A Federated Learning Approach to Anomaly Detection in Smart Buildings

RAED ABDEL SATER and A. BEN HAMZA, Concordia Institute for Information Systems Engineering

Internet of Things (IoT) sensors in smart buildings are becoming increasingly ubiquitous, making buildings more livable, energy efficient, and sustainable. These devices sense the environment and generate multivariate temporal data of paramount importance for detecting anomalies and improving the prediction of energy usage in smart buildings. However, detecting these anomalies in centralized systems is often plagued by a huge delay in response time. To overcome this issue, we formulate the anomaly detection problem in a federated learning setting by leveraging the multi-task learning paradigm, which aims at solving multiple tasks simultaneously while taking advantage of the similarities and differences across tasks. We propose a novel privacy-by-design federated learning model using a stacked long short-time memory (LSTM) model, and we demonstrate that it is more than twice as fast during training convergence compared to the centralized LSTM. The effectiveness of our federated learning approach is demonstrated on three real-world datasets generated by the IoT production system at General Electric Current smart building, achieving state-of-the-art performance compared to baseline methods in both classification and regression tasks. Our experimental results demonstrate the effectiveness of the proposed framework in reducing the overall training cost without compromising the prediction performance.

CCS Concepts: • **Computer systems organization** → Embedded and cyber-physical systems; Sensor networks; • **Security and privacy** → anomaly detection; • **Computing methodologies** → Machine learning; Multi-task learning;

Additional Key Words and Phrases: Federated learning, privacy by design, Internet of Things, recurrent neural network, smart building, anomaly detection

ACM Reference format:
Raed Abdel Sater and A. Ben Hamza. 2021. A Federated Learning Approach to Anomaly Detection in Smart Buildings. *ACM Trans. Internet Things* 2, 4, Article 28 (August 2021), 23 pages.
https://doi.org/10.1145/3467981

1 INTRODUCTION

Smart buildings are making bold use of data collected by **Internet of Things (IoT)** sensors to assist in a wide range of positive outcomes, including cost reduction, improved safety and

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada through the Discovery Grants Program.

Authors' address: R. A. Sater and A. B. Hamza, Concordia Institute for Information Systems Engineering, Concordia University, 1455 de Maisonneuve Blvd. West, EV7.640, Montreal, QC, Canada, H3G 1M8; emails: raed.abdelsater@mail.concordia.ca, hamza@ciise.concordia.ca.

2577-6207/2021/08-ART28 $15.00
https://doi.org/10.1145/3467981

maintenance, and prevention of building equipment downtime. In simple terms, IoT refers to a network of sensors and other devices that are capable of sending and receiving data [32, 35], allowing diverse network components to cooperate and make their resources available to execute a common task. IoT helps create interoperable networks in smart buildings by connecting various types of sensors and other devices from which actionable insights can be extracted through collection and analysis of massive amounts of real-time data.

IoT-enabled smart buildings are changing the way we live, leveraging the power of intelligent devices to remotely monitor and control key equipment in the premises while ensuring greater energy and operational efficiency [19, 27]. In such an intelligent environment, a key objective is to provide support tools to help building managers and users make cost-effective decisions when utilizing, for example, electrical energy. It is estimated that more than half the global electrical energy is consumed by commercial buildings, and around 45% of that energy is generated by **Heating, Ventilation and Air Conditioning (HVAC)** systems [11]. To intelligently improve performance and create smarter buildings, there is a pressing need for developing efficient machine learning models that effectively learn the history patterns of IoT sensors in an effort to increase energy efficiency and help cut costs. Such models not only help facility managers make strategic decisions through data analysis and actionable insights to ensure buildings are working smarter and running at maximum efficiency but can also help buildings self-diagnose and optimize. In particular, an anomaly detection model can be used to deliver insights on the present and future performance of critical assets in smart buildings.

Anomaly detection is the process of identifying anomalous observations, which do not conform to the expected pattern of other observations in a dataset. Detecting anomalies has become a central research question in IoT applications, particularly from IoT time-series data [10, 20, 34]. For instance, a lighting energy consumption pattern might vary in an office building. Smart lighting control systems use occupancy sensors in correlation with light sensors to dim light based on changing occupancy and daylight levels, with the benefit of conserving energy in the building [38]. Moreover, an observation of high energy consumption levels due to heating in a bank might be anomalous after working hours but not during the day. Occupancy sensors can be used to measure occupancy and space utilization, as well as in other systems such as HVAC to detect such anomalies [2]. Multiple types of sensors are of great importance to building operations and can be used for a number of applications in an IoT environment, including energy consumption monitoring and control, and security of critical systems. Most modern structures are equipped with a **building automation system (BAS)**, which enables facility managers to automate and oversee the energy efficiency of a building by controlling various components within a building's structure, such as HVAC.

Early works on detection and prediction tasks in smart buildings have focused on centralized approaches using IoT sensor data. Idé et al. [17] used correlation between multiple sensors to form neighborhood graphs for computing correlation anomaly scores, but they treat a multi-sensor system as a collection of centralized sensors. Bellido-Outeirino et al. [1] presented a building lighting automation system by integrating digital addressable lighting interface devices in wireless sensor networks using a centralized system, in which appliances are managed by a wireless sensor network that focuses on lighting automation, while considering maintenance and energy consumption costs. Yu et al. [41] developed a centralized real-time HVAC system to construct and stabilize virtual queues associated with indoor temperatures by minimizing the long-term total cost associated with the HVAC system in the smart grid, which is considered a key IoT application that involves the incorporation of a secure, two-way information and communication flow, along with a two-way power flow. Li et al. [23] proposed a centralized water leak detection method using a classifier based on artificial neural networks using acoustic emission sensor data in an effort to detect water leaks in municipal pipeline systems. Chandra et al. [5] presented a Bayesian approach to multi-task learning for dynamic time series prediction via cascaded neural networks by decomposing a single task learning problem into multi-task learning through subtasks that have inter-dependencies defined by the size of the window used for embedding. To detect spatial, temporal, and spatio-temporal anomalies in real-time sensor measurement data streams, Chen et al. [7] proposed an anomaly detection framework for real-world environmental sensing systems in a bid to identify outliers in raw measurement data.

In centralized systems, an abnormal sensor behavior is detected by a central model on a server. Typically, all the training data are collected from the sensors through gateways and saved on the cloud. Centralized systems are, however, prone to failures, vulnerable to cyber invasions and infections, and often require longer access times for training data coming from remote devices, leading to potential data exposure during the transmission from clients to server, and hence raising concerns about clients' privacy [14]. Federated learning has recently emerged as a powerful alternative to centralized systems, as it enables the collaborative training of machine learning models from decentralized datasets in users' devices such as mobile phones, wearable devices, and smart sensors without uploading their privacy-sensitive data to a central server or service provider [3, 9, 21, 24–26, 29, 31, 39], while reducing communication cost [4]. For example, in a federated learning system for smart buildings, each sensor performs model training using its own data and sends local updates to the central server for aggregation to update the global model, which is then redistributed to the sensors. The training process of a federated learning model is usually carried out using the federated averaging algorithm [29] and is repeated iteratively until model convergence or the maximum number of training rounds is reached. Moreover, secure aggregation can be used to combine the outputs of local models on sensors for updating a global model by leveraging secure multiparty computation [28].

While federated learning has some privacy-enhancing advantages as compared to sharing private data with a central server, recent studies have shown that an honest-but-curious server can analyze the local model parameter updates to perform gradient leakage attacks [13, 47], which usually occur on comprised sensors, and hence the server may gain access to some private training data. To mitigate this issue of data leakage, federated learning can be used in conjunction with secure multi-party computation, homomorphic encryption, or differential privacy. With secure multi-party computation, for instance, the local model updates are securely combined into a single aggregate update to ensure that the data communicated through federated learning to the centralized server stays private [4]. Another effective strategy to prevent the gradients from leaking private training data is to set those with small magnitudes to zero [47].

In this article, we introduce a federated stacked long short-time memory model using federated learning on time series data generated by IoT sensors for classification and regression tasks such as lighting fault detection and energy usage prediction. In addition to learning long-term dependencies between timesteps of sequence data, the proposed model learns individual feature correlations within each sensor and also shared feature representations across distributed datasets to improve model convergence for faster learning. We perform the training process, which involves input data from multiple heterogeneous data sources across numerous sensors, in a collaborative fashion on IoT sensors in lieu of the server. This strategy not only lowers the cost and anticipates failures, but also helps reduce computational demands and inherits the privacy-enhancing capabilities of federated learning. The proposed framework provides (i) a reduced network traffic by sharing the weights solely with the federated server, (ii) a shorter convergence time thanks to the collaborative learning, and (iii) a federated methodology for training various types of sensors such as occupancy sensors. Moreover, we design a federated gated recurrent unit model and also a federated logistic regression to learn a common representation among multiple sensors. Our main contributions can be summarized as follows:

- We leverage the multi-task learning paradigm to formulate the anomaly detection problem in smart buildings.
- We propose a novel privacy-by-design federated stacked long short-time memory model for anomaly detection in smart buildings using IoT sensor data.
- We demonstrate that our proposed federated stacked **Long Short-Term Memory (LSTM)** model converges 2× faster than the centralized LSTM during the training phase and significantly reduces the communication cost.
- We present experimental results to demonstrate the superior performance of our model in comparison with baseline methods on sensors event log and energy usage datasets.

The rest of this article is organized as follows. In Section 2, we provide a brief overview of centralized and federated learning approaches for anomaly detection in a smart building setting using IoT sensor data. In Section 3, we present our federated learning system as well as our problem formulation and propose a novel federated learning architecture for anomaly detection in smart buildings. We discuss in detail the main components and algorithmic steps of the proposed framework. In Section 4, we present experimental results to demonstrate the superior performance of our approach in comparison with baseline methods. Finally, we conclude in Section 5 and point out future work directions.

2 RELATED WORK

The growing number of smart city centers has accelerated the proliferation of IoT sensors to observe and/or interact with their internal and external environments. In recent years, the advent of deep learning has sparked interest in the adoption of deep neural networks for learning latent representations of IoT sensor data. Du et al. [12] introduced a deep neural network using LSTM to model a system log as a natural language sequence by learning log patterns from normal execution in order detect anomalies when log patterns deviate from the model trained from log data under normal execution. Zhu et al. [46] proposed a multi-task anomaly detection framework for control area network messages using LSTM on in-vehicle network data. Zhang et al. [43] also used an LSTM model to predict power station working conditions from data generated by industrial IoT sensors of a main pump in a power station. However, all the aforementioned approaches are centralized, as data generated by IoT devices are sent directly to the server, raising privacy concerns and resulting in network overload. Moreover, the training process of centralized approaches is bandwidth intensive and comes with significant privacy implications.

More recently, federated learning has emerged as a viable, compelling alternative to the centralized approach. Rather than aggregating increasingly large amounts and types of data into a central location, federated learning distributes the global model training process such that each participating sensor's data are used in situ to train a local model. Nguyen et al. [33] proposed an anomaly detection system for detecting compromised IoT devices using federated learning by aggregating anomaly-detection profiles for intrusion detection. Li et al. [22] presented an autoencoder based anomaly detection approach at the server side to detect anomalous local weight updates from the clients in a federated learning system. Yurochkin et al. [42] developed a probabilistic federated learning framework with a particular emphasis on training and aggregating neural network models by decoupling the learning of local models from their aggregation into a global federated model. Zhao et al. [44] presented a multi-task federated learning method for computer networks anomaly

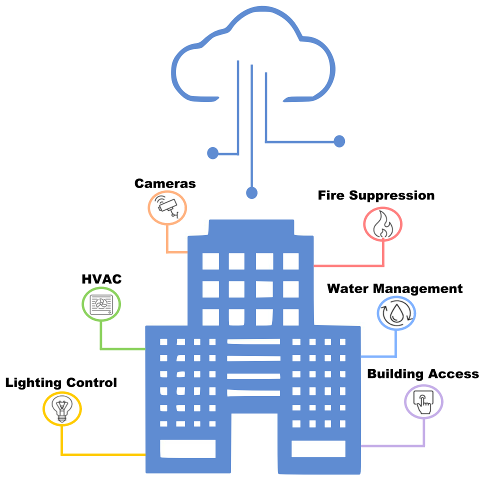

Fig. 1. IoT-enabled smart building.

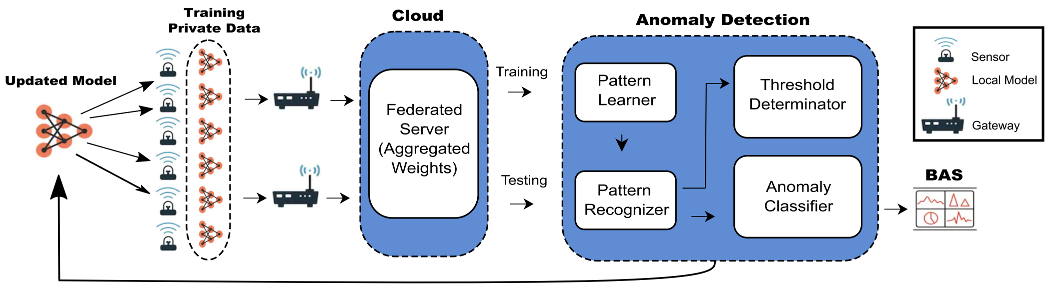

Fig. 2. Federated learning system for anomaly detection.

detection, as well as for traffic recognition and classification tasks. Chen et al. [9] introduced a federated multi-task hierarchical attention model for activity recognition and environment monitoring using multiple sensors.

3 METHOD

Motivated by LSTM networks' capability of learning long-term dependencies between timesteps of sequence data [16], we introduce in this section a novel privacy-by-design federated stacked LSTM model for anomaly detection in smart buildings using IoT sensor data. The proposed approach leverages the multi-task learning paradigm.

In a smart building setting with a federated learning system setup, we assume that there are K sensors, each of which has a dataset $\mathcal{D}_k$ that is kept private, where $k = 1, \ldots, K$. In other words, the data $\mathcal{D}_k$ of the kth sensor are not shared with the server. An IoT-enabled smart building is depicted in Figure 1, which shows several types of sensors for a variety of tasks, including lighting control, building access, water management, HVAC, fire suppression, and building monitoring. Unlike the traditional centralized learning that collects and uses all local data $\mathcal{D} = \cup_{k=1}^{K} \mathcal{D}_k$ from all sensors to train a model, the federated learning paradigm only collects and aggregates updated local models from the sensors to generate a global model.

An overall schematic layout of our federated learning system for anomaly detection is illustrated in Figure 2. The proposed system uses historic sensor data and contextual features to identify temporal anomalies in smart buildings using a multi-task federated recurrent neural network to classify anomalous sensors and predict energy consumption. The main building blocks of this system are local training, cloud aggregation, anomaly detection and global model broadcast.

Local training. Sensor data represent a time stamped input of energy consumption data recorded at regular time intervals. These sensor data are often noisy and incomplete due largely to faulty devices and/or communication errors [30]. To mitigate the negative impact of noisy and incomplete data on the performance of the federated learning system, these data instances are usually discarded or interpolated. Each sensor trains the local model on anomalous instances. The basic assumption is that historic sensor data are predominantly normal. The historical real datasets can then be split into training and test sets with the objective of using the test data to evaluate the capacity of the anomaly detection framework to identify normal behavior during the anomaly detection phase. The primary goal of training is to enhance engine recognition of normal input data patterns. At the end of the training phase, the participating sensors send the learned parameters of their local models for aggregation to a central server via Wireless Area Controller gateways that monitor and control data traffic between the sensors and the IoT system in the cloud [18].

Cloud aggregation. We use PySyft, a Python library for secure and private deep learning [37], which integrates federated learning with secure multi-party computation and differential privacy in an effort to protect against threats within the data center by ensuring that individual sensors' updates remain encrypted in memory. The secure multi-party computation protocol leverages encryption to make individual sensors' updates uninspectable by a server [4]. At each communication round between the server and participating sensors, the server aggregates the learned parameters of the local models using the Federated Averaging algorithm, followed by updating the global model. This training process is repeated for a certain number of communication rounds until a desirable level of performance is achieved. Then, the updated global model is prepared for integration into the anomaly detection engine.

In PySyft, private data leakage is mitigated with secure aggregation [4] by leveraging secure multiparty computation to compute sums of local model parameter updates from individual sensors while maintaining privacy guarantees. With secure aggregation, the local model parameter updates are kept encrypted and their sum is only revealed to the server after a sufficient number of communication rounds. Homomorphic encryption and differential privacy can also be used in PySyft. However, only secure aggregation is used in our algorithm. In differential privacy, for instance, each sensor adds a carefully calibrated amount of noise to its local parameter update in an effort to mask its contribution to the learned global model.

Anomaly detection. The anomaly detection block of the proposed system is composed of four main components: the pattern learner process, pattern recognizer, threshold determinator, and anomaly classifier. The pattern learner receives the global model after the training has been completed and configured for testing. The pattern recognizer tests the global model and evaluates the performance based on the test data to help the threshold determinator find a suitable value for the specific problem. Based on the value of the threshold, the anomaly classifier generates the inference and predicts the anomalous data points that did not pass the threshold determinator and flags them in the anomaly detection system. The updated global model will then be ready for deployment on sensors after this round of training.

Table 1. Notation

Notation	Description
K	Number of participating sensors indexed by k
$\mathbf{X}^k$	Training data for the kth sensor with N_k local examples
$\mathbf{y}^k$	Ground-truth label or target output vector
$\hat{\mathbf{y}}^k$	Predicted probability or response vector
$\mathcal{L}(\cdot,\cdot)$	Loss function
$\sigma(\cdot)$	Sigmoid activation function
B	Local minibatch size
E	Number of local epochs
w_G	Global model
FC	Fully Connected layer
BA	Balanced Accuracy
LR	Logistic Regression
FLR	Federated Logistic Regression
FGRU	Federated Gated Recurrent Unit
LSTM	Long Short-Term Memory network
FLSTM-ℓ	Federated LSTM with ℓ layers
FSLSTM	Federated Stacked LSTM (i.e., FLSTM-3)

Global model broadcast. Once the model weights are updated using the Federated Averaging algorithm, the anomaly detection engine, located in the IoT system, sends back the updated global model to all sensors that are involved in the first round of training. These sensors receive the updated global model with new weights and then replace their local models' parameters with the global parameters to start a new round of training.

3.1 Multi-Task Learning

We leverage the multi-task learning paradigm, which aims at solving multiple tasks simultaneously while taking advantage of the similarities and differences across tasks, to formulate the anomaly detection problem in a federated learning setting. Our notation is summarized in Table 1.

Suppose that we have K learning tasks (i.e., one task per sensor) and denote by $(\mathbf{X}^k, \mathbf{y}^k)$ the training data for the kth task, where $\mathbf{X}^k = (\mathbf{x}_1^k, \ldots, \mathbf{x}_{N_k}^k)^T \in \mathbb{R}^{N_k \times F}$ is a data matrix consisting of N_k samples collected from multiple sensors of the same category (e.g., occupancy sensors), and $\mathbf{y}^k = (y_1^k, \ldots, y_{N_k}^k)^T \in \mathbb{R}^{N_k}$ is a vector of outputs for the N_k samples. The training samples in $\mathbf{X}^k$ are generated by sensors connected to different gateways in the building. Further, we assume that the samples have the same feature dimension F across all tasks. It is important to note that $y_i^k \in \mathbb{R}$ for regression tasks, while $y_i^k \in \{0, 1\}$ for binary classification tasks with 0 and 1 representing "normal" and "anomalous" observations, respectively.

The goal of multi-task learning is to learn the weight parameters of a model by minimizing the following objective function across all sensors

$$\mathcal{E}_{\text{avg}} = \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}(\mathbf{y}^k, \hat{\mathbf{y}}^k), \tag{1}$$

where $\mathcal{L}$ is a loss function and $\hat{\mathbf{y}}^k$ is a vector of predicted values by the model for the kth task.

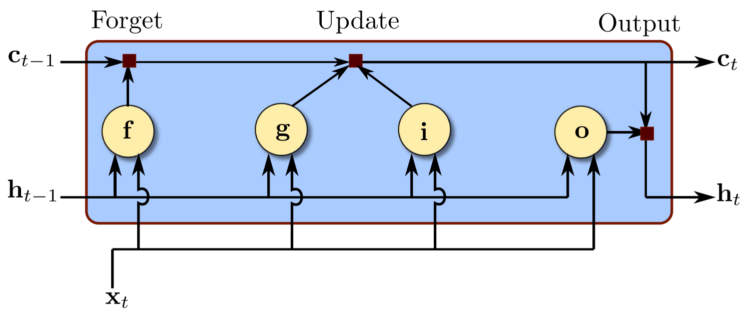

Fig. 3. LSTM block architecture.

For binary classification tasks, we train a model to minimize the cross-entropy loss function given by

$$\mathcal{L}(\mathbf{y}^k, \hat{\mathbf{y}}^k) = -\frac{1}{N_k} \sum_{i=1}^{N_k} y_i^k \log(\hat{y}_i^k), \tag{2}$$

where y_i^k and $\hat{y}_i^k$ are the ground-truth label and predicted probability, respectively, for the kth task.

For regression tasks, we train a model to minimize the **mean squared error (MSE)** given by

$$\mathcal{L}(\mathbf{y}^k, \hat{\mathbf{y}}^k) = \frac{1}{N_k} \sum_{i=1}^{N_k} \left(y_i^k - \hat{y}_i^k\right)^2, \tag{3}$$

where y_i^k and $\hat{y}_i^k$ are the target output and predicted value by the model, respectively, for the kth task.

3.2 LSTM

LSTM networks are a special type of recurrent neural networks, capable of learning long-term dependencies between timesteps of sequence data while being resilient to the vanishing gradient problem [16]. The key to an LSTM network is the cell state, which contains information learned from the previous timesteps and has the ability to remove or add information using gates [6]. These gates control the flow of information to and from the memory. In addition to the hidden state, the architecture of an LSTM block is composed of a cell state, forget gate, memory cell, input gate and output gate, as illustrated in Figure 3. At each timestep, the LSTM block takes as input the current input data vector $\mathbf{x}_t$ and both the hidden state (i.e., short-term memory) $\mathbf{h}_{t-1}$ and cell state (i.e., long-term memory) $\mathbf{c}_{t-1}$ from the previous cell. To decide which information to be retained or discarded at each timestep before passing on the long-term and short-term information to the next cell, the LSTM block uses the forget, input and output gates, which are trainable functions with weights and biases. The forget gate decides which information from the long-term memory to forget, while the input gate can be regarded as a filter that selects what information can be kept and what information to be thrown out. The memory cell $\mathbf{g}_t$ is created by passing the current input and short-term memory into a tanh activation function, which is a shifted version of the sigmoid activation function. The new cell state $\mathbf{c}_t$ is obtained by adding two pointwise multiplication terms; the first term involves the input gate and memory cell, while the second one uses the forget gate and the previous cell state. The cell state $\mathbf{c}_t$ stores information about the input data across timesteps. Finally, the hidden state $\mathbf{h}_t$ is obtained via pointwise multiplication of the output gate $\mathbf{o}_t$ and the new cell state through a tanh activation function. This hidden state (i.e., new short-term memory) is then passed on to the cell in the next timestep.

Formally, given the input $\mathbf{x}_t$, current cell state $\mathbf{c}_{t-1}$ and hidden state $\mathbf{h}_{t-1}$ of the network, the LSTM updates at timestep t are given by

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{R}_f \mathbf{h}_{t-1} + \mathbf{b}_f),\\
\mathbf{g}_t &= \tanh(\mathbf{W}_g \mathbf{x}_t + \mathbf{R}_g \mathbf{h}_{t-1} + \mathbf{b}_g),\\
\mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{R}_i \mathbf{h}_{t-1} + \mathbf{b}_i),\\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t,\\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{R}_o \mathbf{h}_{t-1} + \mathbf{b}_o),\\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t),
\end{aligned} \tag{4}$$

where $\mathbf{f}_t$, $\mathbf{g}_t$, $\mathbf{i}_t$, $\mathbf{o}_t$, $\mathbf{c}_t$, and $\mathbf{h}_t$ are the forget gate, memory cell, input gate, output gate, cell state and hidden state, respectively; $\sigma(\cdot)$ denotes the sigmoid activation function; $\odot$ denotes the pointwise product; $\mathbf{W}_\bullet$ and $\mathbf{R}_\bullet$ are the learnable input and recurrent weight matrices; and $\mathbf{b}_\bullet$ are the learnable bias vectors.

In summary, the input gate controls what new information is added to cell state from current input, while the forget gate controls what information to throw away from memory. The output gate controls what information encoded in the cell state is sent to the network as input in the following timestep. An LSTM network with multiple LSTM layers is referred to as a stacked or deep LSTM, with the output sequence of one LSTM layer forming the input sequence of the next.

3.3 Proposed Framework

To overcome the centralized training issues such as computational demand, IoT device availability and network bandwidth limitation, we introduce a **federated stacked long short-time memory (FSLSTM)** model that contains two main components: a local model and a global model. The local model is a stacked LSTM network composed of three LSTM layers and is applied on input data generated by each sensor to learn a latent feature representation. A **fully connected (FC)** layer is applied to the hidden representation of the last LSTM layer, followed by a softmax or linear activation function for classification and regression tasks, respectively. The global model, however, aggregates the weights of all local models after each round. The architecture of the proposed FSLSTM model consists of three LSTM layers, as illustrated in Figure 4.

In a federated learning setting, the training of a stacked LSTM model is distributed across the participating sensors by iteratively aggregating local models into a joint global model. Suppose we have training local datasets $\{(\mathbf{X}^k, \mathbf{y}^k)\}_{k=1}^K$ from K sensors, where the total number of samples distributed over these sensors is $N = \sum_{k=1}^K N_k$, with N_k denoting the number of samples in the training set $\mathcal{D}_k = (\mathbf{X}^k, \mathbf{y}^k) = \{(\mathbf{x}_i^k, y_i^k)\}_{i=1}^{N_k}$ of the kth sensor. We consider a federated learning task, where these K sensors collaboratively train a model parameter vector $\mathbf{w}$ with the orchestration of a remote server. The goal is to minimize the following global loss function on all the distributed datasets,

$$f(\mathbf{w}) = \sum_{k=1}^{K} \frac{N_k}{N} f_k(\mathbf{w}), \tag{5}$$

where $f_k(\mathbf{w})$ is the local loss function on the collection of data samples $\xi_i^k = (\mathbf{x}_i^k, y_i^k)$ from the kth sensor

$$f_k(\mathbf{w}) = \frac{1}{N_k} \sum_{i=1}^{N_k} \ell(\mathbf{w}; \xi_i^k), \tag{6}$$

and $\ell(\mathbf{w}; \xi_i^k)$ is the loss function of the prediction on the data sample ξ_i^k made with model parameter vector $\mathbf{w}$. It is important to point out that in the case of the IID assumption (i.e., the training

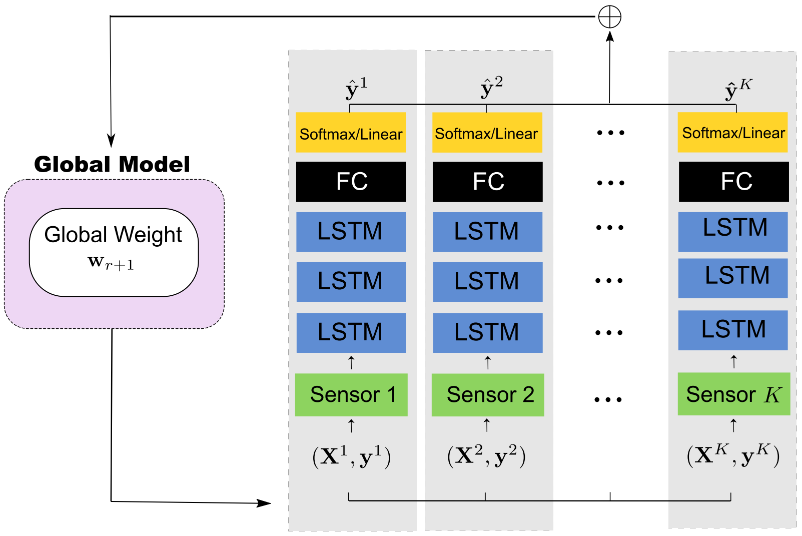

Fig. 4. Architecture of proposed federated stacked LSTM model. Only the local model parameter updates are sent back to the server.

examples are distributed over the clients uniformly at random), we have $\mathbb{E}_{\mathcal{D}_k}[f_k(\mathbf{w})] = f(\mathbf{w})$, where the expectation is taken over the set of samples assigned to a fixed sensor k.

Each local LSTM model receives a local copy of the global model weights based on a time series window with step t.

Local model. Each participating sensor uses its local data to update parameters of the local model, which is a stacked LSTM network with three layers. As stated earlier, at each timestep t, the LSTM block takes as input the current input data vector $\mathbf{x}_t$ and both the hidden state $\mathbf{h}_{t-1}$ and cell state $\mathbf{c}_{t-1}$ from the previous cell. Then, the LSTM network learns to predict the hidden and cell states of the next timestep as follows:

$$\mathbf{h}_t, \mathbf{c}_t = \text{LSTM}(\mathbf{h}_{t-1}, \mathbf{x}_t, \mathbf{c}_{t-1}; \mathbf{w}), \tag{7}$$

where LSTM is an operator representing the operations in Equation (4), $\mathbf{h}_t$ is the new hidden state, $\mathbf{c}_t$ is the new cell state, and $\mathbf{w}$ is the parameter vector of the LSTM model. We initialize $\mathbf{c}_0$ and $\mathbf{h}_0$ as random vectors.

The output of the last hidden state is the hidden vector of the last timestep, and can be viewed as the representation of the whole sequence. Then, we pass this hidden representation to a fully connected layer to get the predicted values for the kth task as follows:

$$\hat{\mathbf{y}}^k = \sigma(\mathbf{W}^{\text{FC}}\mathbf{h}_{\text{last}} + \mathbf{b}^{\text{FC}}), \tag{8}$$

where σ is the softmax or linear activation function, $\mathbf{h}_{\text{last}}$ denotes the hidden vector in the last timestep of LSTM, $\mathbf{W}^{\text{FC}}$ and $\mathbf{b}^{\text{FC}}$ are the learnable weight matrix and bias vector of the FC layer.

Global model. At each round r, the server randomly chooses a subset S_r of sensors for synchronous aggregation and broadcasts the global model parameter vector $\mathbf{w}_r$ to the participating sensors. For the choice of S_r, the random number generator is initialized with the same seed value for all training rounds. Each sensor updates the local model parameters by minimizing the loss function over its local data dataset, starting from the global model parameter vector $\mathbf{w}_r$ shared by the server and using stochastic gradient descent algorithm for E epochs and with a batch size B. At the end of the training phase, the parameters of local models are sent to the server for aggregation, which is done synchronously using the federated averaging algorithm to obtain the global model parameter vector $\mathbf{w}_{r+1}$ for the next round.

The update of local and global models is repeated for a certain number of rounds until the global loss function converges. The main algorithmic steps of our proposed federated stacked LSTM network are summarized in Algorithm 1. This algorithm starts by randomly initializing the global model. Then, the server randomly selects a subset of sensors and distributes the current global model to these sensors. Each sensor trains the global model with its local data independently, and then the server collects the parameters of the locally trained models for all selected sensors and aggregates them using the federated averaging algorithm by computing a weighted average of these parameters to obtain a shared global model.

At the beginning of the training round of communication, each sensor reads the current parameter vector of the global model from the central server and updates it via stochastic gradient descent, where the stochastic gradient computed using a mini-batch sampled uniformly at random from the local dataset of the kth sensor. At the end of the training round of communication, our proposed federated stacked LSTM returns a vector of predicted values. Then, we concatenate all the predicted outcomes from the K sensors to obtain a vector $\hat{\mathbf{y}} \in \mathbb{R}^N$ as follows:

$$\hat{\mathbf{y}} = \hat{\mathbf{y}}^1 \oplus \ldots \oplus \hat{\mathbf{y}}^K, \tag{9}$$

where $\oplus$ denotes the concatenation operator.

Based on these predictions, the model sends back answers to the anomaly detection engine, where the values are passed on to the threshold determinator to make decisions regarding anomalous sensors, and subsequently take appropriate actions for isolation and maintenance.

ALGORITHM 1: Federated Stacked LSTM

```
Input: Training sets $\{(\mathbf{X}^k, \mathbf{y}^k)\}_{k=1}^K$ from $K$ sensors, initial global model parameters $\mathbf{w}_0$, local mini-
    batch size $B$, number of local epochs $E$, learning rate $\eta$, number of rounds $R$, number of sensors
    per round $m$.
Output: Vector $\hat{\mathbf{y}}$ of predicted values.
 1: for $r = 1$ to $R$ do
 2:     Server randomly selects a subset $S_r$ of $m$ sensors.
 3:     Server broadcasts $\mathbf{w}_r$ to the subset $S_r$
 4:     for each sensor $k \in S_r$ in parallel do
 5:         $\mathbf{w}_{r+1}^k \leftarrow$ SensorUpdate$(k, \mathbf{w}_r)$
        $\mathbf{w}_{r+1} \leftarrow \sum_{k=1}^K \frac{N_k}{N} \mathbf{w}_{r+1}^k$                    //aggregate updates

    SensorUpdate$(k, \mathbf{w})$:                                          //run on sensor k
    $\mathcal{B} \leftarrow$ partition local data $(\mathbf{X}^k, \mathbf{y}^k)$ into batches of size $B$
 6: for each local epoch from 1 to $E$ do
 7:     for mini-batch sample $\xi \in \mathcal{B}$ do
 8:         $\mathbf{h}_t, \mathbf{c}_t = \mathrm{LSTM}(\mathbf{h}_{t-1}, \mathbf{x}_t, \mathbf{c}_{t-1}; \mathbf{w})$          //LSTM at timestep t
 9:         $\hat{\mathbf{y}}^k = \sigma(\mathbf{W}^{\mathrm{FC}} \mathbf{h}_{\mathrm{last}} + \mathbf{b}^{\mathrm{FC}})$                //predicted outcomes
10:         $\mathbf{w} \leftarrow \mathbf{w} - \frac{\eta}{B} \sum_{\xi \in \mathcal{B}} \nabla \ell(\mathbf{w}; \xi)$             //update local model
    return learned parameter vector w to server
11: Concatenate predicted values: $\hat{\mathbf{y}} \leftarrow \hat{\mathbf{y}}^1 \oplus \ldots \oplus \hat{\mathbf{y}}^K$
```

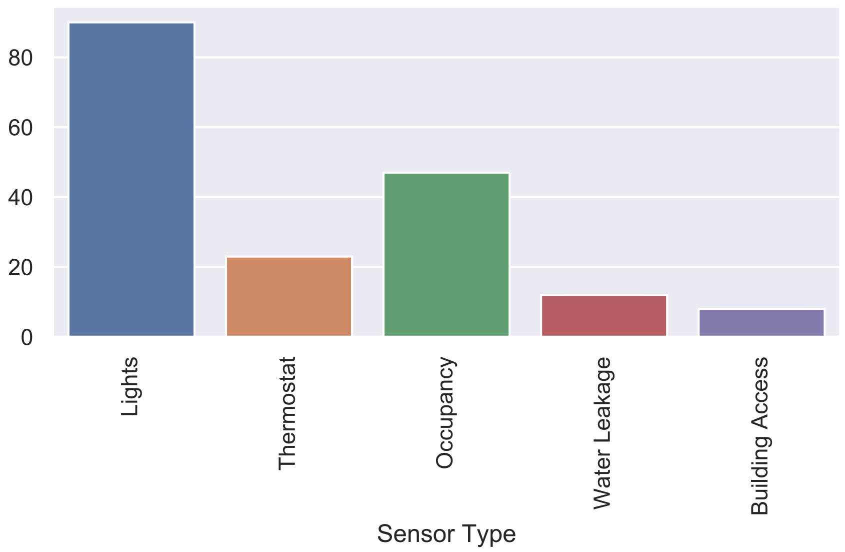

Fig. 5. Distribution of sensors' categories.

4 EXPERIMENTS

In this section, we conduct experiments to demonstrate and analyze the performance of the proposed FSLSTM model in anomaly detection and regression on three real-world datasets generated by the IoT production system at General Electric Current smart building in Montreal. For privacy concerns, we hashed the primary keys and a few privacy-sensitive features such as vendor name, device identification and some of the prototypes information. The effectiveness of our approach is validated by performing comparison with several baseline methods.

Datasets. We use a Sensors Event Log dataset for anomaly detection and an Energy Usage dataset for electricity consumption. We also use a Weather API dataset in conjunction with energy usage to enrich model learning. The weather data are collected from the building API that displays the basic weather parameters such as temperature, barometric pressure, humidity, precipitation, solar radiation and wind speed from May to August 2019. The weather data are used as a secondary source of information to enrich the Sensors Event Log dataset and help the model predict the energy usage. We simulate each zone in a separate category and allocate a specific task to it for learning.

The Sensors Event Log and Energy Usage datasets are generated by 180 sensors from five different categories: lights, HVAC thermostats, occupancy sensors, water leakage sensors, and access sensors. The distribution of these categories is shown in Figure 5. We use these categories for classification tasks, as the target is to identify which sensor is faulty, rendering it unable to properly communicate with HVAC and other control devices. In addition, the energy usage metric of thermostat sensors is used for regression tasks, with the aim of predicting future energy consumption of IoT devices, such as temperature, humidity, and pressure sensors, which are the most commonly used sensors for HVAC and building equipment applications. The pre-processing phase of data generation and wrangling are performed to ensure the confidentiality of the datasets used in our experiments and the privacy of the company's infrastructure.

- **Sensors Event Log Dataset:** Sensors data such as occupancy sensors, lights, thermometer and humidity are collected from 180 devices. These sensors are categorized into 5 different groups and are distributed all over the building. In our experiments, we select 1 million event logs with a window time of 4 months. We limit data collection to one particular season (summer in our case) to learn feature vectors of time and frequency domain variables for this specific period of the year. We believe that each season should be considered separately for better learning purposes. For example, the heating pattern during the winter season is different from the cooling stages during summer. We consider each sensor as a separate task and predict its activities such as drop in temperature, excessive energy usage and running water.

- **Energy Usage Dataset:** Energy data gathered from sensors measure the electricity consumed per device. In our case, appliances can represent any equipment that is connected to the building automation system and falls under the five aforementioned categories. The unit is measured in kW/h for a variety of appliances, including LED light bulbs, rooftop units, humidity sensors, smart sensors that capture water leakage events, and occupancy activities around the building. The latter are sophisticated sensors, which are equipped with a smart dashboard for data analytics and different measuring tools to monitor important indicators related to indoor farms and laboratories. The data are aggregated every 15 minutes, stored in the Energy Usage table, and then merged into the Event Log dataset based on the sensor ID.

Baseline methods. We evaluate the performance of the proposed FSLSTM network consisting of three LSTM layers against several baseline models, including centralized **logistic regression (LR)**, LSTM, **federated logistic regression (FLR)**, **federated gated recurrent unit (FGRU)**, **federated LSTM with one LSTM layer (FLSTM-1)**, and **federated LSTM with two LSTM layers (FLSTM-2)**. Both LSTM and GRU networks address the vanishing/exploding gradient problem of traditional recurrent neural networks. While the LSTM block consists of three gates (forget, input and output), the GRU block has only two gates (update and reset). The update gate in the GRU block determines the amount of previous information that needs to be passed along the next state, while the reset gate decides how much of the past information is needed to neglect. It is worth pointing out that we designed the federated logistic regression and federated GRU baseline models for comparison purposes with FSLSTM, whereas the centralized logistic regression and LSTM are well-known prediction baselines in the literature. Our proposed framework leverages the popular federated averaging algorithm, which uses stochastic gradient descent as both sensor and server optimizers to update the local and global parameters, where the server learning rate is equal to 1. More recently, Sashank et al. [36] have introduced an adaptive optimization framework, in which federated versions of popular adaptive algorithms are incorporated for the sensor-side or server-side model updates.

Implementation details. All experiments are carried out on a Linux desktop computer running 4.4 GHz and 64-GB RAM with an NVIDIA GeForce RTX 2080 Ti GPU. The algorithms are implemented in PyTorch. The hyper-parameters are optimized using grid search. For fair comparison, we set the number of hidden units to 128 per layer in both LSTM and GRU blocks. We also set the number of nodes in the fully connected layer to 100. In addition, we use regular and recurrent dropouts of 20% and apply the ReLU activation function in an effort to avoid overfitting. Our grid search selects a batch size of 1,024 for training and an initial learning rate of 0.001 as the best combination. We use cross-entropy and MSE as loss functions for classification and regression tasks, respectively. In all experiments, we split our datasets into 80% training and 20% testing.

4.1 Results

The effectiveness of our FSLSTM model (i.e., FLSTM-3) is assessed by conducting a comprehensive comparison with the baseline methods using several performance evaluation metrics. The results are summarized in Table 2.

Table 2. Performance Comparison Results of FSLSTM Against Baselines Models on the Sensors Event Log Test Set for Anomaly Detection (Classification) and Energy Usage Test Set (Regression)

	Sensors Data				EU Data		
Methods	Precision	Recall	F1	BA	MAE	MSE	RMSE
LR	0.57	0.60	0.52	0.72	0.341	0.48	0.692
LSTM	0.66	0.61	0.58	0.71	0.243	0.33	0.574
FLR (ours)	0.65	0.71	0.70	0.69	0.339	0.34	0.583
FGRU (ours)	0.84	0.66	0.59	0.80	0.211	0.29	0.538
FSLSTM (ours)	**0.89**	**0.79**	**0.87**	**0.90**	**0.162**	**0.19**	**0.435**

Bold numbers indicate the best performance.

For regression tasks, we use the **mean absolute error (MAE)**, MSE, and **root mean squared error (RMSE)** as evaluation metrics, which are given by

$$\mathrm{MAE} = \frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}_i|, \tag{10}$$

$$\mathrm{MSE} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2, \tag{11}$$

and RMSE $= \sqrt{\mathrm{MSE}}$, where m is the number of samples in the test set, y_i is the actual (ground truth) value, and $\hat{y}_i$ is the model's predicted value. A small value of these error metrics indicates a better performance of the model. We rely on a 15 minute window aggregation to reduce buffer overflow over the network and also to remove missing values that are sometimes generated by some sensors due to inactivity. As shown in Table 2, FSLSTM significantly outperforms the baselines methods on the Energy Usage datasets. In terms of MAE, for example, FSLSTM yields a much smaller error compared to the one obtained by LSTM, and more than half the error obtained by LR and FLR. Moreover, FSLSTM yields a 4.9 percentage points performance improvement over FGRU.

For classification tasks, we use precision, recall, F1 score, BA, **receiver operating characteristic (ROC)** curve, and **area under the ROC curve (AUC)** as evaluation metrics. Precision and recall are defined as

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \quad \text{and} \quad \mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \tag{12}$$

with **true positives (TP)**, **false positives (FP)**, **true negatives (TN),** and **false negatives (FN)**, respectively. TP is the number of correctly predicted anomalous observations, while TN is the number of correctly predicted normal observations. Recall, also known as **true positive rate (TPR)**, is the percentage of positive instances correctly classified, and indicates how often a classifier misses a positive prediction.

AUC summarizes the information contained in the ROC curve, which plots TPR versus FPR = FP/(FP + TN), the false positive rate, at various thresholds. Larger AUC values indicate better performance at distinguishing between anomalous and normal observations. FPR is is the ratio of normal observations that were incorrectly classified as anomalous.

Since our datasets are highly imbalanced, we use the balanced F1 score defined as the harmonic mean of precision and recall, and the balanced accuracy given by

$$\mathrm{BA} = \frac{\mathrm{TPR}+\mathrm{TNR}}{2}, \tag{13}$$

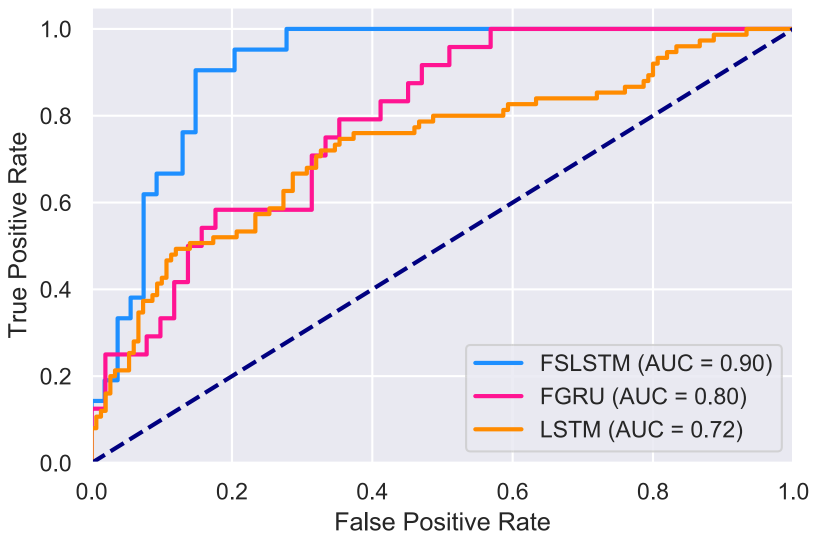

Fig. 6. ROC curves for FSLSTM and baseline methods, along with the corresponding AUC values, on the Sensors Event Log test set.

where TNR = 1 − FPR, also called specificity, refers to the ability of a classifier to correctly identify the observations that are not anomalous.

As shown in Table 2, the proposed model outperforms all the baseline methods by a large margin in terms of F1 score, yielding performance improvements of 29 percentage points over LSTM and 28 percentage points over FGRU. In terms of balanced accuracy, FSLSTM also achieves notable performance improvements of 19 percentage points over LSTM and 10 percentage points over FGRU. This demonstrates the significant prediction ability of FSLSTM in anomaly detection. It is also important to note that LSTM yields comparable performance to FLR, due in large part to the fact that linear models cannot effectively represent temporal dependency, even in a federated learning setting.

Figure 6 displays the ROC curves, which show the better performance of FSLSTM in comparison with baseline methods on the Sensors Event Log test set. Each point on ROC represents different tradeoff between false positives and false negatives. An ROC curve that is closer to the upper right indicates a better performance (TPR is higher than FPR). The overall performance of FSLSTM is significantly better than the baselines, as indicated by both the ROC curves and AUC values.

Federated vs. centralized learning. The LSTM model adopts a centralized approach, which requires the training data to be aggregated on the server, meaning that no training is performed on the edge devices. The learning is carried out by partitioning the data into training and test sets in a central machine. However, centralized training is privacy intrusive, especially for clients with important sensors' data aggregated on the cloud, as some sensors may contain private patterns, such as occupancy and access information. In a centralized setting, sensors have to trade their privacy by sending and storing data on the cloud owned by a third-party service provider.

Unlike the centralized training framework, our federated learning based approach is decentralized and enables sensors located at different locations inside the building to collaboratively learn a federated model, while keeping all data that may contain private information on devices. Therefore, sensors can benefit from using a federated model without sending raw data to the cloud. Sensitive data may include the sensor's model number, manufacturer name, or even serial numbers of newly produced prototypes that are still in the testing phase. Another limitation of centralized learning is the computing power [8]. In our setting, however, the computing power for each sensor is negligible compared to dedicated GPUs. In a centralized learning environment, a large amount of data collected from different IoT devices need to be merged into one dataset and then we wait for the machine to finish training. In our proposed federated framework, we train the model on the data of each device in parallel, resulting in better performance and faster convergence time. The better performance of our federated stacked LSTM model may be attributed not only to the strong capabilities of LSTM in prediction tasks, but also to the stacked LSTM layers that help generate a deep feature representation of the input data.

The ability to quickly detect and respond to anomalies is critical to the successful operation of smart buildings. Models trained on data of individual devices help improve critical incidents detection, such as technical glitches that may arise in smart buildings, where smart fire sensors, for instance, need not only be triggered in the case of an emergency, but also be capable of predicting specific scenarios and patterns. During the training phase, our FSLSTM model converges twice as fast as the centralized LSTM on the same datasets. As shown in Figure 7 (top), the centralized LSTM model does not seem to attain a stable state even after 50 epochs. The federated stacked LSTM model, however, is capable of significantly reducing the loss function and reaching a stable performance with higher accuracy in only 20 epochs, as shown in Figure 7 (bottom). Each epoch represents a full round of the 180 sensors that are participating in the experiment. Moreover, notice that the learning curves of the federated model are less fluctuating compared to the centralized LSTM. This smoothness property is attributed, in large part, to the number of sensors involved in the learning process.

Convergence performance. An important factor in our experiments is the convergence speed of the model. We test the effect of the parameter K (i.e., number of sensors) on the convergence performance of the proposed approach. The results of convergence time for FSLSTM, FGRU, and LSTM using a varying number of sensors on the Sensors Event Log dataset are shown in Figure 8. Note that unlike the centralized LSTM model, the convergence time of both federated models decreases when the number of sensors increases. Training a deep neural network involves using an optimization algorithm to find a set of weights to best map inputs to outputs, while convergence describes a progression towards a stable state where the network has learned to properly respond to a set of training patterns within some margin of error. The choice of the network's hyperparameters, such as the number of sensors, play an important role in convergence. To assess the performance of FSLSTM with respect to the number of sensors K, we use the mini-batch stochastic gradient descent optimization algorithm with a fixed batch size for the local update by increasing the value of K from 20 to 200. For all sensors, even when the learning rate is tuned carefully, FSLSTM achieves, in all batch size settings, a better performance, as reported in Table 2. Figure 8 shows that an increase in the number of participating sensors positively impacts the convergence time of FSLSTM, reducing the time from 6 hours for the centralized LSTM to only 2 hours for FSLSTM using 200 sensors. However, a larger number of participating sensors requires higher local computation at the sensors, resulting in an increase in energy consumption by these sensors.

It is also important to point out that there a is slight increase of convergence time when the number of sensors is between 20 and 40, but then the convergence time decreases when the number of sensors exceeds 42. This is largely due to the delay in response of the rooftop units' thermostats that are connected to the network through the Wireless Area Controller gateways. The latency time of these sensors is twice longer than other sensors in the buildings, leading to a slight increase in convergence time. Once these sensors are equipped with all the global parameters that are necessary for the training phase, the convergence time drops proportionally to the number of participating sensors.

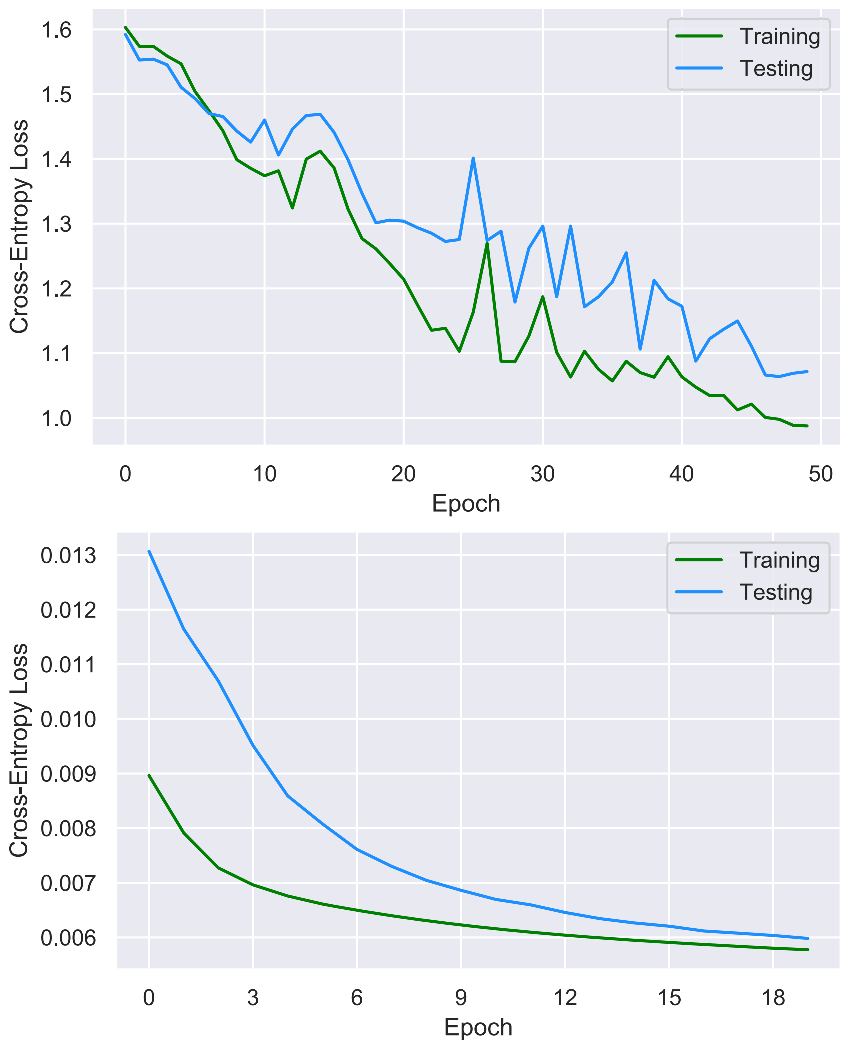

Fig. 7. Training and testing learning curves of LSTM (top) and FSLSTM (bottom) on the Sensors Event Log dataset.

Collective and contextual anomalies. A collective anomaly describes a group of data points that exhibits an anomalous behavior compared to the rest of the dataset [45]. An individual instance within the anomalous group is not necessarily anomalous on its own. If a data instance is anomalous in a specific context, but not otherwise, then it is termed as a contextual anomaly, also referred to as conditional anomaly [15]. We train FSLSTM on normal data before performing a live prediction for each timestep, which is equal to 600 minutes. Instead of considering each timestep separately, the observation of prediction errors from a certain number of timesteps is now used for detecting collective anomalies. The prediction errors from a number of the latest timesteps above a threshold, as set by the threshold determinator, will indicate a collective anomaly. We take a sample of our initial data, which includes four HVAC sensors, all measuring the electricity from power meters located in different areas within the building. In the first step, FSLSTM determines

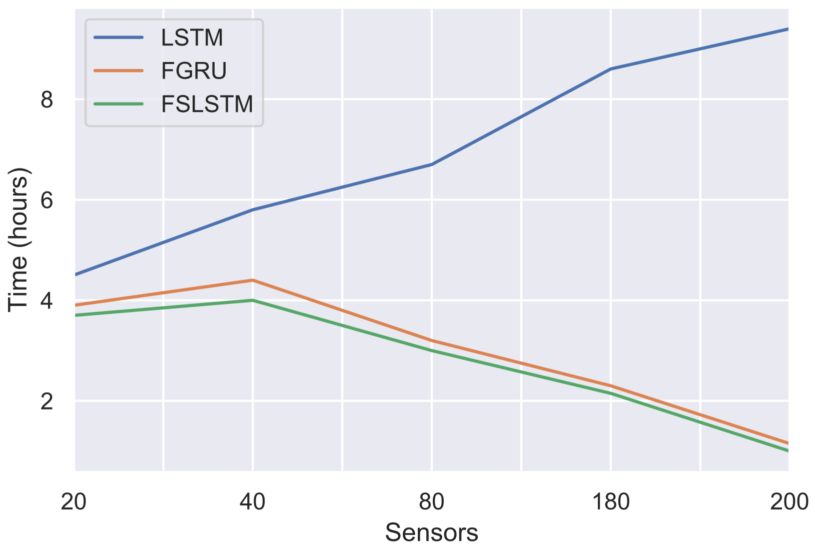

Fig. 8. Convergence time comparison between FSLSTM, FGRU, and LSTM using a varying number of sensors on the Sensors Event Log dataset.

Table 3. Collective and Contextual Anomaly Detection Comparison

Method	Collective Anomalies		Contextual Anomalies	
	Correct Alarms Triggered (%)	False Alarms Triggered (%)	Correct Alarms Triggered (%)	False Alarms Triggered (%)
LR	56	54	63	48
LSTM	66	33	74	29
FLR (ours)	65	21	78	18
FGRU (ours)	74	12	82	7
FSLSTM (ours)	**88**	**9**	**90**	**4**

the point anomalies in real time, while in the second step, the anomaly detection engine decides if these anomalies are contextual or collective based on different alarm profiles and system rules, as well as specific temperature profiles for each category of devices. Our proposed model efficiently determines context/collection based anomalies in real time, as reported in Table 3.

As expected, we noticed through experimentation that it is possible to obtain a higher accuracy on collective anomalies detection, but the number of false alarms triggered tends to be quite high. As shown in in Table 3, FSLSTM yields superior performance by correctly detecting 88% of the alarms of 1000 instances in this experiment with only nine false alarms. On the contextual side, FSLSTM achieves even a better performance of 90% with only four false alarms.

Prediction performance. For regression tasks, we test the performance of FSLSTM on the Energy Usage dataset to predict the building energy consumption on a window time of 600 minutes for two main reasons. First, the confidence interval of the model shows a high accuracy of energy prediction, as illustrated in Figure 9, which displays the actual vs. predicted building energy consumption using FSLSTM. Second, it makes more sense from an industrial perspective as 600 min equals 10 hours, which roughly represents a full working day. This time frame provides building

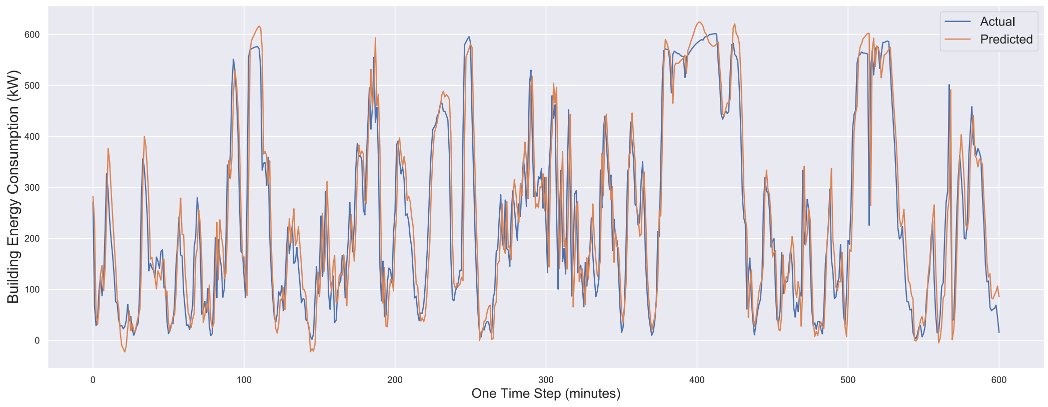

Fig. 9. Actual vs. predicted building energy consumption using FSLSTM on the Energy Usage test set.

managers with plenty of insightful information to anticipate and predict the daily usage of energy, and allows them to plan activities accordingly. As shown in Figure 9, the FSLSTM model has a stable performance and is able to predict energy consumption in the building with 90% accuracy.

For the sensors fault detection, we run the classification task on the Sensors dataset to detect anomalous devices. The experiment includes lights, thermostats of rooftop units and water leakage sensors for a window time ranging from 4 to 6 days. Figure 10 demonstrates the effectiveness of the FSLTM model in detecting outstanding and malfunctioning behaviors expressed by these sensors at different times of the day. As can be seen, the model is able to capture outliers with high accuracy, along with the corresponding value and time stamp. During working hours (light orange color), the weather temperature (dark gray color) shows an increase during the day due to sunlight and heat, while the site energy demand (green color) is stable during the majority of hours. As shown in Figure 10 (top), the weather temperature is higher during working hours than during non-working hours (white bars). During this window time, the site has stable energy demands ranging between 0 and 35 Kw. FSLSTM is capable of capturing an anomalous amount of energy demands, caused by a cluster of malfunctioning devices, highlighting it as an outlier in real time and eventually recording these devices with their manufacturing information and sending it back to the federated learning system. Then, the threshold determinator makes the appropriate decision and sends it over the BAS system for maintenance.

Communication Cost. One of the key challenges in our federated learning system is the communication cost, which is typically expressed as a function of data volume, e.g., Megabytes. As shown in Figure 11, the proposed FSLSTM model significantly reduces the communication cost compared to the baseline methods on the Sensors Event Log dataset. We argue that the superior performance of FSLSTM over the centralized LSTM and LR models is largely attributed to the effective participation of sensors in the training phase [40]. Federated learning models enable decentralized IoT devices to collaboratively learn a shared prediction model without sending the actual data that is generated during the specific round to the central server. This property of federated learning enhances the communication quality and reduces the cost, while keeping all the training data on device and hence reducing concerns on data security and privacy.

4.2 Ablation Study

To validate the effectiveness of our FSLSTM framework, we perform an ablation study under different configurations by changing the number of LSTM layers and retraining the models on the

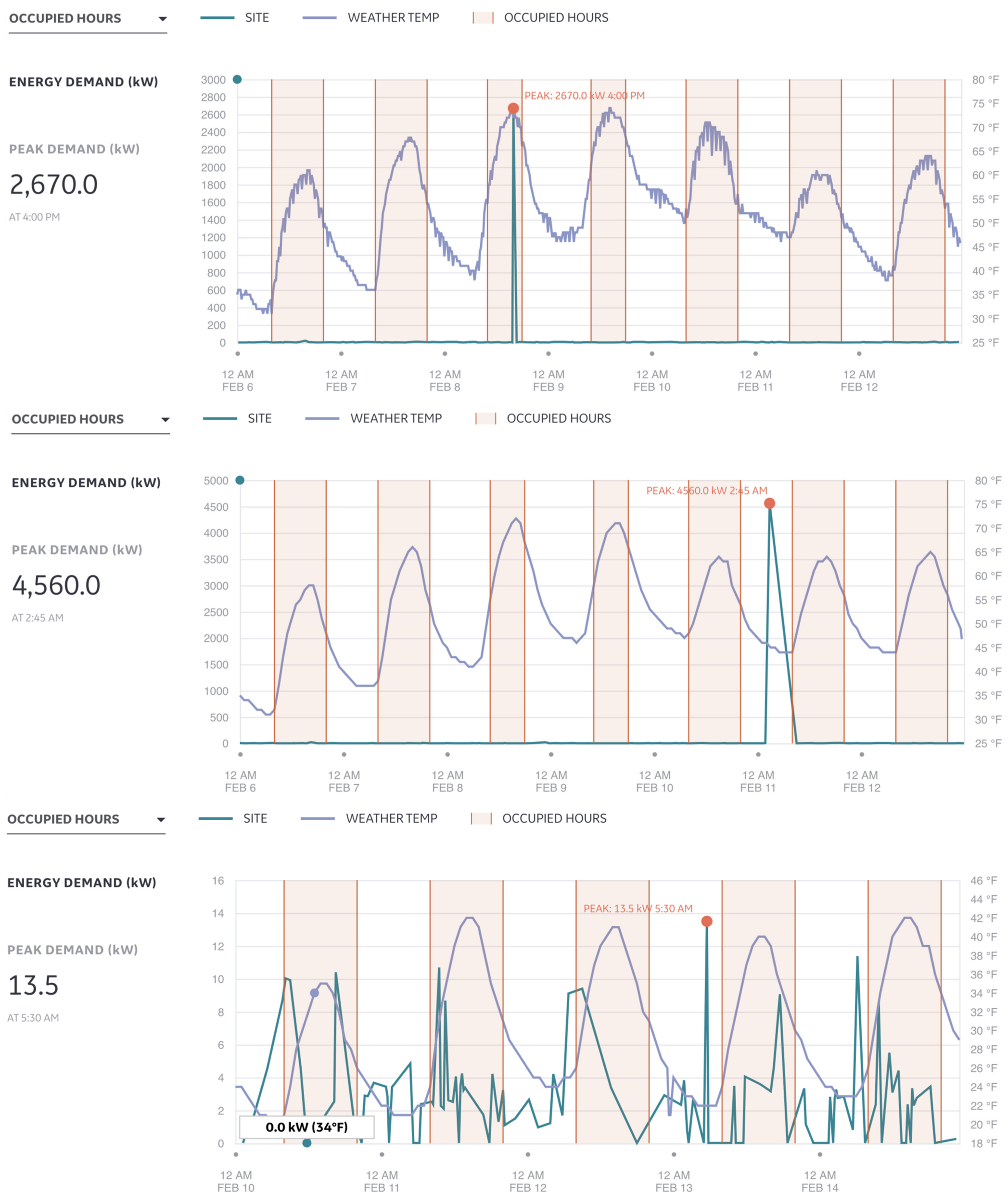

Fig. 10. Testing FSLSTM for anomaly detection on three types of sensors: lights, thermostats, and water leakage. The red dot indicates an anomaly.

same datasets. We design the FLSTM-1 and FSLSTM-2 models by removing one and two LSTM layers, respectively, from the proposed FLSTM architecture, which consists of three LSTM layers. The results are reported in Table 4, which shows that both FSLSTM-1 and FSLSTM-2 achieve better performance than LSTM and FGRU. It is important to note that with only a single LSTM layer, the federated LSTM model is able to outperform all baselines methods in terms of balanced accuracy and mean absolute error metrics. We also experimented with more than three LSTM layers, but we

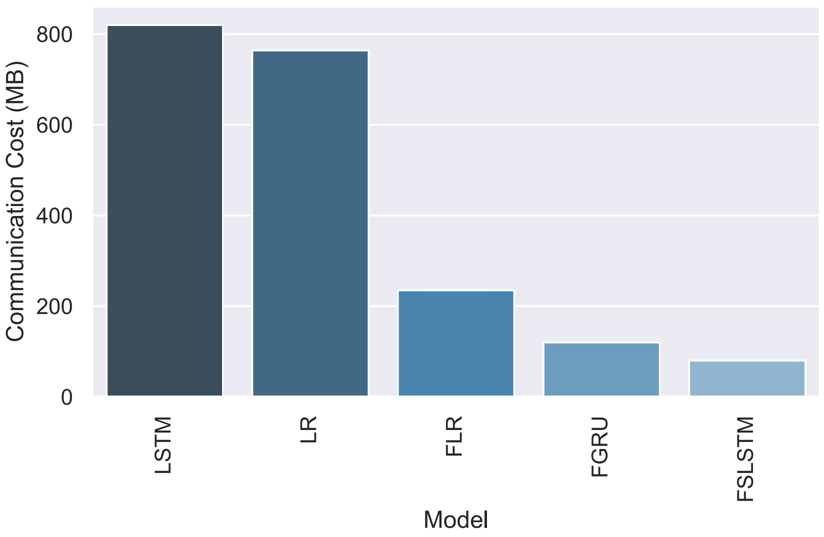

Fig. 11. Communication cost comparison between FSLSTM and baseline methods, all trained for 50 epochs.

Table 4. Ablation Study of FSLSTM on The Sensors Event Log and Energy Usage Test Sets Using Different LSTM Layers

	Sensors Data				EU Data		
Methods	Precision	Recall	F1	BA	MAE	MSE	RMSE
FLSTM-1	0.85	0.68	0.69	0.81	0.192	0.24	0.489
FLSTM-2	0.87	0.73	0.71	0.83	0.171	0.21	0.458
FSLSTM	**0.89**	**0.79**	**0.87**	**0.90**	**0.162**	**0.19**	**0.435**

Bold numbers indicate the best performance.

did not notice significant performance improvements. In addition, some small sensors (e.g., water leakage sensors) are unable to support the growing size of the model.

5 CONCLUSION

In this article, we introduced a federated stacked LSTM framework for anomaly detection in smart buildings using federated learning for IoT sensor data. The proposed FSLSTM network consists of a local LSTM model that captures individual sensors' data and a global model that aggregates the weights, updates the parameters and shares them again with sensors across all tasks. Experimental results on two datasets demonstrated FLSTM's ability to significantly improve the results of a variety of centralized models in IoT settings, achieving much better performance compared to baseline methods in both classification and regression tasks. We also showed that our federated stacked LSTM model converges twice as fast than the centralized LSTM during the training phase. In addition, we conducted an ablation study on our FSLSTM network to evaluate its performance and robustness under different configurations by changing the number of LSTM layers and retraining the models on the same datasets. In the future, we plan to investigate other IoT applications in a federated learning setting such as blockchain and electric vehicle charging networks.

REFERENCES

[1] Francisco Jose Bellido-Outeirino, Jose Maria Flores-Arias, Francisco Domingo-Perez, Aurora Gil-de Castro, and Antonio Moreno-Munoz. 2012. Building lighting automation through the integration of DALI with wireless sensor networks. *IEEE Trans. Consum. Electr.* 58, 1 (2012), 47–52.

[2] Alex Beltran and Alberto E. Cerpa. 2014. Optimal HVAC building control with occupancy prediction. In *Proceedings of the ACM Conference on Embedded Systems for Energy-Efficient Buildings.* 168–171.

[3] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. 2019. Towards federated learning at scale: System design. In *Proceedings of the Conference on Systems and Machine Learning.*

[4] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the ACM Conference on Computer and Communications Security.* 1175–1191.

[5] Rohitash Chandra and Sally Cripps. 2018. Bayesian multi-task learning for dynamic time series prediction. In *Proceedings of the IEEE International Joint Conference on Neural Networks.* 1–8.

[6] Junkun Chen, Xipeng Qiu, Pengfei Liu, and Xuanjing Huang. 2018. Meta multi-task learning for sequence modeling. *Neurocomputing* (2018).

[7] Ling-Jyh Chen, Yao-Hua Ho, Hsin-Hung Hsieh, Shih-Ting Huang, Hu-Cheng Lee, and Sachit Mahajan. 2018. ADF: An anomaly detection framework for large-scale PM2.5 sensing systems. *IEEE IoT J.* 5, 2 (2018), 559–570.

[8] Xu Chen, Lei Jiao, Wenzhong Li, and Xiaoming Fu. 2015. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Trans. Netw.* 24, 5 (2015), 2795–2808.

[9] Y. Chen, Y. Ning, Z. Chai, and H. Rangwala. 2019. Federated multi-task hierarchical attention model for sensor analytics. arXiv:1905.05142. Retrieved from https://arxiv.org/abs/1905.05142.

[10] Andrew Cook, Göksel Mısırlı, and Zhong Fan. 2019. Anomaly detection for IoT time-series data: A survey. *IEEE IoT J.* (2019).

[11] Simona D'Oca, Tianzhen Hong, and Jared Langevin. 2018. The human dimensions of energy use in buildings: A review. *Renew. Sust. Energy Rev.* 81 (2018), 731–742.

[12] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security.*

[13] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting Gradients—How easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems.*

[14] Dale L. Goodhue and Detmar W. Straub. 1991. Security concerns of system users: A study of perceptions of the adequacy of security. *Inf. Manage.* 20, 1 (1991), 13–27.

[15] Michael A. Hayes and Miriam A. M. Capretz. 2015. Contextual anomaly detection framework for big sensor data. *J. Big Data* 2, 1 (2015), 2.

[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9, 8 (1997), 1735–1780.

[17] Tsuyoshi Idé, Spiros Papadimitriou, and Michail Vlachos. 2007. Computing correlation anomaly scores using stochastic nearest neighbors. In *Proceedings of the IEEE International Conference on Data Mining.* 523–528.

[18] Ramtin Kazemi, Rein Vesilo, and Eryk Dutkiewicz. 2011. A novel genetic-fuzzy power controller with feedback for interference mitigation in wireless body area networks. In *Proceedings of the IEEE Vehicular Technology Conference.* 1–5.

[19] A. R. Khamesi, S. Silvestri, D. A. Baker, and A. De Paola. 2020. Perceived-value-driven optimization of energy consumption in smart homes. *ACM Trans. IoT* 1 (2020).

[20] Hokeun Kim, Eunsuk Kang, David Broman, and Edward A. Lee. 2020. Resilient authentication and authorization for the internet of things (IoT) using edge computing. *ACM Trans. IoT* 1 (2020).

[21] J. Konecný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, D. Bacon, and P. Richtárik. 2016. Federated learning: Strategies for improving communication efficiency. In *Proceedings of the NIPS Workshop on Private Multi-Party Machine Learning.* 322–334.

[22] Suyi Li, Yong Cheng, Yang Liu, Wei Wang, and Tianjian Chen. 2019. Abnormal client behavior detection in federated learning. In *Proceedings of the NeurIPS Workshop on Federated Learning for Data Privacy and Confidentiality.* 740–750.

[23] Suzhen Li, Yanjue Song, and Gongqi Zhou. 2018. Leak detection of water distribution pipeline subject to failure of socket joint based on acoustic emission and pattern recognition. *Measurement* 115 (2018), 39–44.

[24] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Sign. Process. Mag.* 37, 3 (2020), 5–60.

[25] P. P. Liang, T. Liu, L. Ziyin, R. Salakhutdinov, and L.-P. Morency. 2018. Think locally, act globally: Federated learning with local and global representations. In *Proceedings of the NeurIPS Workshop on Federated Learning for Data Privacy and Confidentiality.*

[26] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao. 2020. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Commun. Surv. Tutor.* 22, 3 (2020), 2031–2063.

[27] Li Ma, Nian Liu, Lingfeng Wang, Jianhua Zhang, Jinyong Lei, Zheng Zeng, Cheng Wang, and Minyang Cheng. 2016. Multi-party energy management for smart building cluster with PV systems using automatic demand response. *Energy Build.* 121 (2016), 11–21.

[28] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. 2018. Learning differentially private recurrent language models. In *Proceedings of the International Conference on Learning Representations.*

[29] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the International Conference on Artificial Intelligence and Statistics.*

[30] Harvey B. Mitchell. 2007. *Multi-Sensor Data Fusion: An Introduction.* Springer Science & Business Media.

[31] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. 2019. Agnostic federated learning. In *Proceedings of the International Conference on Machine Learning.*

[32] A. Mosenia and N. K. Jha. 2017. A comprehensive study of security of internet-of-things. *IEEE Trans. Emerg. Top. Comput.* 5 (2017), 586–602.

[33] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N. Asokan, and Ahmad-Reza Sadeghi. 2019. DÏoT: A federated self-learning anomaly detection system for IoT. In *Proceedings of the IEEE International Conference on Distributed Computing Systems.* 756–767.

[34] H. H. Pajouh, R. Javidan, R. Khayami, A. Dehghantanha, and K.-K. R. Choo. 2019. A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in iot backbone networks. *IEEE Trans. Emerg. Top. Comput.* 7 (2019), 314–323.

[35] C. Perera, C. H. Liu, and S. Jayawardena. 2015. The emerging internet of things marketplace from an industrial perspective: A survey. *IEEE Trans. Emerg. Top. Comput.* 3 (2015), 585–598.

[36] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konecny, Sanjiv Kumar, and H. Brendan McMahan. 2021. Adaptive federated optimization. In *Proceedings of the International Conference on Learning Representations.*

[37] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. 2018. A generic framework for privacy preserving deep learning. In *Proceedings of the NeurIPS Workshop in Privacy Preserving Machine Learning.*

[38] Giulia Violatto, Ashish Pandharipande, Shuai Li, and Luca Schenato. 2019. Classification of occupancy sensor anomalies in connected indoor lighting systems. *IEEE IoT J.* 6, 4 (2019), 7175–7182.

[39] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. 2020. Federated learning with matched averaging. In *Proceedings of the International Conference on Learning Representations.*

[40] Xin Yao, Chaofeng Huang, and Lifeng Sun. 2018. Two-stream federated learning: Reduce the communication costs. In *Proceedings of the IEEE Visual Communications and Image Processing.* 1–4.

[41] Liang Yu, Di Xie, Tao Jiang, Yulong Zou, and Kun Wang. 2017. Distributed real-time HVAC control for cost-efficient commercial buildings under smart grid environment. *IEEE IoT J.* 5, 1 (2017), 44–55.

[42] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. 2019. Bayesian nonparametric federated learning of neural networks. In *Proceedings of the International Conference on Machine Learning.*

[43] Weishan Zhang, Wuwu Guo, Xin Liu, Yan Liu, Jiehan Zhou, Bo Li, Qinghua Lu, and Su Yang. 2018. LSTM-based analysis of industrial IoT equipment. *IEEE Access* 6 (2018), 23551–23560.

[44] Y. Zhao, J. Chen, and D. Wu. 2019. Multi-task network anomaly detection using federated learning. In *Proceedings of the International Symposium on Information and Communication Technology.*

[45] Yu Zheng, Huichu Zhang, and Yong Yu. 2015. Detecting collective anomalies from multiple spatio-temporal datasets across different domains. In *Proceedings of the SIGSPATIAL International Conference on Advances in Geographic Information Systems.* 1–10.

[46] Konglin Zhu, Zhicheng Chen, Yuyang Peng, and Lin Zhang. 2019. Mobile edge assisted literal multi-dimensional anomaly detection of in-vehicle network using LSTM. *IEEE Trans. Vehic. Technol.* 68, 5 (2019), 4275–4284.

[47] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. In *Advances in Neural Information Processing Systems.*

Received May 2020; revised April 2021; accepted May 2021