Our future research directions include:

- Investigating different workload data characteristics to further enhance the performance of FL-LSTM.

- Exploring clustering techniques for VMs or clusters based on identified characteristics.

- Extending FL-LSTM to predict multiple related workload metrics simultaneously (e.g., CPU usage, memory usage, network traffic) within a multi-task learning framework.

REFERENCES

[1] W.D.Patrick, Natural sciences citations and references. Y. Wu, K. Hwang, Y. Yuan, and W. Zheng, “Adaptive workload prediction of grid performance in confidence windows,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 7, pp. 925–938, 2009.

[2] V. Priya and C. N. K. Babu, “Moving average fuzzy resource scheduling for virtualized cloud data services,” *Computer Standards & Interfaces*, vol. 50, pp. 251– 257, 2017.

[3] X. Fu and C. Zhou, “Predicted affinity based virtual machine placement in cloud computing environments,” *IEEE Transactions on Cloud Computing*, vol. 8, no. 1, pp. 246–255, 2017.

[4] R. Hu, J. Jiang, G. Liu, L. Wang *et al.*, “Efficient resources provisioning based on load forecasting in cloud,” *The Scientific World Journal*, vol. 2014, 2014.

[5] S. Di, D. Kondo, and W. Cirne, “Google hostload prediction based on bayesian model with optimized feature combination,” *Journal of Parallel and Distributed Computing*, vol. 74, no. 1, pp. 1820–1832, 2014.

[6] M. Xu, C. Song, H. Wu, S. S. Gill, K. Ye, and C. Xu, "esdnn: deep neural network based multivariate workload prediction in cloud computing environments," *ACM Transactions on Internet Technology (TOIT)*, vol. 22, no. 3, pp. 1–24, 2022.

[7] A. Rossi, A. Visentin, S. Prestwich, and K. N. Brown, "Bayesian Uncertainty Modelling for Cloud Workload Prediction," *IEEE International Conference on Cloud Computing, CLOUD*, vol. 2022-July, pp. 19–29, 2022. [8] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016. [Online]. Available: http://arxiv.org/abs/1602.05629

[8] J. Wilkes, "Yet more Google compute cluster trace data," Google research blog, Mountain View, CA, USA, Apr. 2020, posted at

[9] https://ai.googleblog.com/2020/04/yet-more-googlecompute-cluster-trace.html.

[10] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *2012 IEEE Network Operations and Management Symposium*. IEEE, 2012, pp. 1287–1294.

[11] A. A. Adebiyi, A. O. Adewumi, and C. K. Ayo, "Comparison of arima and artificial neural networks models for stock price prediction," *Journal of Applied Mathematics*, vol. 2014, no. 1, p. 614342, 2014.

[12] A. Rossi, A. Visentin, S. Prestwich, and K. N. Brown, "Bayesian uncertainty modelling for cloud workload prediction," in *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*. IEEE, 2022, pp. 19–29.

[13] B. Song, Y. Yu, Y. Zhou, Z. Wang, and S. Du, "Host load prediction with long short-term memory in cloud computing," *The Journal of Supercomputing*, vol. 74, pp. 6554–6568, 2018.

[14] S. Gupta, A. D. Dileep, and T. A. Gonsalves, "A joint feature selection framework for multivariate resource usage prediction in cloud servers using stability and prediction performance," *The Journal of Supercomputing*, vol. 74, pp. 6033–6068, 2018.

[15] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2023. [Online]. Available: https://arxiv.org/abs/1602.05629

[16] D. Yin, Y. Chen, K. Ramchandran, and P. L. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," *CoRR*, vol. abs/1803.01498, 2018. [Online]. Available: http://arxiv.org/abs/1803.01498

[17] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konecný,ˇ S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *CoRR*, vol. abs/2003.00295, 2020. [Online]. Available: https://arxiv.org/abs/2003.00295

[18] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[19] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Byzantine-tolerant machine learning," *CoRR*, vol. abs/1703.02757, 2017. [Online]. Available: http://arxiv.org/abs/1703.02757

[20] V. Perifanis, N. Pavlidis, R.-A. Koutsiamanis, and P. S. Efraimidis, "Federated learning for 5g base station traffic forecasting," *Computer Networks*, vol. 235, p. 109950, 2023.

[21] Y. Liu, J. James, J. Kang, D. Niyato, and S. Zhang, "Privacy-preserving traffic flow prediction: A federated learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7751–7763, 2020.

[22] M. Savi and F. Olivadese, "Short-term energy consumption forecasting at the edge: A federated learning approach," *IEEE Access*, vol. 9, pp. 95949–95969, 2021.

[23] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 153–167.

DỰ BÁO TẢI TRONG ĐIỆN TOÁN ĐÁM MÂY SỬ DỤNG HỌC LIÊN HỢP

Tóm tắt: Dự đoán tải CPU là một bài toán đầy thách thức trong điện toán đám mây do bản chất biến động của việc sử dụng CPU. Hơn nữa, việc thu thập dữ liệu sử dụng CPU từ nhiều máy ảo để phát triển phương pháp dự đoán làm dấy lên những lo ngại về quyền riêng tư dữ liệu, chi phí truyền tải và khả năng mở rộng hệ thống. Để giải quyết những thách thức trên, nghiên cứu giới thiệu giải pháp FL-LSTM, một kỹ thuật dự đoán tải kết hợp mạng Long short-term memory (LSTM) với học liên hợp (Federated Learning). Trong phương pháp FL-LSTM, mỗi máy khách sử dụng LSTM cùng với dữ liệu tải CPU cục bộ của nó để tạo ra mô hình cục bộ. Các mô hình cục bộ này sau đó được tổng hợp để hình thành một mô hình toàn cục sử dụng thuật toán Trung bình hóa Liên kết (FedAvg) tiêu chuẩn. Chúng tôi đã tiến hành đánh giá toàn diện về FL-LSTM sử dụng dữ liệu tải của tám cụm Google và tám cụm của Bộ dữ liệu Azure. Kết quả của chúng tôi chứng minh rằng FedAvg vượt trội hơn các chiến lược FL thay thế, trong khi FL-LSTM có độ chính xác đạt hoặc vượt qua của các phương pháp tiên tiến khác trong dự đoán tải trọng đám mây. Đáng chú ý, FL-LSTM đạt được MAE là 0,00438, cho thấy sự cải thiện về độ chính xác lần lượt là 74,7% và 9,4% ứng với với ARIMA và HBNN. Những phát hiện này nhấn mạnh tiềm năng của FL-LSTM như một giải pháp hiệu quả để dự đoán nhu cầu CPU trong môi trường điện toán đám mây.

Từ khóa: Dự đoán tải CPU, Điện toán đám mây, Học Liên hợp, LSTM

Nguyen Quoc Khanh, PhD student at the School of Information and Communication Technology, Hanoi University of Science and Technology. Research areas: machine learning, cloud computing, information security.

Email: khanh.nq@soict.hust.edu.vn

Tran Quang Duc, received his PhD in 2015 and became an Associate Professor in 2020. Currently working at the School of Information and Communication Technology, Hanoi University of Science and Technology. Research areas: machine learning, biometrics, information security.

Email: ductq@soict.hust.edu.vn

Nguyen Van Toan, Ph.D. (2018), is currently working as a researcher at Salesforce, Inc., United States. His research interests include security, infrastructure security, cloud computing, and human-computer interaction.

Email: nguyentoan@salesforce.com

Tong Van Van , Ph.D. (2021), is currently a faculty member at the School of Information and Communication Technology, Hanoi University of Science and Technology. His research interests include machine learning, information security, and blockchain technology.

Email: vantv@soict.hust.edu.vn