Collaborative Learning at the Edge for Air Pollution Prediction

I Nyoman Kusuma Wardana, *Graduate Student Member, IEEE,* Julian W. Gardner, *Fellow, IEEE,* Suhaib A. Fahmy, *Senior Member, IEEE*

***Abstract*—The rapid growth of connected sensing devices has resulted in enormous amounts of data being collected and processed. Air quality data collected from different monitoring stations is spatially and temporally correlated, and hence, collaborative learning can improve deep learning model performance. Research on collaborative learning at the edge has not specifically focused so far on air quality prediction, which is the subject of this work. We compare three collaborative learning strategies and implement them on edge devices, such as the Raspberry Pi and Jetson Nano, with communication facilitated through the MQTT protocol. Federated learning is shown to enhance model accuracy in comparison to local training alone. An approach called clustered model exchange reduces communication costs during training. Finally, our proposed spatiotemporal data exchange approach exploits information from neighboring sensing stations to enhance model performance. It achieves the highest accuracy in air quality predictions, outperforming other methods in minimizing loss during training. It results in RMSE improvements ranging from 0.525% to 8.934% when compared to models that are only trained locally. We compare the real training costs of the three methods on real hardware to validate them.**

***Index Terms*—Air pollution, collaborative learning, edge devices, federated learning, spatiotemporal.**

I. Introduction

POPULATION growth and economic development, expansion of urban areas, and industrial development have all contributed to the increase in air pollution worldwide [1], [2]. The major causes of air pollution are vehicle exhausts, industrial emissions, agricultural practices, and natural disasters. Exposure to air pollution has negative effects on human health [3] and economic activity [4]. Emotional and cognitive impairments [5], cardiovascular diseases [6], and lung cancer [7] are associated with air pollution. Air pollution can be in the form of particulate matter (PM), nitrogen dioxide (CO_2), carbon monoxide (CO), ozone (O_3), and sulfur dioxide (SO_2), among other pollutants [8]. Hence, building a forecasting system for air quality is crucial for health alerts [9].

Manuscript received Month Date, 2023; accepted Month Date, 2023. Date of publication Month Date, 2023; date of current version Month Date, 2023. This work was partly supported by the Indonesia Endowment Fund for Education (LPDP), Ministry of Finance, the Republic of Indonesia, under grant number Ref: S-1027/LPDP.4/2019.

I Nyoman Kusuma Wardana is with the School of Engineering, University of Warwick, Coventry CV4 7AL, UK, and also with the Department of Electrical Engineering, Politeknik Negeri Bali, Badung, 80364, Bali, Indonesia E-mail: Kusuma.Wardana@warwick.ac.uk.

Julian W. Gardner is with the School of Engineering, University of Warwick, Coventry CV4 7AL, UK. E-mail: J.W.Gardner@warwick.ac.uk.

Suhaib A. Fahmy is with King Abdullah University of Science and Technology, Thuwal 23955, Saudi Arabia. E-mail: suhaib.fahmy@kaust.edu.sa.

An extensive network of low-cost sensor nodes has recently been proposed to monitor air quality status [10]. Under this new paradigm, air quality monitoring aims to collect spatiotemporal air pollution data using many sensing devices, supplementing the traditional methodology with more accurate and expensive instrumentation [11]. Data collected from these sensing devices is often transmitted through the Internet, making monitoring air quality remotely possible. In response to the rapid growth of sensing devices, the amount of data generated, stored, and transmitted has increased significantly [12].

Machine Learning (ML) methods are being applied to air quality prediction tasks [13]. Deep Learning (DL), the subset of machine learning, offers a promising approach to predicting air quality status with its ability to extract features in the spatial and temporal domains.

Traditional centralized learning operates by gathering all data from end devices, which is then stored and processed in a central data center to train models. However, with the increasing number of connected devices, network congestion has led to a preference for processing more data at the edge [14], [15]. Edge computing involves processing data close to the data source [16], in our case, air quality sensing devices. Edge computing offloads computing tasks from the centralized cloud to the near-sensing devices and reduces transferred data by performing preprocessing functions at the edge [17]. Regarding memory cost, energy consumption, and latency, edge computing excels in various applications.

Air quality data collected from different air monitoring stations are spatially and temporally correlated [18]. Over a given period, the air quality of a particular area typically does not change significantly, and a similar variation in spatial dimensions can often be observed in adjacent areas [19]. Understanding spatial and temporal dependencies is essential [32], and we can make more informed decisions regarding air quality status by understanding these dependencies. Furthermore, collaborative learning using spatiotemporal data may improve model performance [19].

In this work, we investigate collaborative learning strategies for on-device training and inference. Specifically, our contributions are as follows:

1) We propose new approaches for air quality prediction based on spatiotemporal data and deep learning model sharing and compare them to the commonly implemented federated learning.
2) We develop algorithms for running collaborative learning directly on edge devices communicating using the MQTT protocol.

3) We evaluate various characteristics of the proposed collaborative learning strategies, including training losses, learning accuracy, learning period, and communication cost.
4) We extend this body of work by discussing some insights regarding the expansion of edge device networks.

II. Related Work

Existing methods for air quality predictions are mainly categorised into three: deterministic methods, statistical methods, and machine learning methods. Deterministic methods use mathematical equations and need high-speed calculation and simulation to predict atmospheric pollutant concentration [20], which is considered not viable for edge devices due to their limited energy capacity and computational resources [21]. Statistical methods implement statistical models by trying to find the relationship between influencing factors (meteorological data, spatiotemporal factors, and others) and air pollutants [22]. Many statistical models are based on linear assumptions that affect their prediction accuracy for commonly non-linear real problems. To overcome these problems, researchers implement non-linear machine learning approaches.

For example, Reid et al. [23] used ensemble machine learning models to forecast daily $PM_{2.5}$ levels. These models were trained using 24-hour $PM_{2.5}$ measurements gathered from monitoring stations situated across 11 states in the western United States. Input features were derived by temporally merging the data using the date and spatially merging the nearest spatial observations based on latitude and longitude. Although spatiotemporal factors were considered in this study, the training and prediction processes were centralized based on gathered data from the sensors.

Moreover, diverse types of machine learning models have been proposed for forecasting $PM_{2.5}$ and other pollutant levels in recent years. Promising results have been reported using Long Short-Term Memory (LSTM) models [24]–[27]. Additionally, combining Convolutional Neural Network (CNN) and LSTM models has received significant attention [28]–[30]. Li et al. [31] demonstrated the effectiveness of the CNN-LSTM approach using a one-dimensional CNN to extract features from sequence data, followed by an LSTM to predict future values.

Collaborative learning has emerged to address large-scale machine learning problems [33]. Henna et al. [34] utilized topological dependencies in mobile edge networks using a graph neural network (GNN) for efficient inference. The collaborative GNN-edge approach is trained using a compressed learning algorithm, outperforming cloud-based prediction. Song and Chai [35] proposed a collaborative learning framework for multi-class classification problems that reduces generalization errors and increases robustness to incorrect labels or data augmentation.

A widespread approach for collaborative learning is model averaging used in the federated learning (FL) approach [36]. FL eliminates the need to upload sensitive data to a central server [37], allowing edge devices to train a shared global model locally using their own data. To address poor convergence on heterogeneous data and a lack of generalization, personalized federated learning (PFL) has also been proposed [38].

Gholizadeh and Musilek showed how the convergence time of federated learning could be reduced with hyperparameter-based clustering [16] for predicting individual and aggregate electrical loads. Mi et al. [39] claimed that model averaging could be effectively implemented for distributed learning by using a cyclical learning rate and increasing the local training epochs. FL has also been implemented in various applications, such as cybersecurity for IoT [40], healthcare [41], and manufacturing [42]. In the case of air quality studies, it is possible to utilize FL as multiple monitoring stations can share their knowledge.

Our proposed approach differs from previous works as follows. Most previous work only investigates model development using large collected datasets trained centrally; our work implements collaborative learning directly on edge devices, offloading computation close to the data source. Furthermore, the current body of collaborative learning research has not specifically focused on the subject of air quality prediction, which is the subject of our work. We show that the primary benefit of collaborative learning lies in its capacity to enhance accuracy compared to individual learning. This approach is highly suited for real deployments of monitoring devices, eliminating the necessity for cloud access. We consider the availability of air pollution data from multiple neighboring observation stations, the spatial and temporal correlation between stations, and the possibility of improving air quality prediction through collaborative learning. We examine the effectiveness of the proposed collaborative learning methods, including losses during training, accuracy, and communication costs for each method alongside a common baseline.

III. Proposed Approach

A. Research Framework Overview

Our research framework is depicted in Fig. 1. Initially, a subset of air quality monitoring stations was chosen from the available dataset, considering device availability. The original dataset contained missing values, which were addressed by filling them with the most recent valid measurements. Subsequently, the dataset was divided into three divisions: training, validation, and test sets.

A feature selection process was conducted to determine the optimal number of input variables. The preprocessed datasets were then subjected to collaborative learning techniques, implemented directly at the edge utilizing the MQTT protocol. Three collaborative learning methods, namely *FedAvg*, *ClustME*, and *SpaTemp*, were employed in this study.

In addition to the collaborative learning approach, the locally trained model was also evaluated. Several evaluation metrics were employed, including Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), coefficient of determination (R^2), and improvement on RMSE (RIR). Furthermore, edge computing performance was specifically evaluated, particularly in terms of the time required for each device to complete the training session.

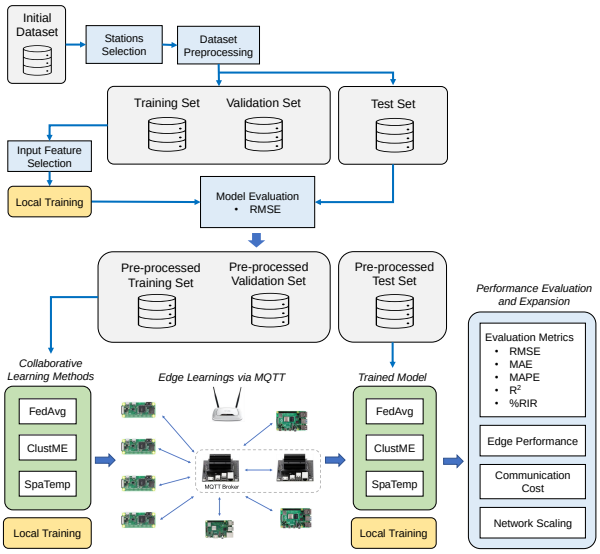

Fig. 1. Methodology framework for collaborative learning at the edge.

B. Air Quality Dataset

We used the Beijing air quality dataset provided by Zhang et al. [43], which can be obtained from the University of California, Irvine (UCI) Machine Learning Repository. The dataset is a collection of air quality data collected from 12 monitoring sites in Beijing and its surroundings, namely Aotizhongxin, Changping, Dingling, Dongsi, Guanyuan, Gucheng, Huairou, Nongzhanguan, Shunyi, Tiantan, Wanliu, and Wanshouxigong. The locations of all air quality monitoring sites can be shown in Fig.2. Each monitoring station has 12 columns and 36,064 rows, collected from 1 March 2013 to 28 February 2017. Each row record is hourly data, consisting of several pollutant data ($PM_{2.5}$, PM_{10}, SO_2, CO, NO_2, and O_3) and meteorological data (temperature, air pressure, dew point, rain, wind direction and wind speed). The wind direction is categorical data, admitting 16 values: N, NNE, NE, ENE, E, ESE, SE, SSE, S, SSW, SW, WSW, W, WNW, NW and NNW, which were converted to degrees. For missing values in the dataset, we filled them with the last timestamp data. All attributes were normalized to 0 and 1.

We arranged the stations alphabetically according to their names and chose the first eight stations for this study.

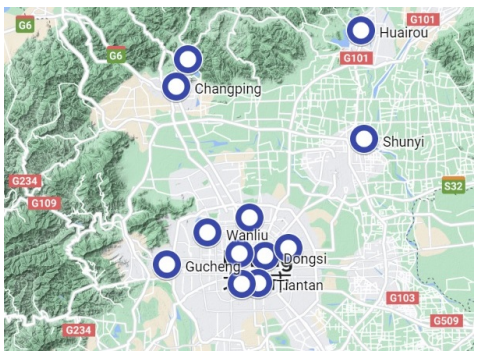

Fig. 2. Map of air quality monitoring stations in Beijing and its surroundings.

The selected stations are Aotizhongxin, Changping, Dingling, Dongsi, Guanyuan, Gucheng, Huairou, and Nongzhanguan. In this work, the selected stations are labelled as Station-01, Station-02, Station-03, and so on, up to Station-08. Finally, we used eight edge devices representing all monitoring stations.

The dataset was partitioned as follows: approximately 70% of the data was allocated for training, around 20% for validation during learning, and the remaining 10% for testing purposes. This work focuses on predicting $PM_{2.5}$ concentrations, evaluating the best model for short-term predictions, specifically the next one-hour $PM_{2.5}$ concentrations. Additionally, the work includes a discussion on predictions for longer time periods.

C. Collaborative Learning Overview

We implemented three collaborative learning strategies. As illustrated in Fig.3, these methods are federated learning using federated averaging, learning with clustered cyclic peer-to-peer model exchanges, and learning with spatiotemporal data exchanges. We refer to these three learning strategies as FedAvg, ClustME, and SpaTemp, respectively. In federated learning (FedAvg) and learning with cyclic peer-to-peer model exchange (ClustME), the edge devices transfer models between them, with no measurement data exposed to other devices. In learning with spatiotemporal data exchanges (SpaTemp), measurement data is exchanged.

FedAvg consists of four steps, as illustrated in Fig. 3(a). First, model initialization is conducted directly on the edge devices (stations). Then, each station performs model updates by training its model with local data. Once local training is complete, each station sends its local update to the coordinator. The coordinator collects all updated models from stations and performs aggregation. Finally, the coordinator transmits the newly aggregated model to all stations. The process repeats until the desired number of training rounds is achieved.

Fig. 3(b) shows the ClustME workflow. Time-series clustering is performed before local updates and model exchanges. Two clusters of air monitoring stations are created based on time-series similarity using K-means clustering. Each cluster follows the same process. Each station trains its model using local data and then shifts its trained model to another monitoring station. Each station also receives a trained model from another station, forming a circular exchange. The process repeats until all stations obtain all trained models from other stations.

SpaTemp is based on the strategy described in [44]. For each station, we selected the three most correlated nearby stations based on Pearson's correlation. First, a station requests data from the three most correlated stations. Second, the station serves other stations that request its local data. Finally, once these subscribe/publish processes are finished, each station trains locally using the shared data. Fig. 3(c) depicts the SpaTemp workflow.

D. Federated Learning (FedAvg)

We follow the Federated Average (FedAvg) algorithm proposed by McMahan et al. [36], which is broadly used as a

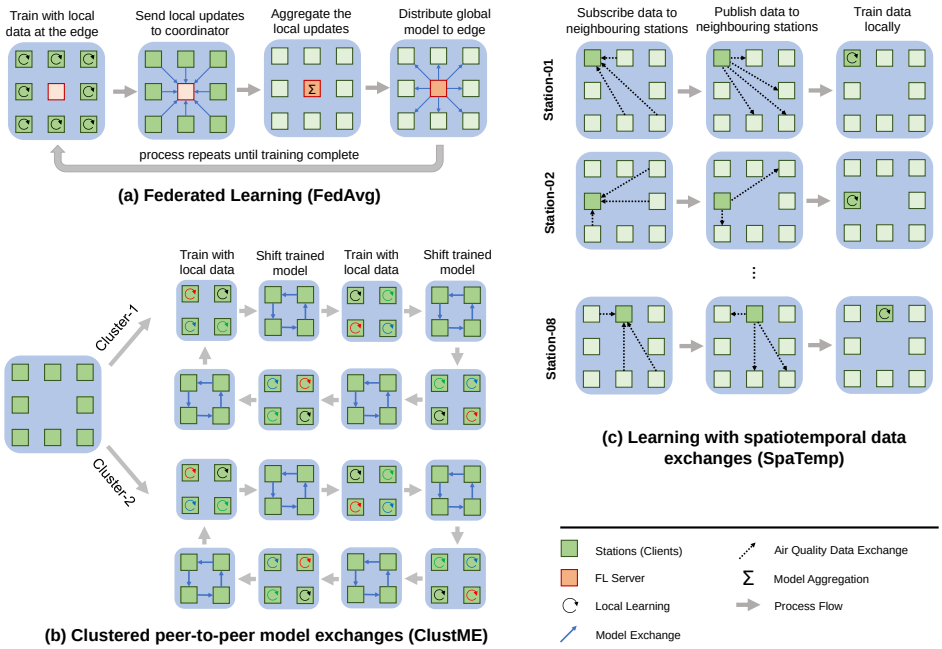

Fig. 3. Implemented collaborated learning strategies.

Algorithm 1 Modified Federated Averaging implemented in this work (FedAvg).

COORDINATOR EXECUTES:
  for each global round $t = 1, 2, \ldots, R$ **do**
    for each station $k \in K$ **in parallel do**
      $\theta_{t+1}^k \leftarrow$ **StationUpdate**(θ_t)
    $\theta_{t+1}^k \leftarrow \sum_{k=1}^{K} \frac{1}{K} \theta_{t+1}^k$

StationUpdate(θ) :
  $\mathcal{B} \leftarrow$ (split local dataset into batches of size $\mathcal{B}$)
  for each local epoch i from 1 to E **do**
    for each batch $b \in \mathcal{B}$ **do**
      $\theta \leftarrow \theta - \eta \nabla \mathcal{L}(\theta; b)$
  return θ to coordinator

standard FL setting. Algorithm 1 shows the FedAvg modified for this work. In this work, we have K edge devices that represent air quality monitoring stations, where $K = 8$. Each station has its local air quality data and trains its model locally rather than sending raw data to a centralized coordinator. The local dataset is split into batches with the size of $\mathcal{B}$ and is used to train the local model during the local epochs E. At round t, each station k trains its model locally. For each local epoch and batch, the station updates its respective model, that is $\theta_t^k \leftarrow \theta_t^k - \eta \nabla \mathcal{L}_k(\theta_t^k)$, where θ_t^k represents the model parameters, η is the learning rate, and $\mathcal{L}_k(\theta_t^k)$ is the loss function. Each device then transmits the parameter update, and the coordinator receives this update from each station as θ_{t+1}^k. The coordinator aggregates the parameters received from all participating stations, that is $\theta_{t+1}^k \leftarrow \sum_{k=1}^{K} \frac{1}{K} \theta_{t+1}^k$. The coordinator sends the aggregated results back to the stations, and the process repeats until R rounds are complete.

It is worth noting that all edge devices are involved in all FL rounds. Consequently, the coordinator does not sample stations in each round. The air quality datasets were trained at all stations using the same size, and the updated parameters received from stations were equally weighted during the aggregation step. To reduce the communication cost in the first round, we initialized the model's weight directly at the edge instead of sending the initial weights from the coordinator to edge devices.

The MQTT publish/subscribe topics manage the FL workflow on edge devices. The FL process path can be executed by determining the topics required at each step and selecting which device should publish/subscribe to a specific topic.

E. Clustered Peer-to-Peer Model Exchange (ClustME)

The locally trained models are circulated among the stations in this learning strategy. The model trained at a station will be passed to another station. As all stations perform the same procedures, a cyclic peer-to-peer model transfer is performed. To reduce the number of model transfers, we used a clustering technique. As our target is to predict the value of $PM_{2.5}$, we implement K-means clustering based on the series of $PM_{2.5}$ data. The clustering method allows many edge devices to be involved in collaborative learning by reducing the number of exchanges, and, hence, the total learning time.

The history of $PM_{2.5}$ data (the training data) from all stations is collected, and the clustering method is performed

TABLE I
CLUSTER OF STATIONS BASED ON TIME-SERIES OF $PM_{2.5}$ DATA

Cluster-1	Cluster-2
Station-01,-02,-03, and -07	Station-04,-05,-06, and -08

Algorithm 2 Clustered peer-to-peer model exchanges (ClustME).

for each cluster $c = 1, 2, \ldots$ **do**
  for each global round $t = 1, 2, \ldots, R$ **do**
    for each station $k = 1, 2 \ldots, K$ **in parallel do**
      $\theta_t^k \leftarrow$ **StationUpdate**(θ_t)
    for each station $k = 1, 2 \ldots, K$ **do**
      $\theta_{t+1}^k \leftarrow$ **ModelSharing**(θ_t^k)

StationUpdate(θ) :
  $\mathcal{B} \leftarrow$ (split local dataset into batches of size $\mathcal{B}$)
  for each local epoch i from 1 to E **do**
    for each batch $b \in \mathcal{B}$ **do**
      $\theta \leftarrow \theta - \eta \nabla \mathcal{L}(\theta; b)$
  return θ

ModelSharing(θ^k) :
  // if $k = K$, then send to $k = 1$
  Send θ^k to $k + 1$
  // if $k = 1$, then receive from $k = K$
  Receive θ^{k-1} from $k - 1$
  // update model
  $\theta^k \leftarrow \theta^{k-1}$
  return θ^k

using dynamic time warping as the metric for assignment and barycenter computations. We use the Tslearn Python package to determine the time-series cluster [45]. We divided the stations into two clusters, and Table I shows the clustering results. Station-01, Station-02, Station-03, and Station-07 belong to the same group, and other stations form another group. As each cluster consists of four stations, the model must be shifted three times after the local updates (see Fig. 3(b)) to complete one learning workflow.

Algorithm 2 shows the general workflow of ClustME performed in a station cluster. Instead of sending local updates to the coordinator, the locally trained model is sent to a target station. For example, in Cluster-1, Station-01 sends its updated model to Station-02, Station-02 sends its updated model to Station-03, Station-03 sends its updated model to Station-07, and Station-07 sends its updated model to Station-01. By performing these steps, a cyclic peer-to-peer model exchange is formed. There is no aggregating method conducted in this method. A station updates the received model with its local data and sends the updated model to a target station.

In this work, we assigned a coordinator to orchestrate the entire process. Each station has different speeds to complete the training rounds. Thus, a coordinator is required to manage data transmitted from one station to another.

F. Learning with Spatiotemporal Data Exchange (SpaTemp)

In SpaTemp, we send pollutant data instead of deep learning models to other stations. Each station combines local and nearby pollutant data to train its local model. With these data, each station trains the model locally without transmitting the updated model to other stations or the coordinator, as

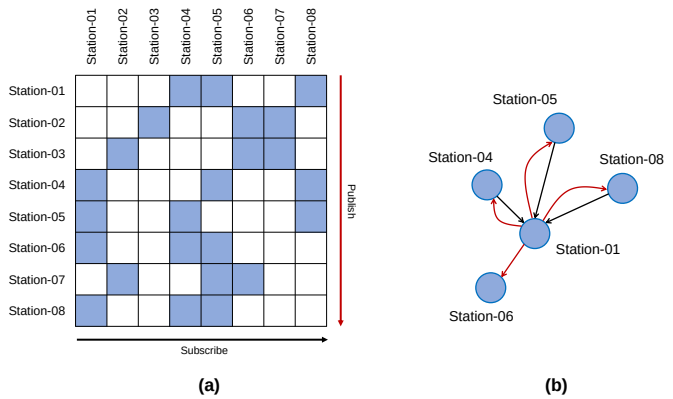

Fig. 4. (a) Subscribing and publishing data pairs performed in SpaTemp, and (b) An example of publishing and subscribing implemented at Station-01.

performed in FedAvg and ClustME. Our previous work [44] utilized spatiotemporal data to predict $PM_{2.5}$ concentrations. Based on the proposed model in our previous work, we adjusted parameters to reduce the number of filters in the convolutional layers. Each station collects data from the three most correlated nearby stations based on their $PM_{2.5}$ values.

Besides collecting data from the three most correlated nearby stations, each station also sends pollutant data to other stations that request it. We use the MQTT protocol concepts of publishing and subscribing to enable this. For example, if Station-01 wants to collect data from Station-04, Station-01 will subscribe to a specific topic that Station-04 publishes. Thus, data is transmitted from Station-04 to Station-01. After performing the Pearson correlation coefficient on all stations, we obtained the subscribing and publishing pairs, as shown in Fig. 4(a). Each station subscribes to the three most correlated nearby stations and publishes its data to one or more stations.

Algorithm 3 Learning with spatiotemporal data exchanges (SpaTemp).

STATION DATA COLLECTION:
  $\mathcal{M} \leftarrow$ (Collect $PM_{2.5}$ data from participating stations)
  $\mathcal{C} \leftarrow$ (Perform Pearson's correlation on $\mathcal{M}$)
  for station $k \in K$ **do**
    // create a list of nearby stations
    $\mathcal{S} \leftarrow$ (Select three significant $\mathcal{C}$ relative to k)
    $\mathcal{D} \leftarrow$ (Collect $PM_{2.5}$ from all members of $\mathcal{S}$)
    $\mathcal{H} \leftarrow$ (Combine $\mathcal{D}$ with station's local dataset)

for each global round $t = 1, 2, \ldots, R$ **do**
  for each station $k = 1, 2 \ldots, K$ **in parallel do**
    $\theta_{t+1}^k \leftarrow$ **StationUpdate**(θ_t)

StationUpdate(k, θ) :
  $\mathcal{B} \leftarrow$ (split local dataset into batches of size $\mathcal{B}$)
  for each local epoch i from 1 to E **do**
    for each batch $b \in \mathcal{B}$ **do**
      $\theta \leftarrow \theta - \eta \nabla \mathcal{L}(\theta; b)$
  return θ

For instance, Station-01 subscribes to data from Station-05, Station-04, and Station-08. Besides subscribing data, Station-01 also publishes data to Station-04, Station-05, Station-06, and Station-08, as shown in Fig. 4(b). The SpaTemp process workflow can be presented in Algorithm 3.

G. Deep Learning Models

Fig. 5 shows the proposed deep learning models used in this work. We built these models using the Tensorflow framework [46]. These models mainly consist of three different layers: one-dimensional convolutional (Conv1D) layers, long-term short memory (LSTM) layers, and fully connected (Dense) layers. The main properties of each layer (e.g., number of filters, kernel size and unit size) are shown in Fig.5. Except for the last layer, each layer uses the rectified linear unit (ReLU) activation function. No activation is applied for the output (Dense) layer, and other layer parameters follow the default properties set by the framework.

While FedAvg and ClustME utilize the same model architectures (Fig. 5(a)), SpaTemp uses a slightly different design (Fig. 5(b)). SpaTemp has a parallel structure of convolutional layers, and both paths share the same layer properties. In this architecture, the first path extracts the local data features, and the second path obtains the characteristics of shared spatiotemporal data. These spatiotemporal data are collected from three nearby air quality stations with the highest Pearson's correlation coefficients to the target station. In our case, we used time-series data for regression purposes. Thus, the air quality data sequence should be maintained to extract the correct information. In addition to retaining the time-series data sequence, SpaTemp also collects multiple data from other stations. Thus, we proposed a parallel structure previously mentioned to cover these requirements.

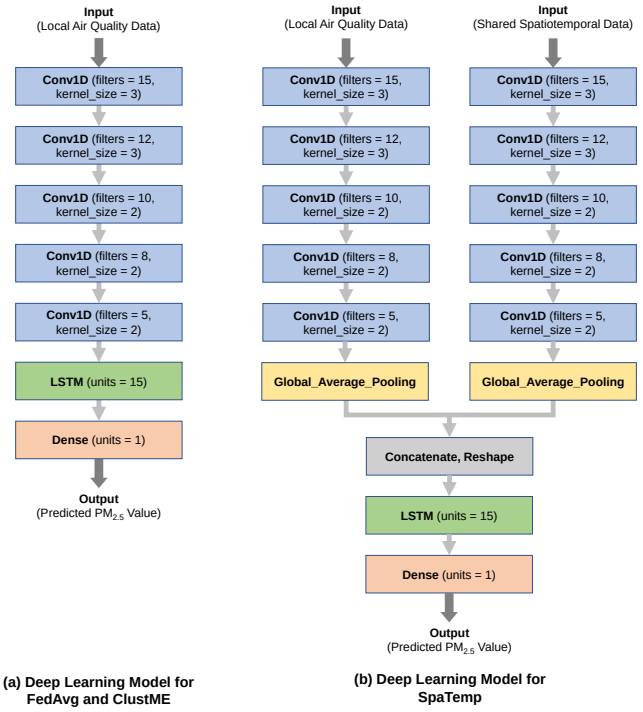

Fig. 5. Proposed deep learning model architectures.

All models process data for the current and past seven hours, making the length of the input equal to eight. As the selected dataset consists of 11 columns (i.e., $PM_{2.5}$, PM_{10}, SO_2, CO, NO_2, O_3, temperature, air pressure, dew point, wind direction and wind speed), the size of the input sets is 8×11. These input sets are used to train the model and predict the hourly values of $PM_{2.5}$. The deep learning model used in SpaTemp accepts not only local air quality data (the first path of the input model) but also the collection of $PM_{2.5}$ data from itself and the other three stations (the second path of the input model). Thus, the second path of the input model accepts a matrix with the size of 8×4.

H. Collaborative Learning Evaluation

The performance of the proposed collaborative learning approaches is evaluated across a number of metrics. The first step is input feature selection. The number of features is evaluated from the original dataset to find the optimum performance. Then, we visualize the losses during training to understand how well the model learns from training data. Next, we evaluate the model performance for each round against the unseen data (test data). The time required by each method to complete the learning task is evaluated, and the performance of each edge device is measured. Also, the communication costs that occurred while performing a learning strategy are assessed. Finally, we discuss the expansion of edge device networks.

The model performances on test data were evaluated using root mean squared error (RMSE), mean average error (MAE), mean absolute percentage error (MAPE), and coefficient of determination (R^2). Their definitions are given in Equations (1), (2), (3) and (4):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2} \tag{1}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y}_i| \tag{2}$$

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{3}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4}$$

where n is the total number of data samples, y_i are the actual values, $\hat{y}_i$ are the predicted values, and $\bar{y}$ is the overall mean of the actual values.

Additionally, to measure the improvement of the proposed collaborative learning strategies compared with the locally learned method, we implemented a metric called rate of improvement on RMSE (RIR) using the following equation [2], [47]:

$$RIR^{L,C} = \frac{RMSE^L - RMSE^C}{RMSE^L} \times 100\% \tag{5}$$

where, $RMSE^L$ denotes the RMSE obtained using the locally-learned method, and $RMSE^C$ is the RMSE obtained using the proposed collaborative learning.

Pearson's correlation coefficient was employed to ascertain the relationships among features within the dataset and to identify neighboring stations for the SpaTemp method. Pearson's correlation coefficient between two series is described as follows [2], [44]:

$$r(\boldsymbol{S}^t, \boldsymbol{S}^s) = \frac{\sum((s_i^t - \mu_t)(s_i^s - \mu_s))}{\sqrt{\sum(s_i^t - \mu_t)^2 \sum(s_i^s - \mu_s)^2}} \quad (6)$$

where, $r(\boldsymbol{S}^t, \boldsymbol{S}^s)$ denotes the Pearson's correlation coefficient between the time series $\boldsymbol{S}^t$ and $\boldsymbol{S}^s$, s_i^t and s_i^s represent the i-th samples of $\boldsymbol{S}^t$ and $\boldsymbol{S}^s$, respectively. Finally, $\mu t = \frac{1}{n}\sum_{i=1}^{N} s_i^t$ and $\mu s = \frac{1}{n}\sum_{i=1}^{N} s_i^s$ denote the mean values of time series $\boldsymbol{S}^t$ and $\boldsymbol{S}^s$, respectively.

I. Application Scenario

As shown in Fig 6, edge devices used for the experiment represent eight air quality monitoring stations. Three variants of Raspberry Pi (RPi) boards are employed for Station-01 to Station-07, and an NVIDIA Jetson Nano 2GB Developer Kit is used as both Station-08 and the coordinator.

The Edge devices' network communication is based on the MQTT (Message Queuing Telemetry Transport) protocol. This protocol implements a client-server architecture, performs message transmission based on topic publishing and subscribing, and works over TCP/IP [48]. The publish/subscribe protocol provides a scalable and reliable way to connect resource-constrained edge devices over the Internet or within a local area network (LAN). Moreover, the MQTT protocol is simple and lightweight, designed to transmit information over unreliable networks with low bandwidth or high latency [49].

Stations can act as both publishers and subscribers. The messages can be air quality measurement data or deep learning models in this work. Stations receive messages from other stations by subscribing to topics of interest to the server (MQTT broker). A wireless router is used to create a local area network, and the devices communicate with the broker via MQTT.

All boards used run off a 5V power supply, with varying current usage from 1.2–3A. Rpi boards use Raspberry Pi OS Lite version 10 (Debian Buster), whereas the Jetson Nano uses NVIDIA JetPack 4.6 with Ubuntu 18.04. In this work, all boards operate in headless mode.

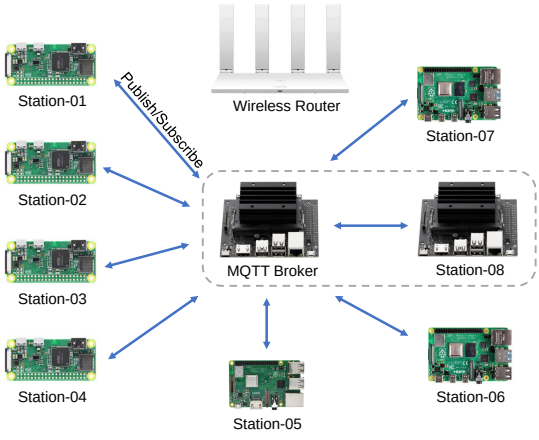

Fig. 6. Application scenario of the edge devices. The devices communicate with the server via the MQTT protocol within a local area network (LAN). Station-08 also acts as an MQTT broker.

IV. Results and Discussion

A. Feature Selection

This study aims to predict $PM_{2.5}$ levels and builds upon our prior research [44] in selecting input features. We calculated Pearson's correlation coefficients among features and determined the overall correlation by averaging the coefficients across all monitoring stations. The averaged correlation coefficients are reported in Fig. 7.

$PM_{2.5}$ levels exhibit strong positive correlations with PM_{10}, NO_2, and CO ($r > 0.6$), moderate positive correlation with SO_2 ($r = 0.47$), and weak negative correlation with O_3 ($r = -0.13$). Rainfall (RAIN), dew point (DEWP), air pressure (PRES), and air temperature (TEMP) display the weakest correlations with $PM_{2.5}$. We evaluated the absence of these four features to determine the optimal input set for the deep learning model by training the model locally without utilizing collaborative learning. The model architecture used for training is illustrated in Fig. 5(a).

The presence of weak correlation coefficients implies that there is minimal redundancy or information duplication among these variables, rendering them suitable for direct utilization as inputs for our prediction model [50]. It is important to note that this does not diminish the importance of rainfall and other meteorological factors. Instead, given our focus on resource-constrained devices with limited computational power, we aim to identify the most efficient input features.

Table II shows that the best performance is achieved by removing the rain feature during training, resulting in the selection of 11 attributes: $PM_{2.5}$, PM_{10}, SO_2, CO, NO_2, O_3,

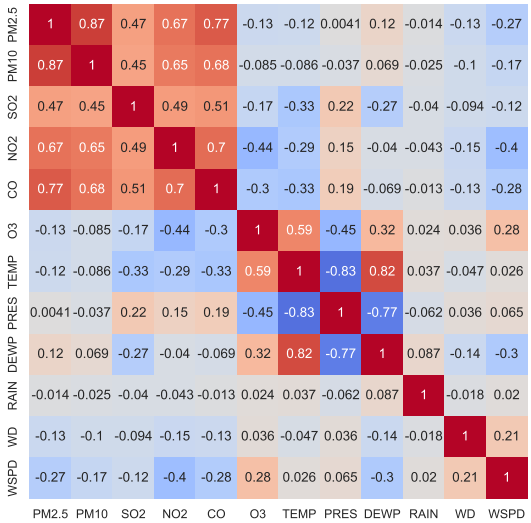

Fig. 7. Average correlation coefficients among features.

TABLE II
FEATURE SELECTION RESULTS BASED ON THE AVERAGE OF RMSE AT ALL STATIONS

Input Feature	# inputs	Avg. RMSE
All	12	23.126
Without PRES	11	23.370
Without RAIN	11	**23.018**
Without PRESS and RAIN	10	23.316
Without PRESS, RAIN, and DEWP	9	23.424
Without PRESS, RAIN, and TEMP	9	23.270
Without PRESS, RAIN, DEWP, and TEMP	8	23.526

temperature, air pressure, dew point, wind direction and wind speed as model input features.

B. Losses During Collaborative Training

Losses measure how well a predictor can map the relationship between inputs and the provided targets. The better the predictor, the smaller the loss. While the training losses are calculated during training, the validation losses are measured at the end of each epoch. The training loss indicates how well the model fits the training data, whereas validation loss is evaluated on validation data. This work measures training and validation losses using mean squared error (MSE). Examples of training and validation losses evaluated at Station-06 are shown in Fig. 8(a) and 8(b), respectively.

We initialized the proposed deep learning models with the same random seeds for all methods to provide reproducible model outputs and less biased results. We then evaluated training performance. Consequently, FedAvg and ClustME yield the same training and validation losses during the first round as these methods use the same model architecture, training and validation data. Then, due to different learning approaches, the variation of losses between FedAvg and ClustME occurs after the first round. Fig. 9 illustrates how training losses vary after the first round (when there is a significant reduction in losses for all approaches).

Fig. 9 shows that learning by utilizing spatiotemporal data (SpaTemp) outperforms other methods in minimizing loss during training at all participating stations. The collaborative pollutant data sharing among stations effectively improves model performance. As mentioned previously, SpaTemp does not involve model sharing or aggregation processes to gain knowledge from other stations but includes raw measurement data as additional inputs.

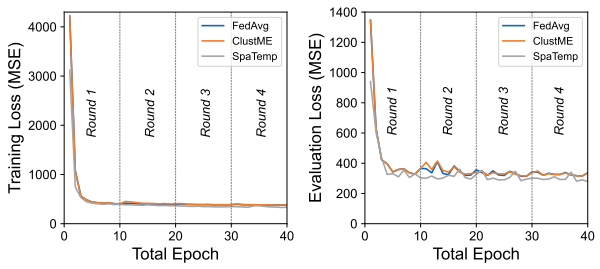

Fig. 8. Examples of (a) training loss and (b) validation for Station-06.

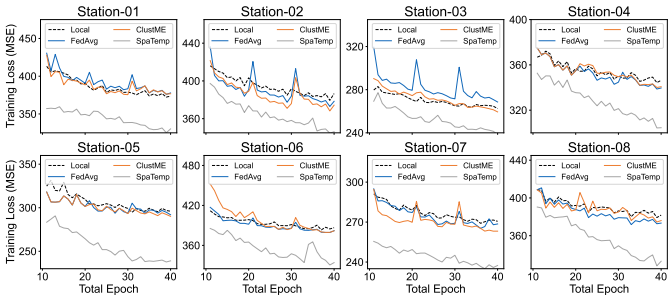

Fig. 9. Better presentation of training losses at all stations after the first round and local training performances.

C. $PM_{2.5}$ Prediction on Test Data

We reserve about 10% of the dataset for use as test data. Table III shows the model performance evaluated using RMSE, MAE, MAPE, R^2, and RIR for the first four stations. This table also includes the performance of locally-learned models without collaborative strategies. A locally-learned model means that the model is trained using its local data only and excludes aggregation or model-sharing as performed by FedAvg and ClustME. Also, the model gains knowledge from its local data without collecting pollutant data from other stations, as implemented in SpaTemp. In conclusion, no data is leaving the station during training.

SpaTemp outperforms other methods in predicting unseen data for all monitoring stations, displayed in bold in Table III. Moreover, based on RMSE, the RIR can be calculated, and the results are reported in the table. A positive RIR value indicates an improvement in the proposed collaborative learning method against the locally learned approach. SpaTemp outperforms other collaborative learning methods, with RIR

TABLE III
MODEL PERFORMANCE IN PREDICTING $PM_{2.5}$ ON TEST DATA FOR THE FIRST FOUR STATIONS.

	Station-01	Station-02	Station-03	Station-04
RMSE(μg/m^3)				
FedAvg	21.743	23.135	21.097	25.118
ClustME	21.978	23.345	20.772	25.021
SpaTemp	**19.917**	**22.140**	**20.738**	**24.299**
Local	21.871	23.744	20.890	25.170
MAE(μg/m^3)				
FedAvg	12.008	13.182	10.981	12.916
ClustME	12.028	13.263	10.910	12.863
SpaTemp	**10.729**	**12.173**	**10.741**	**12.395**
Local	11.967	13.708	10.967	12.965
MAPE(%)				
FedAvg	26.881	28.441	30.090	37.727
ClustME	27.351	28.411	31.288	37.299
SpaTemp	**26.239**	**27.594**	**28.645**	**36.817**
Local	26.347	28.747	30.768	37.996
R^2				
FedAvg	0.958	0.935	0.950	0.953
ClustME	0.957	0.934	0.951	0.953
SpaTemp	**0.964**	**0.941**	**0.952**	**0.956**
Local	0.957	0.932	0.951	0.952
RIR(%)				
FedAvg	0.588	2.564	-0.995	0.208
ClustME	-0.488	1.681	0.565	0.593
SpaTemp	**8.934**	**6.756**	**0.725**	**3.461**
Local (baseline)	0.000	0.000	0.000	0.000

ranging from 0.525% to 8.934%. The most significant improvement is recorded at Station-01. Nevertheless, not all collaborative learning strategies performed better than the locally-learned method. ClustME degrades model performance slightly at Station-01, Station-05, and Station-08, indicated by the minus values of RIR. These degradations vary from 0.092% to 0.488%. The federated learning approach degrades performance by 0.995% and 0.636% at Station-03 and Station-05, respectively.

Fig. 10 provides a more intuitive representation of model performance, specifically at Station-02. The plots provide a quick assessment of the proximity between the predicted and observed values using diagonal lines as a reference. The SpaTemp model performs better in capturing extreme $PM_{2.5}$ levels, resulting in a higher R^2 score of 0.941. The figure illustrates that the predicted values closely align with the observed values, particularly in capturing very high concentrations of $PM_{2.5}$ compared to other methods.

It is also feasible to estimate longer time periods, with 3-hour, 6-hour, 9-hour, and 12-hour period results presented in Fig. 11 for Station-05. The figure indicates a decline in model performance as the prediction period extends. Evaluation based on R^2 scores reveals that the model's accuracy ranges from approximately 0.8 for 3-hour predictions to about 0.4 for 12-hour predictions. Moreover, the SpaTemp approach demonstrates better performance compared to the locally-trained model.

D. Learning Execution Time

We also consider the time taken to complete the collaborative learning strategies. We executed the approaches four times and averaged the period to complete training. Federated Learning (FedAvg) completed training faster than other methods (about 49 minutes). In contrast, SpaTemp took about 61 minutes. SpaTemp utilizes a larger deep learning model and

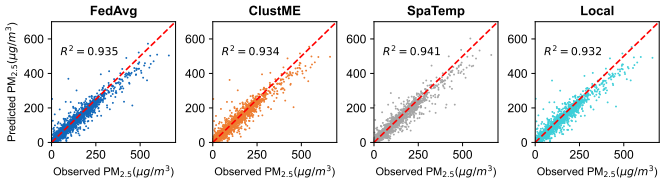

Fig. 10. Comparison between the observed and predicted of hourly $PM_{2.5}$ values at Station-02

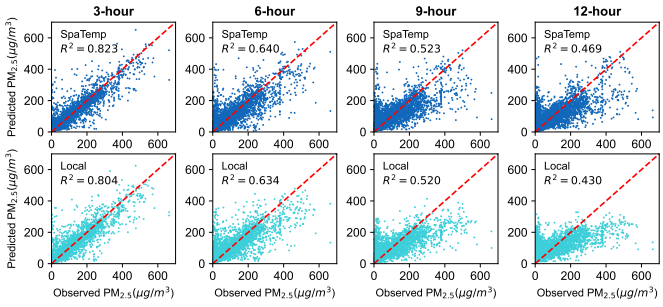

Fig. 11. Longer prediction hours of $PM_{2.5}$ evaluated at Station-05

TABLE IV
AVERAGE TIME TO COMPLETE THE COLLABORATIVE TRAINING

	FedAvg	ClustME (Cluster-1)	ClustME (Cluster-2)	SpaTemp
Time(s)	2954.40	3220.62	2979.62	3682.09

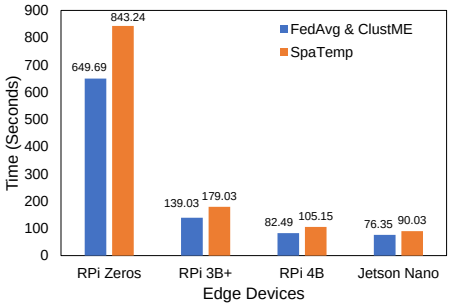

Fig. 12. Average time for edge devices to complete training rounds.

input data, resulting in slower epoch completion than other methods. In Table IV, we report the training time using the slowest edge device as SpaTemp essentially trained the model locally without model-sharing.

For ClustME, we trained two different clusters. The first cluster (Station-01,-02,-03, and -07) consists of slower edge devices, the Raspberry Pi Zeros. In another cluster (Station-04,-05,-06, and -08), the completion times are about 240 seconds faster than the first cluster, as it mainly consists of faster devices, such as Raspberry Pi 3B+, Raspberry Pi 4B, and Jetson Nano.

We also reported the average time each edge device needs to complete the training steps. After performing 40 rounds, we obtained the average times shown in Fig 12. As we used multiple Raspberry Pi (RPi) 4 and Zeros, we averaged the execution time for device types. The figure shows that the Jetson Nano 2GB developer kit outperforms other devices, being up to nine times faster than Raspberry Pi Zero when executing SpaTemp. The slower devices contribute to longer completion times of collaborative learning strategies.

E. Communication Cost Considerations

We consider communication costs based on the transferred payload contained in MQTT topics. During learning, FedAvg and ClustME exchange models, while SpaTemp exchanges pollutant data during data collection. As the devices involved in collaborative learning have different speeds, we utilize some governing MQTT topics with zero-length payloads to synchronize. For example, the coordinator uses a governing topic to start local updates or instruct participating devices to send their models. We exclude these topics from the communication cost measurement.

In this work, the initial model size for FedAvg and ClustME is about 44 kB and for SpaTemp about 77 kB. All models are compiled with Adam optimizer. After completing one round of training, the model file sizes increase, as the Adam optimizer maintains important gradient states during training: to 91 kB for FedAvg and ClustME and 149 kB for SpaTemp. The deep

TABLE V
COMMUNICATION COST OF COLLABORATIVE LEARNING.

	FedAvg	ClustME	SpaTemp
Comm. cost (MB)	11.648	5.824	8.415

learning models for FedAvg and ClustME are transmitted as byte array objects. However, for SpaTemp, pollutant data is represented using a 5-byte UTF-8 text string [51].

Table V shows the amount of data transferred for both incoming and outcoming payloads at all participating stations for the three methods. In FedAvg, the coordinator receives models from participants, aggregates them and transfers them back to participants. In ClustME and SpaTemp, the coordinators only function as governing devices. As a result, FedAvg consumes significantly more data communication, while ClustME reduces communication costs by half. SpaTemp transmits data much more frequently, up to 35,064 rounds during data collection, resulting in a communication cost between FedAvg and ClustME.

F. Network Scaling

We now consider how these approaches deal with an increased number of stations, which would increase the amount of data exchanged during learning. For FedAvg and ClustME, the amount of data transmitted during learning can increase as the number of rounds increases. In SpaTemp, however, the amount of data sent increases with the increasing size of hourly pollutant data, regardless of the number of learning rounds.

For FedAvg, the coordinator sends θ_0 (initialization weights or model) to all stations. Each station k trains on its own local data and sends the result to the coordinator. The coordinator aggregates all collected θ_t^k and returns the updated global parameters θ_{t+1}^k to all stations. The number of data exchanges can be expressed as follows:

$$D_{sta} = N\theta_{(0)} + \sum_{i=1}^{T} C_i K \theta_{i(tx)} + \sum_{i=1}^{T} C_i K \theta_{i(rx)} \quad (7)$$

where D_{sta} is the number of data exchanges on the station's side, K is the total number of stations, N is the number of participating stations, C is the fraction of stations participating in each round, T is the number of FL rounds and θ_i is the amount of information exchanged per round.

The amount of data transmitted to the coordinator ($\theta_{i(tx)}$) is the same as the amount of received data by the station ($\theta_{i(rx)}$). In our work, we force all stations to be involved in FL rounds ($N = K$ and $C = 1$). Therefore, the number of data exchanges becomes:

$$D_{sta} = K\left(\theta_0 + \sum_{i=1}^{T} 2\theta_i\right) \quad (8)$$

On the coordinator's side, the amount of data exchanged is the same as on the station's side. Also, in this work, the initial model is generated on the station side ($\theta_0 = 0$), and every round consists of the same number of epochs (i.e., the model size is the same at every round). Thus, the total communication cost between stations and the coordinator can be expressed as:

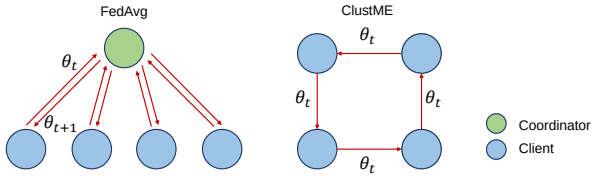

Fig. 13. Comparison of model exchanges between FedAvg and ClustME with the same number of participating stations.

$$D_{FedAvg} = 4KT\theta \quad (9)$$

ClustME does not require the stations to transmit data to the coordinator, as the coordinator is only responsible for orchestrating the workflow. Thus, the payloads are transferred among stations, as shown in Fig. 13.

A station receives and transmits the same amount of data. Despite having the same number of participants, the communication cost in ClustME is half that of FedAvg. Equation 10 expresses the communication cost for ClustME, considering the total number of participating stations without explicitly mentioning the communication cost in each cluster.

$$D_{ClustME} = 2KT\theta \quad (10)$$

In SpaTemp, the information transferred to a target station is measurement data (not deep learning models), specifically $PM_{2.5}$. The communication cost depends on the number of nearby stations sending their data to the target stations and the total required hourly data. In this work, we selected three nearby stations. The data exchange can be indicated as the coloured squares in Fig.3(a), where subscribe-publish pairs exist. Finally, the communication cost in SpaTemp can be expressed as follows:

$$D_{SpaTemp} = 2KN_{neigh}H\varphi \quad (11)$$

where K is total number of stations, N_{neigh} is the number of participating nearby stations, H is the number of hourly data, and φ the amount of information exchanged during learning.

As shown in Fig. 14, the communication costs for FedAvg and ClustME increase linearly with the number of rounds

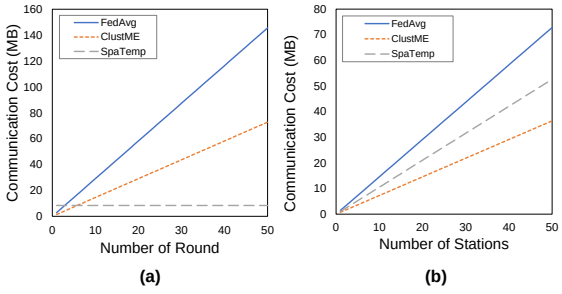

Fig. 14. The amount of communication cost calculated by changing (a) the number of rounds, and (b) the number of stations.

(Fig. 14(a)), while the communication cost for SpaTemp remains constant. Increasing the number of participating stations affects the communication cost for all methods, as shown in Fig. 14(b).

V. Conclusions and Future Work

Air quality data collected from various monitoring stations exhibits spatial and temporal correlations, suggesting that collaborative learning could enhance the performance of deep learning models. Unfortunately, collaborative learning has not been applied to air quality prediction specifically. In this study, we have compared three collaborative learning methods for predicting air pollutant levels: federated learning, learning with clustered model exchange, and our proposed learning with spatiotemporal data exchanges. We show that the latter two approaches complement the well-established federated learning approach for achieving collaborative learning on edge devices. The effectiveness of spatiotemporal data exchanges during training results in a significant 8.934% improvement in RMSE compared to locally trained models. However, it is worth noting that it incurs slower training times due to its larger model size, approximately 700 seconds slower than federated learning when tested on edge devices.

The communication costs associated with these methods may depending on several factors, including the number of participating stations, rounds, and dataset size as we have explored. We assessed the impact of increasing the number of edge devices and increasing the number of rounds and dataset size for each method. Learning with model sharing demonstrated the lowest communication costs, approximately two times lower than federated learning and 1.4 times lower than spatiotemporal learning.

We utilized single-board computers to validate this work. Our future work will explore this collaborative learning framework on more varied edge devices including those with more constrained computational resources as well as exploring the role of edge offloading in improving performance.

Acknowledgments

This work was supported in part by the Indonesia Endowment Fund for Education (LPDP), Ministry of Finance, Republic of Indonesia, under grant number Ref: S-1027/LPDP.4/2019.

References

[1] S. Ameer et al., "Comparative Analysis of Machine Learning Techniques for Predicting Air Quality in Smart Cities," IEEE Access, vol. 7, pp. 128325–128338, 2019.

[2] I. N. K. Wardana, J. W. Gardner, and S. A. Fahmy, "Estimation of missing air pollutant data using a spatiotemporal convolutional autoencoder," Neural Comput & Applic, vol. 34, no. 18, pp. 16129–16154, Sep. 2022.

[3] T. Xue, T. Zhu, Y. Zheng, and Q. Zhang, "Declines in mental health associated with air pollution and temperature variability in China," Nat Commun, vol. 10, no. 1, Art. no. 1, May 2019.

[4] E. Lanzi, R. Dellink, and J. Chateau, "The sectoral and regional economic consequences of outdoor air pollution to 2060," Energy Economics, vol. 71, pp. 89–113, Mar. 2018.

[5] J. M. Delgado-Saborit, V. Guercio, A. M. Gowers, G. Shaddick, N. C. Fox, and S. Love, "A critical review of the epidemiological evidence of effects of air pollution on dementia, cognitive function and cognitive decline in adult population," Science of The Total Environment, vol. 757, p. 143734, Feb. 2021.

[6] Q. Chen, Q. Wang, B. Xu, Y. Xu, Z. Ding, and H. Sun, "Air pollution and cardiovascular mortality in Nanjing, China: Evidence highlighting the roles of cumulative exposure and mortality displacement," Chemosphere, vol. 265, p. 129035, Feb. 2021.

[7] Y. Guo et al., "The association between lung cancer incidence and ambient air pollution in China: A spatiotemporal analysis," Environmental Research, vol. 144, pp. 60–65, Jan. 2016.

[8] A. R. Alsaber, J. Pan, and A. Al-Hurban, "Handling Complex Missing Data Using Random Forest Approach for an Air Quality Monitoring Dataset: A Case Study of Kuwait Environmental Data (2012 to 2018)," International Journal of Environmental Research and Public Health, vol. 18, no. 3, Art. no. 3, Jan. 2021.

[9] A. Dairi, F. Harrou, S. Khadraoui, and Y. Sun, "Integrated Multiple Directed Attention-Based Deep Learning for Improved Air Pollution Forecasting," IEEE Transactions on Instrumentation and Measurement, vol. 70, pp. 1–15, 2021.

[10] M. A. Fekih et al., "Participatory Air Quality and Urban Heat Islands Monitoring System," IEEE Transactions on Instrumentation and Measurement, vol. 70, pp. 1–14, 2021.

[11] A. C. Rai et al., "End-user perspective of low-cost sensors for outdoor air pollution monitoring," Science of The Total Environment, vol. 607–608, pp. 691–705, Dec. 2017.

[12] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated Machine Learning: Survey, Multi-Level Classification, Desirable Criteria and Future Directions in Communication and Networking Systems," IEEE Communications Surveys & Tutorials, vol. 23, no. 2, pp. 1342–1397, 2021.

[13] X. Dai, B. Zhang, X. Jiang, L. Liu, D. Fang, and Z. Long, "Has the Three-Year Action Plan improved the air quality in the Fenwei Plain of China? Assessment based on a machine learning technique," Atmospheric Environment, vol. 286, p. 119204, Oct. 2022.

[14] S. Douch, M. R. Abid, K. Zine-Dine, D. Bouzidi, and D. Benhaddou, "Edge Computing Technology Enablers: A Systematic Lecture Study," IEEE Access, vol. 10, pp. 69264–69302, 2022.

[15] M. Ashouri, P. Davidsson, and R. Spalazzese, "Quality attributes in edge computing for the Internet of Things: A systematic mapping study," Internet of Things, vol. 13, p. 100346, Mar. 2021.

[16] P. Velentzas, M. Vassilakopoulos, and A. Corral, "GPU-aided edge computing for processing the k nearest-neighbor query on SSD-resident data," Internet of Things, vol. 15, p. 100428, Sep. 2021.

[17] H. Li, K. Ota, and M. Dong, "Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing," IEEE Network, vol. 32, no. 1, Art. no. 1, Jan. 2018.

[18] S. Abirami and P. Chitra, "Regional air quality forecasting using spatiotemporal deep learning," Journal of Cleaner Production, vol. 283, p. 125341, Feb. 2021.

[19] G. Zhao, G. Huang, H. He, H. He, and J. Ren, "Regional Spatiotemporal Collaborative Prediction Model for Air Quality," IEEE Access, vol. 7, pp. 134903–134919.

[20] G. Zhou et al., "Numerical air quality forecasting over eastern China: An operational application of WRF-Chem", Atmos. Environ., vol. 153, pp. 94–108, Mar. 2017.

[21] J. Liu, K. Luo, Z. Zhou, and X. Chen, 'ERP: Edge resource pooling for data stream mobile computing', IEEE Internet Things J., vol. 6, no. 3, pp. 4355–4368, Jun. 2019.

[22] H. Liu et al., 'An intelligent hybrid model for air pollutant concentrations forecasting: Case of Beijing in China', Sustain. Cities Soc., vol. 47, no. 101471, p. 101471, May 2019.

[23] C. E. Reid, E. M. Considine, M. M. Maestas, and G. Li, "Daily PM2.5 concentration estimates by county, ZIP code, and census tract in 11 western states 2008–2018," Sci Data, vol. 8, no. 1, Art. no. 1, Apr. 2021.

[24] R. Navares and J. L. Aznarte, "Predicting air quality with deep learning LSTM: Towards comprehensive models," Ecological Informatics, vol. 55, p. 101019, Oct. 2019.

[25] X. Li et al., "Long short-term memory neural network for air pollutant concentration predictions: Method development and evaluation," Environmental Pollution, vol. 231, pp. 997–1004, Dec. 2017.

[26] T. Xayasouk, H. Lee, and G. Lee, "Air Pollution Prediction Using Long Short-Term Memory (LSTM) and Deep Autoencoder (DAE) Models," Sustainability, vol. 12, no. 6, Art. no. 6, Mar. 2020.

[27] D. Seng, Q. Zhang, X. Zhang, G. Chen, and X. Chen, "Spatiotemporal prediction of air quality based on LSTM neural network," Alexandria Engineering Journal, vol. 60, no. 2, Art. no. 2, Apr. 2021.

[28] Y. Qi, Q. Li, H. Karimian, and D. Liu, "A hybrid model for spatiotemporal forecasting of PM2.5 based on graph convolutional neural network and long short-term memory," Science of The Total Environment, vol. 664, pp. 1–10, May 2019.

[29] Q. Zhang, J. C. Lam, V. O. Li, and Y. Han, "Deep-AIR: A Hybrid CNN-LSTM Framework forFine-Grained Air Pollution Forecast," arXiv:2001.11957 [eess], Jan. 2020, Accessed: May 28, 2020. [Online]. Available: http://arxiv.org/abs/2001.11957
[30] D. Qin, J. Yu, G. Zou, R. Yong, Q. Zhao, and B. Zhang, "A Novel Combined Prediction Scheme Based on CNN and LSTM for Urban PM2.5 Concentration," IEEE Access, vol. 7, pp. 20050–20059, 2019.
[31] T. Li, M. Hua, and X. Wu, "A Hybrid CNN-LSTM Model for Forecasting Particulate Matter (PM2.5)," IEEE Access, vol. 8, pp. 26933–26940, 2020.
[32] F. Amato, F. Guignard, S. Robert, and M. Kanevski, "A novel framework for spatio-temporal prediction of environmental data using deep learning," Sci Rep, vol. 10, no. 1, p. 22243, Dec. 2020.
[33] X. Deng, T. Sun, F. Liu, and D. Li, "SignGD with error feedback meets lazily aggregated technique: Communication-efficient algorithms for distributed learning," Tsinghua Science and Technology, vol. 27, no. 1, pp. 174–185, Feb. 2022.
[34] S. Henna and A. Davy, "Distributed and Collaborative High-Speed Inference Deep Learning for Mobile Edge with Topological Dependencies," IEEE Transactions on Cloud Computing, vol. 10, no. 2, pp. 821–834, Apr. 2022.
[35] G. Song and W. Chai, "Collaborative Learning for Deep Neural Networks," in Advances in Neural Information Processing Systems, 2018, vol. 31. Accessed: Oct. 02, 2022.
[36] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Apr. 2017, pp. 1273–1282. Accessed: Aug. 22, 2022.
[37] J. Mills, J. Hu, and G. Min, "Multi-Task Federated Learning for Personalised Deep Neural Networks in Edge Computing," IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 3, pp. 630–641, Mar. 2022.
[38] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards Personalized Federated Learning," IEEE Transactions on Neural Networks and Learning Systems, pp. 1–17, 2022.
[39] H. Mi et al., "Collaborative deep learning across multiple data centers," Sci. China Inf. Sci., vol. 63, no. 8, p. 182102, Aug. 2020.
[40] B. Ghimire and D. B. Rawat, "Recent Advances on Federated Learning for Cybersecurity and Cybersecurity for Federated Learning for Internet of Things," IEEE Internet of Things Journal, vol. 9, no. 11, pp. 8229–8249, Jun. 2022.
[41] R. Kumar et al., "Blockchain-Federated-Learning and Deep Learning Models for COVID-19 Detection Using CT Imaging," IEEE Sensors Journal, vol. 21, no. 14, pp. 16301–16314, Jul. 2021.
[42] K. I.-K. Wang, X. Zhou, W. Liang, Z. Yan, and J. She, "Federated Transfer Learning Based Cross-Domain Prediction for Smart Manufacturing," IEEE Transactions on Industrial Informatics, vol. 18, no. 6, pp. 4088–4096, Jun. 2022.
[43] S. Zhang, B. Guo, A. Dong, J. He, Z. Xu, and S. X. Chen, "Cautionary tales on air-quality improvement in Beijing," Proc. R. Soc. A, vol. 473, no. 2205, Art. no. 2205, Sep. 2017.
[44] I. N. K. Wardana, J. W. Gardner, and S. A. Fahmy, "Optimising Deep Learning at the Edge for Accurate Hourly Air Quality Prediction," Sensors, vol. 21, no. 4, p. 1064, Feb. 2021.
[45] R. Tavenard et al., "Tslearn, A Machine Learning Toolkit for Time Series Data," Journal of Machine Learning Research, vol. 21, no. 118, pp. 1–6, 2020.
[46] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems." arXiv, Mar. 16, 2016. Accessed: Aug. 31, 2022. [Online]. Available: http://arxiv.org/abs/1603.04467
[47] J. Ma et al., "Transfer learning for long-interval consecutive missing values imputation without external features in air pollution time series," Advanced Engineering Informatics, vol. 44, p. 101092, Apr. 2020.
[48] X. Liu, T. Zhang, N. Hu, P. Zhang, and Y. Zhang, "The method of Internet of Things access and network communication based on MQTT," Computer Communications, vol. 153, pp. 169–176, Mar. 2020.
[49] A. Velinov, A. Mileva, S. Wendzel, and W. Mazurczyk, "Covert Channels in the MQTT-Based Internet of Things," IEEE Access, vol. 7, pp. 161899–161915, 2019.
[50] Q. Tao, F. Liu, Y. Li, and D. Sidorov, 'Air pollution forecasting using a deep learning model based on 1D convnets and bidirectional GRU', IEEE Access, vol. 7, pp. 76690–76698, 2019.
[51] T. A. Hilal and H. A. Hilal, "Turkish Text Compression via Characters Encoding," Procedia Computer Science, vol. 175, pp. 286–291, 2020.

I Nyoman Kusuma Wardana (Graduate Student Member, IEEE) received the B.Eng. degree in engineering physics from the University of Gadjah Mada, Indonesia, in 2008, and the M.Sc. degree in electronic engineering from Politecnico di Torino, Italy, in 2012. Since 2015 he has been a lecturer of electrical engineering at Politeknik Negeri Bali, Indonesia. He is working toward a Ph.D. degree at the School of Engineering, the University of Warwick, U.K. His research interests include embedded machine learning and the Internet of Things.

Julian William Gardner (Fellow, IEEE) received the B.Sc. in physics from Birmingham University, Birmingham, U.K., in 1979, the Ph.D. degree in physical electronics from Cambridge University, Cambridge, U.K., in 1993, and the D.Sc. degree in electronic engineering from the University of Warwick, Coventry, U.K., in 1997. He has five years of experience in the industry as a Research and Development Engineer and 30 years in the field of chemical microsensors at the University of Warwick, and is currently a Professor of electronic engineering. He founded several electronic nose companies in 1990s. In 2008, he co-founded Cambridge CMOS Sensors Ltd., with Florin Udrea and was its first CTO; in 2013, he became its Chief Scientist. His expertise is in the design of MEMS-based chemical sensors and signal processing/data analysis of sensor arrays, including electronic noses and electronic tongues. He has published over 500 papers in the field of micro sensors, authored ten books, and holds over 25 patents.,Dr. Gardner has won various awards and medals from the IET, IEEE, and Royal Society. He is also a fellow of both the U.K. Institution of Engineering and Technology and the Royal Academy of Engineering.

Suhaib A. Fahmy (Senior Member, IEEE) received the MEng degree in information systems engineering and the PhD degree in electrical and electronic engineering from Imperial College London, U.K., in 2003 and 2007, respectively. From 2007 to 2009, he was a research fellow with Trinity College Dublin and a visiting research engineer with Xilinx Research Labs, Dublin. From 2009 to 2015, he was an assistant professor with Nanyang Technological University, Singapore. From 2015 to 2020, he was an associate professor and then a reader of computer engineering with the University of Warwick, U.K. Since 2020 he has been an associate professor of computer science with the King Abdullah University of Science and Technology, Saudi Arabia. His research interests include reconfigurable computing, high-level system design, and acceleration in a networked context. He was the recipient of Best Paper Award at the IEEE Conference on Field Programmable Technology in 2012, IBM Faculty Award in 2013 and 2017, and ACM TODAES Best Paper Award in 2019. He is a senior member of the ACM, and chartered engineer and member of the IET.