EAI Endorsed Transactions
on Security and Safety

Research Article EAI.EU

# FedADMP: A Joint Anomaly Detection and Mobility Prediction Framework via Federated Learning

Zezhang Yang[1], Jian Li[1,*], Ping Yang[2]

[1]Department of Electrical and Computer Engineering, Binghamton University, State University of New York, Binghamton, NY 13902
[2]Department of Computer Science, Binghamton University, State University of New York, Binghamton, NY 13902

## Abstract

With the proliferation of mobile devices and smart cameras, detecting anomalies and predicting their mobility are critical for enhancing safety in ubiquitous computing systems. Due to data privacy regulations and limited communication bandwidth, it is infeasible to collect, transmit, and store all data from mobile devices at a central location. To overcome this challenge, we propose FedADMP, a federated learning based joint Anomaly Detection and Mobility Prediction framework. FedADMP adaptively splits the training process between the server and clients to reduce computation loads on clients. To protect the privacy of user data, clients in FedADMP upload only intermediate model parameters to the cloud server. We also develop a differential privacy method to prevent the cloud server and external attackers from inferring private information during the model upload procedure. Extensive experiments using real-world datasets show that FedADMP consistently outperforms existing methods.

Received on 07 August 2021; accepted on 05 October 2021; published on 21 October 2021

**Keywords:** Federated learning, Anomaly detection, Mobility prediction, Privacy-preserving system



doi:10.4108/eai.21-10-2021.171595

## 1. Introduction

With the proliferation of mobile devices and smart cameras, anomaly detection and mobility prediction become an important and emerging topic in ubiquitous computing systems, where an anomaly can be a human, a car, and so on. In many situations, especially when solving crime cases, anomaly detection and mobility predication are indispensable. Historically, solving crime cases has been the prerogative of criminal justice and law enforcement experts. With the increasing use of computer systems to track and identify crimes, data analysts have begun to help law enforcement officers speed up the process of solving crime cases [1]. For example, accurate anomaly detection and mobility prediction can help the authority quickly identify the potential threat (i.e., the anomaly) and its movement paths (i.e., the mobility) so as to schedule forces to meet the requirement (e.g., taking the threat into custody) with minimum delay. For smart cities, with the help of anomaly detection and mobility prediction, the government can better understand the traffic flows and propose proper policies to run the city more efficiently.

Given its great value in practical applications, there are a large number of studies on anomaly detection (e.g., [2–6]) and mobility prediction (e.g., [7–10]). While these models achieve good performance, they are in general designed for a single task, i.e., either anomaly detection or mobility prediction. The intrinsic design makes it hard to generalize these models for the purpose of joint anomaly detection and mobility prediction in ubiquitous computing systems subject to its increasingly demanding services. Furthermore, there is an increasing concern on privacy issues raised in these models due to the data sharing between client devices and the central server.

Federated learning (FL) [11–14] has recently been proposed for decentralized privacy-preserving training, which enables client devices, such as mobile phones and smart cameras, to collaboratively learn a shared (global) model while keeping all the training data on client devices. There is a central server in FL that

*Corresponding author. Email: lij@binghamton.edu

orchestrates the whole training process. In each training round, the server collects local models from eligible client devices, which are then averaged to improve the shared model. Similar to the conventional centralized machine learning framework, the *wall-clock time* for training a model to reach a certain accuracy (i.e., time-to-accuracy) is a key performance objective. FL has been deployed across user devices for computer vision and natural language processing tasks [15, 16], medical imaging AI creation [17], video streaming [18], imaging training and AI cameras testing in smart cities [19, 20].

Despite the above success, FL cannot be directly applied for joint anomaly detection and mobility prediction in ubiquitous computing systems due to the following challenges. First, model training through e.g., deep neural network (DNN) is known to be extremely computation-intensive. Thus, it would be infeasible to train the model on client devices alone, which often have limited resources for emerging applications in ubiquitous computing systems. Second, the model uploading and aggregation process in FL are not secure, which may result in the leak of private information of individuals. This is because the central server and external attackers may be able to obtain user's private information by analyzing the uploaded model weights. Finally, the application of privacy protection via e.g., differential privacy method [21] may lead to significant performance degradation in joint anomaly detection and mobility prediction task. Adding to these challenges is the fact that FL testing is usually performed on real-life client data, some of the client devices may not be available for the FL training and testing in ubiquitous computing systems [12, 22], and the system performance of client devices often varies. As a result, a robust FL model is desired for ubiquitous computing systems where not all client devices are available.

**Research contributions:** In this paper, we propose FedADMP, a FL-based joint anomaly detection and mobility prediction (ADMP) framework, to address the aforementioned challenges. First, FedADMP protects the privacy of client devices. In FedADMP, the raw data is stored at the client devices and only intermediate model parameters such as the gradients of the trained neural network is uploaded to the server. Second, to reduce the computation load on client devices, FedADMP provides a novel joint training process in which the training task is performed collaboratively by client devices and the cloud server in each epoch. In contrast, the conventional FL framework performs the model training only on client devices. In addition, we consider a practical attack scenario and develop a lightweight group activation mechanism to protect client devices from such an attack without performance degradation. Finally, FedADMP works on situations where only partial client devices are available for FL training and testing.

In summary, the main contributions of the paper are:

- To the best of our knowledge, FedADMP is the first work that addresses the joint anomaly detection and mobility prediction problem using an FL-based approach. By storing users' private data on client devices and uploading only model parameters to the server, FedADMP successfully protects the privacy of client devices.
- FedADMP provides *a novel joint training process* between client devices and the cloud server in each epoch to reduce the computation load on client devices, which often have limited computational capability in ubiquitous computing systems. In particular, each client device performs partial training using its raw data and then adaptively offloads the remaining training to the cloud server by uploading its intermediate results i.e., the activation of LSTM to the cloud server. Our extensive experiments using real-world traces show that FedADMP not only dramatically reduces the convergence time to achieve a certain accuracy, but also reduces the energy consumption on client devices and the communication costs, which are desirable for resource-constrained client devices in ubiquitous computing systems.
- We consider a practical attack scenario and develop a lightweight group activation mechanism to protect client devices from such attack.
- We strengthen the robustness of our proposed FedADMP framework by considering the situation in which only partial client devices are available for training and testing. Our experimental results show that FedADMP achieves the desired time-to-accuracy with reduced energy consumption and communication costs when partial client devices contribute to the model training.

The rest of the paper is organized as follows. We formulate the joint anomaly detection and mobility prediction problem in Section 2. Section 3 presents the design of our FedADMP system design. The experimental results of FedADMP on five real-world datasets are given in Section 4. Section 5 discusses the related work and Section 6 concludes the paper. Additional experimental results are provided in the appendix.

## 2. Preliminaries

In this section, we provide a brief overview of the joint ADMP problem in ubiquitous computing systems. With

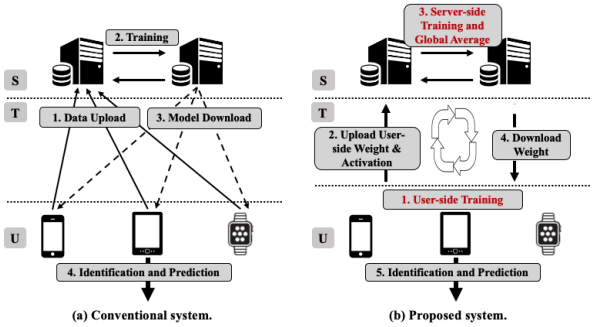


**Figure 1.** Workflow comparison between conventional machine learning systems and FedADMP. "$S$", "$T$" and "$U$" represent cloud servers, transfer environment, and end users, respectively.

the advance of location based applications and largely-deployed smart cameras, location-based services have played a significant role in the safety of smart cities [8, 23, 24]. Below, we give a generation definition of the ADMP data trajectory.

**Definition 1.** (Data trajectory) Let $q$ represent data that is recorded as a tuple of four elements: image pixel $m$, location $l$, timestamp $t$, user identity $u$ (i.e., $q = (m, l, t, u)$). An ADMP trajectory $\mathcal{S}$ of target $u$ is defined as a set of ordered image sequences $\{q_1, \cdots, q_n\}$ (i.e., $\mathcal{S}_u = \{(m_1, l_1, t_1), \cdots, (m_n, l_n, t_n)\}$). We use $\mathcal{S}$ to denote $\{\mathcal{S}_u\}_{u \in \mathcal{U}}$, where $\mathcal{U}$ is the set of all targets.

Note that the above definition is applicable to general application domains [23, 25–27]. For simplicity, we transform all location information into a new unique ID, and quantify the time interval into fixed value, following the recent practice [10, 24]. In our paper, we choose 250 seconds as the default time interval given the consideration of anomaly mobility and general location frequency in smart city services. However, such spatial and temporal resolution can be easily generalized based on the requirement of different services.

**Definition 2.** (ADMP) Given an image sequence $\mathcal{S}$, the goal of ADMP is to detect a particular target $u$ and predict the possible location $l_{n+1}$ for target $u$ in the next time step $t_{n+1}$.

## 3. FedADMP System Design

In this section, we first provide an overview of FedADMP and then describe each core component of FedADMP. Similar to many other conventional federated learning frameworks, we assume that the raw data is collected by each client, rather than being collected at a central location.

Figure 1 compares the workflow of conventional machine learning systems and FedADMP on anomaly detection and mobility prediction. As illustrated in Figure 1 (a), conventional machine learning systems perform the following four steps to solve the joint anomaly detection and mobility prediction problem. First, the raw data is collected from client devices and transmitted to a central server, which usually resides in a remote data center. Next, a joint model is trained on the server using the raw data. The trained model is then distributed to the client devices through communication channels. Finally, the client devices infer the anomaly and the corresponding mobility behavior. Although such a centralized model works well on conventional systems where client devices have limited computation resources, it fails to protect users' privacy data as raw user data is uploaded to the cloud server. Recently, FL was proposed to address this issue by performing all computations (e.g., model training) on client devices so that no raw user data is shared with the cloud server. However, client devices usually have limited computation resources and storage, particularly for emerging applications in ubiquitous computing systems, which may significantly degrade the system performance.

To address the above challenges, we propose FedADMP, a novel FL based model for joint anomaly detection and mobility prediction in ubiquitous computing systems. The workflow of FedADMP is given in Figure 1 (b). In FedADMP, client devices and the cloud server collaboratively train the model in each epoch. We assume that all clients use the Long Short Term Memory (LSTM) model [28] for training. Each client device first partially trains the model and then adaptively offloads the model training to the cloud server while keeping the sensitive user data on the device. Specifically, during the forward propagation process, each client device uploads only *the activation* (i.e., the intermediate output of the first layer of LSTM to the cloud server, which further processes the forward propagation based on the activation received from each client. Then the cloud server computes the gradient of the loss function in the backward propagation, which is used to update the model respect to each client. Once the server completes processing all clients in each epoch, a global aggregation is performed to compute the updated global model, which is then sent to all clients. As the client device performs only partial training, FedADMP significantly reduces the computation loads on client device. Furthermore, in departure of the conventional split learning methods (e.g., [29, 30]), the client devices in FedADMP uploads only the activation rather than the whole local model, to the cloud server. This significantly reduces the communication cost as the size of the activation is significantly smaller than the whole model. In addition, we have developed a differential privacy based method to prevent external attackers and the cloud server from extracting clients' personal information from the uploaded activations.

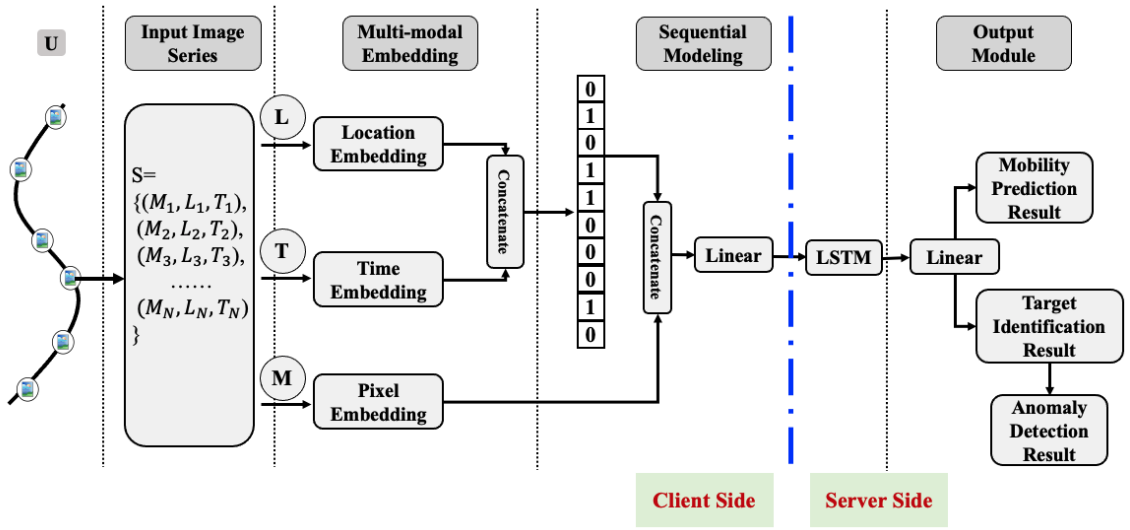


**Figure 2.** The architecture of FedADMP.

## 3.1. Anomaly Detection and Mobility Prediction

Figure 2 gives the architecture of FedADMP, which consists of three components: an input module with multi-modal embedding, an LSTM-based sequential module, and an output module.

**Input module with multi-modal embedding:** In FedADMP, we convert the raw data into a sequence $\mathcal{S}$ based on Definition 1, where $\mathcal{S}_u$ represents a sequence of images captured by client devices (e.g., smart cameras, mobile devices, etc.). As there are thousands of locations in the dataset, the dimension of one location can be up to thousands. To address this issue, we use embedding methods to reduce the feature dimension and learn the dense representation of discrete location, time, and pixel values. Similar to conventional embedding methods such as [24, 31, 32], the embedding table is only a lookup table with dense representation of difference indexes, which significantly reduces the dimension. To improve the performance, the embedding model is trained and optimized with the whole network training process. Finally, we concatenate the location, the time, and the pixel vectors to obtain a representation of a spatial-temporal point.

**Sequential modeling unit:** Given the tremendous success of recurrent networks, in particular the LSTM in sequential modeling, we develop a LSTM-based modeling unit for sequential transition relationships. Our modeling unit combines the last output of the neural network and the current input to make the network learn to capture the relationship between sequential inputs. LSTM is formulated as follows:

$$i_t = \sigma(W_{ix}x_t + W_{gh}h_{t_1} + b_i),$$
$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t_1} + b_f),$$

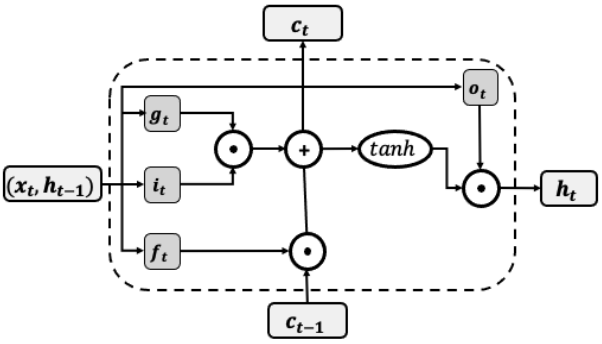


**Figure 3.** The computation flow of LSTM.

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t_1} + b_o),$$
$$g_t = tanh(W_{gx}x_t + W_{gh}h_{t_1} + b_g),$$
$$c_t = f_t \circ c_{t-1} + i_t \circ g_t,$$
$$h_t = o_t \circ tanh(c_t),$$

where $\circ$ denotes the Hadamard product, $x_t$ is the input, $h_t$ is the hidden state, $c_t$ is the cell state, $i_t, f_t, o_t$ are the input, forget and output gates, and $g_t$ is the useful information from the input. The computation flow of such a LSTM is illustrated in Figure 3.

**Output module:** Following the above sequential module, we consider a projection-based method for output module (shown in Figure 4), which directly uses the hidden state vector to calculate the correlation between the hidden state and the dense location embedding representation. Given the correlation results, we use the *sigmoid* function to obtain the final hybrid output which contains the identification probability distribution concatenated with the prediction probability distribution. We use a linear layer to directly project the hidden state onto

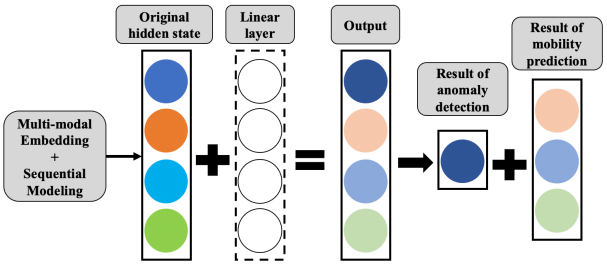


**Figure 4.** The output module.

high-dimensional identification and position vectors. The sigmoid function is then applied to the projection output to obtain a probability distribution of the identified task and the predicted position. The output module is formulated as

$$z = wh + b, \quad \text{output} = \frac{exp(z)}{exp(z) + 1},$$

where $h$ is the hidden state of previous sequential unit, $w$ and $b$ represent learnable parameters of projection linear layer.

Given the above hybrid output that contains the concatenated result of task identification and mobility prediction, we can simply separate them through slicing to obtain the results for the task identification and the mobility prediction. We further determine if the identification task is an anomaly or not. Let $y_d$ be the detection output. We compute the normal score as $s = (y_d - y_l)^2$, where $y_l$ is the one hot code information of the target's label. Once we obtain the score $s$, we compare it with a decision making threshold $\delta$. If $s$ is less than $\delta$, then an anomaly is considered to be detected.

## 3.2. Privacy-Preserving Mechanism

Although FedADMP does not upload raw data to the cloud server, it is still possible to divulge sensitive information collected by client devices by analyzing the differences between the activation and the shared global model transmitted between client devices and the cloud in each epoch [33, 34].

**A real attack example [23]:** Although it is much harder to extract private information from the activation and the shared global model than the raw data due to the complexity and the implicit nature of the model, advanced techniques were recently developed to attack the model upload procedure and extract private information with some prior knowledge [35]. For example, assume that a client device receives a global model $M^{t-1}$ at time step $t-1$ from the cloud server. Based on this global model and the local data used for training, the client device obtains a new local model $M^t$ at the step $t$. Since the raw data from different client devices are often not independent and

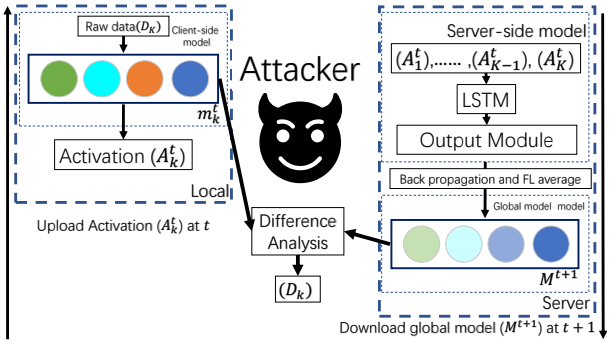


**Figure 5.** A real attack scenario where attackers may extract private information from the model upload procedure.

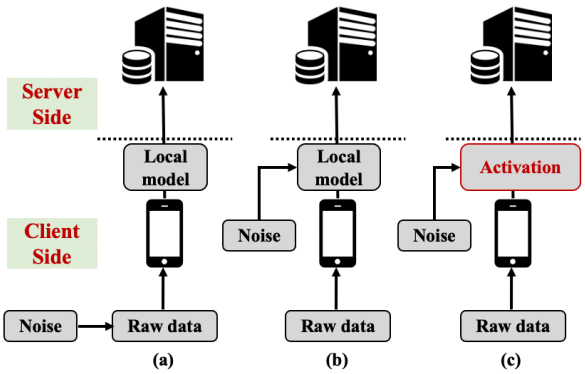


**Figure 6.** Different differential privacy mechanisms.

identically distributed in practice, the contribution of each client device is varying. Therefore, by comparing the differences between the global model $M^{t-1}$ and the local model $M^t$, the attacker can deduce which client is involved in the training process. A simple attack of the above procedure is presented in Figure 5.

**Activation optimization with differential privacy:** To mitigate the potential privacy risk, we propose a privacy-preserving local activation technique on client devices based on the differential privacy (DP) method [21]. We introduce DP into the local activation optimization to obtain the controlled embedding table for model sharing with privacy guarantees. Figure 6 compares our proposed activation optimization method against the conventional DP methods. In conventional FL based framework, the noise is added either to the raw data (Figure 6 (a)) or the the entire local model (Figure 6 (b)). However, this is usually computational expensive due to the sheer size of raw data and local model, particularly for client devices in ubiquitous computing systems considered in this paper. In departure from these methods, FedADMP adds artificial noise only to the activation uploaded to the cloud server (Figure 6 (c)).

**Computing the global model:** After client devices upload the activation to the cloud server, the cloud

**Algorithm 1** The training procedure of FedADMP

1: **Global parameters:** $M^t$: global model at $t$-step; $K$: number of clients; $\alpha$: learning rate.
2: **Local parameters:** Client $k$, $m_k$: local model; $\epsilon$: DP parameter; $A_k$: activation.
3: **Server:** initialize $M^0$
4: **for** round $t \in \{1, 2, ...\}$ **do**
5: **for** client $k \in \{1, 2, ..., K\}$ **do**
6: $A_k^t \leftarrow$ **Client**$(M^t, \alpha)$
7: complete the rest of training
8: $m_k^{t+1} \leftarrow M^t - \alpha \nabla M^t$
9: **end for**
10: $M^{t+1} = \sum_{n=1}^{K} m_k^{t+1}/K$
11: **end for**
12: **Client:**
13: $A_k^t \leftarrow$ forward propagation //construct noise with DP $\epsilon$
14: $A_k^t \leftarrow A_k^t$ + noise // apply DP
15: upload $A_k^t$ to the server

server continues the training process as described in Figure 1. Upon the completion of the training process in each epoch, the server aggregates results from all client devices. Here we choose the widely used Adaptive Moment Estimation (Aadm [36]) algorithm to update the model in FedADMP. Assume that there are $k$ client devices involved in a task. Client $k$ sends its activation $A_k^t$ to the server after completing the client-side training, and the server will use $A_k^t$ to complete the rest of the forward propagation. Then the server updates the local model $m_k^{t+1}$ of client $k$ via $m_k^{t+1} \leftarrow M^t - \alpha \nabla M^t$. Once this is done for all clients, the server updates the new global model by aggregation: $M^{t+1} = \sum_{n=1}^{K} m_k^{t+1}/K$. Finally, the server distributes the new global model $M^{t+1}$ to all clients for the next epoch.

### 3.3. Training Procedure

Algorithm 1 summarizes our training process. We use the cross-entropy loss function for local training on client devices and the basic SGD algorithm for global optimization. Lines 3–11 in the algorithm describes the training process performed on the server side. First, the server randomly generates an initial parameter $M_0$ (line 3). The server then process the activation received from client devices (line 6) and completes the rest of the training process (line 7). Next, the server updates the local model (line 8). After all local models are updated, the server computes the global model and sends the global model to all client devices to start a new epoch (line 9). At the client side, each client performs local training on its raw data, adds noise to the activation to protect privacy, and then uploads the activation to the server (lines 12 and 13).

### 3.4. Discussion: Adversarial Attacks on FedADMP

A few adversarial attacks were proposed on federated learning model, including a data poisoning [37] in which malicious participants aim to poison the global model by sending model updates derived from mislabeled data, a backdoor attack [38] where the goal of the adversary is to reduce the performance of the model on targeted tasks, the inference attack [39] which reconstructs the training samples by comparing generative deep neural networks with discriminative deep neural networks to generate samples that appear to come from the training set, the server side attack [40] which uses GAN with a multitask discriminator to recover private data. FedADMP is not resistant to the above attacks. We can apply existing defense mechanisms such as the ones proposed in [41], to defend against adversarial machine learning attacks.

## 4. Experimental Results

In this section, we evaluate the performance of FedADMP using real-world datasets. All experiments were conducted on an Intel(R) core i7-9750H machine with 2.6GHz 6-core CPU, GTX 1660Ti GPU, and 16GB memory.

### 4.1. Datasets

We evaluate the performance of FedADMP using five data trajectories from the following three datasets.
**Geo-life dataset** [25] contains real GPS trajectories of hundreds of users from several cities collected between April 2007 and August 2012. We select the trajectories of three different cities – Beijing, Krong Siem Reap (KSR) and Seattle – to conduct the experiments.
**Foursquare dataset** [26] contains trajectories of location based social networks, which attract millions of users and contain their massive digital footprints. In particular, we select the trajectory of New York City (NYC) Restaurant, which contains the check-in, tip and tag data of restaurant venues in NYC collected between 24 October 2011 and 20 February 2012, from 3,112 users and 3,298 venues with 27,149 check-ins and 10,377 tips.
**Cabspotting dataset** [27] contains GPS traces of taxi cabs in San Francisco (SF), collected in May 2008.

To better understand the characteristics of the data, we analyze the data and draw their distributions in Figure 7. We observe that the time interval in most of the datasets is 5 seconds. We also observe that the visiting frequency of all locations follows similar long-tail distribution with Beijing trajectory having slightly higher visiting frequency.

Our work focuses on identifying suspicious objects and predicting their future locations in ubiquitous computing systems. However, the above mobility

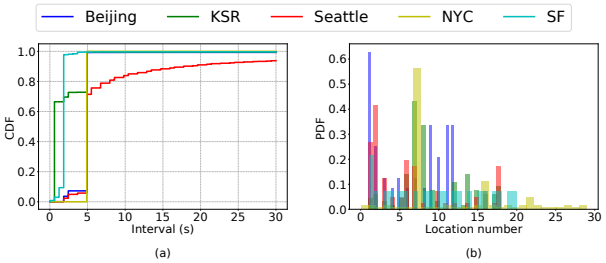


**Figure 7.** Characteristic of datasets. (a): Distribution of time interval of nearby location records. (b): Visiting frequency distribution of locations.

datasets do not contain data for object recognition and hence cannot be directly used for our task. To address this issue, we perform several pre-processing steps on the above mobility datasets together with the insights from two object recognition databases: MNIST datasets [42] and Fashion-MNIST datasets [43]. In particular, we reconstruct the mobility datasets so that each data contains the four components describe in Section 2: image, time, location, and ID. After the reconstruction, each object in the recognition database corresponds to a specific ID in the mobility database. Since the trajectory data of the Geo-life database is expressed in latitude and longitude, to make the experiment practical, we follow the recent practice [10, 24] to transform various location identifications into new unique location IDs. We choose 250 seconds as the time interval in our experiments. For the ease of exposition, we present only results for using the MNIST dataset as the object recognition database. Similar trends are observed with the Fashion-MNIST dataset, which are relegated to the Appendix A.

## 4.2. Baselines and Metrics

We compare FedADMP with state-of-the-art methods in terms of convergence, prediction accuracy, communication costs, and energy consumption on client devices. To achieve high precision, we focus on evaluating the convergence of different models in terms of *wall-clock time* instead of the number of iterations. This is because the model training and task execution can be time consuming, and achieving a high precision in a timely manner is of significant importance for ADMP to maintain safety in ubiquitous computing systems. For completeness, we also present the results in term of the number of iterations to show the advantange of FedADMP. We compare FedADMP against the following four state-of-the-art methods:

- **PMF** [23] is a FL-based privacy preserving mobility prediction framework with all computations done on client devices.
- **LSTM** is the simplified version of PMF model, which only contains LSTM layers.
- **FedMA** [44] constructs the shared global model in a layer-wise manner by matching and averaging hidden elements with similar feature extraction signatures.
- **FedAvg** [11] is a state-of-the-art method for distributed training with privacy guarantees.

## 4.3. Parameters

The parameters of FedADMP are as follows: (a) model parameters: the size of the location embedding is 12 for Beijing trajectory, 18 for KSR trajectory, 19 for Seattle trajectory, 27 for NYC trajectory and 20 for SF trajectory. The size of the time embedding is 28 and the size of the hidden state is 128. (b) Hyperparameters for local training model: the learning rate is 0.001 and the dropout rate is 0.5. (c) Hyperparameters for global training model: the learning rate is 0.001 and the dropout rate is 0.5. (d) Hyperparameters for global aggregation model: the number of client devices is 30, the local epoch is 1. Furthermore, we consider an ideal setting for the cloud server, i.e., there are always enough computational resources on the cloud server.

## 4.4. Performance results

In this section, we compare the performance of FedADMP against the above four state-of-the-art methods on five data trajectories. We observe that FedADMP dramatically reduces the time taken to achieve a certain accuracy level with significantly reduced energy consumption on client devices and no additional communication costs.

**Convergence and accuracy:** Figures 8 and 9 compare the convergence and model accuracy of FedADMP against the above four state-of-the-art methods using time-to-accuracy and epoch-to-accuracy metrics, respectively, where the MNIST dataset is used as the object recognition database. In particular, when an anomaly appears in the system, FedADMP first identifies the anomaly and then predicts its mobility. The accuracy presented in the paper is the joint accuracy of anomaly detection and mobility prediction. Note that in most cases, the ground truth of the anomaly is not available and hence are hard to be directly quantified. However, once the targeted anomaly appears in the time series data, FedADMP can learn to predict its next location and detect/identify the anomaly accordingly.

Figures 8 and 9 show that FedADMP consistently outperforms other four state-of-the-art methods although the accuracy improvement over PMF is limited in some traces. More importantly, FedADMP dramatically reduces the convergence time to achieve a particular accuracy. For example, FedAMDP achieves 60% accuracy within 10 seconds in the Seattle trajectory, while the best performed baseline (i.e., the PMF) achieves on

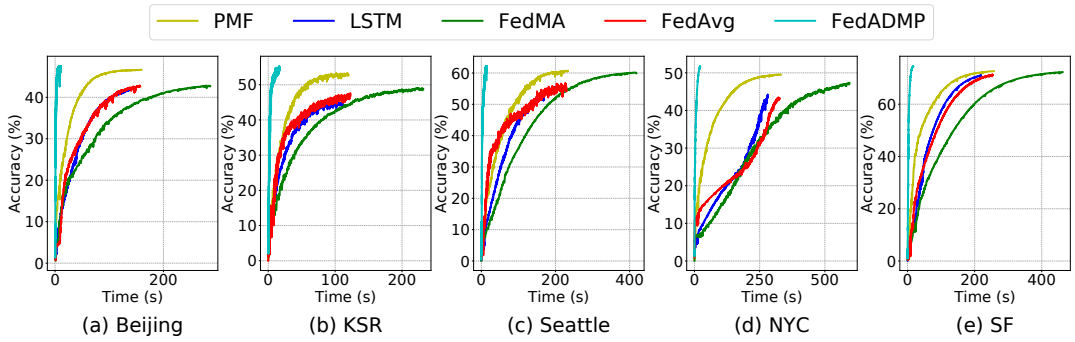


**Figure 8.** Convergence and accuracy performance measure in time (i.e., time-to-accuracy).

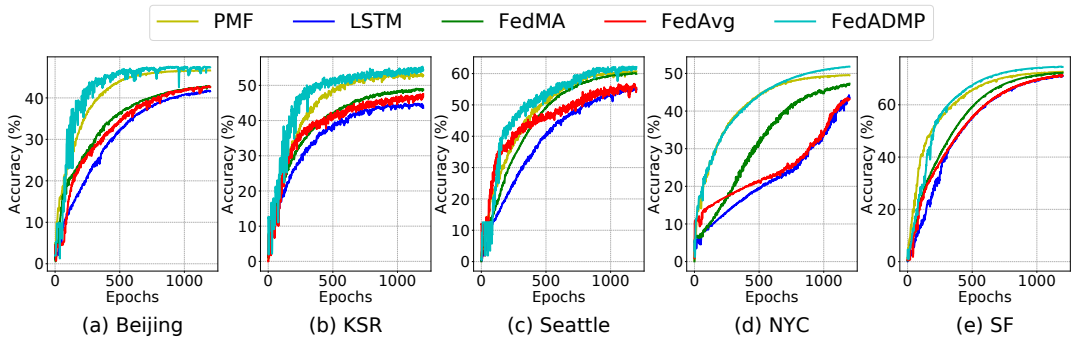


**Figure 9.** Convergence and accuracy performance measure in epochs (i.e., epoch-to-accuracy).

15% accuracy (as shown in Figure 10). Similar observations are made in other trajectories. This property is highly desirable for the joint anomaly detection and mobility prediction task in ubiquitous computing systems since a timely decision is critical in many emerging applications as motivated in the introduction. Similar results are observed when the Fashion-MNIST dataset is used as the object recognition database; the corresponding results are provided in Appendix A for the ease of exposition.

**Energy consumption and communication costs:** As client devices in ubiquitous computing systems are usually battery-powered and resource constrained, it is important to keep the energy consumption and communication costs low in order not to quickly drain the battery of client devices. We compare the energy consumed by client devices in different models for achieving the same accuracy. In our experiments, we assume an ideal cloud server (e.g., in data center) that always has enough computation resources. Figure 11 *(Left)* compares the energy consumed by client devices in different models when achieving 40% accuracy. The figure shows that FedADMP consumes the least energy. This is because FedADMP offloads part of the training tasks to the server, as described in Section 3. We also measure the size of the data transmitted between the server and clients. In our experiments, all clients are executed in parallel on the machine. As a result, the communication environment is ideal and the communication time between the server and clients are negligible. When the server and clients execute on different machines, the communication time depends on the size of data transmitted between the server and clients. Figure 11 *(Right)* presents the size of data transmitted between the server and clients. As client devices in FedADMP send only the activation rather than the entire model parameters to the server, the size of data transmitted between the server and clients is small. Similar observations can be made for other performance accuracy and hence are omitted here.

**Impact of the number of clients:** Although collecting information from a large number of clients helps improve the accuracy of anomaly detection, it also increases the search space for mobility prediction. Figure 12 (a) shows that within the same amount of training time, when the number of clients increases, the model accuracy reduces slightly. This is because, when the number of clients increases, the number of training epochs increases. On the other hand, for a certain accuracy (e.g., 40%), it will take less time to achieve such an accuracy as more data is used for training at each epoch when the number of clients increases, as shown in Figure 12 (b).

**Impact of differential privacy parameters:** We also investigate the impact of differential privacy parameter $\epsilon$ on the performance of FedADMP. Figure 13 shows that the accuracy of FedADMP increases when $\epsilon$ increases $\epsilon$ (i.e., when the differential privacy

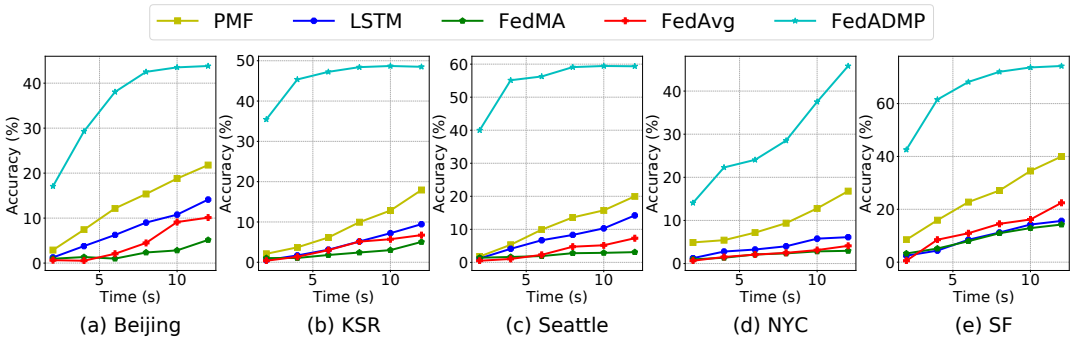


**Figure 10.** Accuracy performance measured in time.

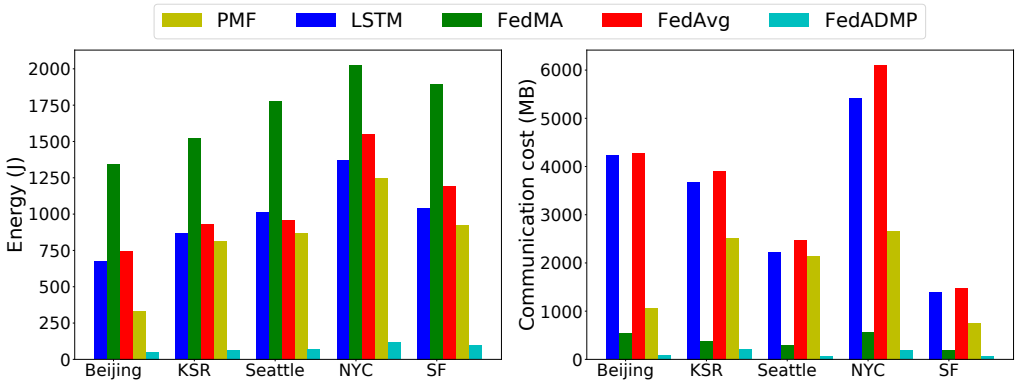


**Figure 11.** *(Left):* Energy consumption. *(Right):* Communication cost.

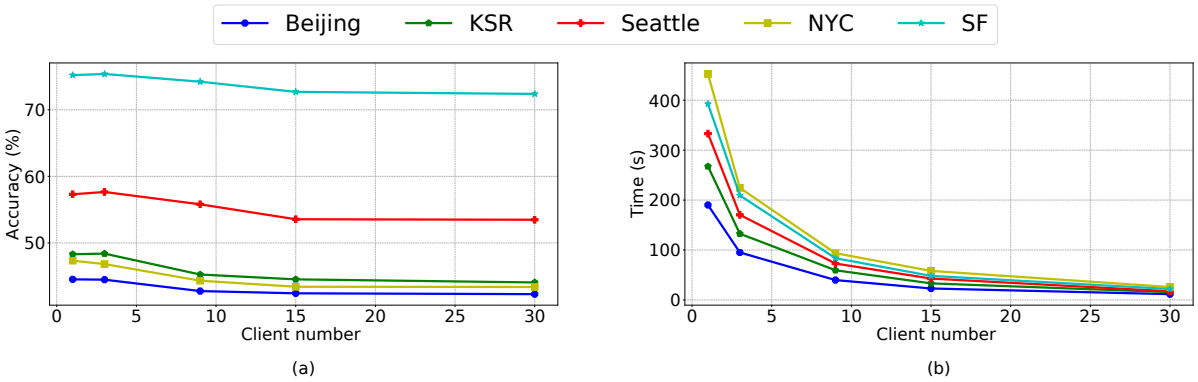


**Figure 12.** The impact of different numbers of client devices.

requirement is less strict). We also observe that, when $\epsilon$ is small (i.e., less than 40), the accuracy of FedADMP increases dramatically when $\epsilon$ increases slightly, especially for the SF trajectory. This is because the SF trajectory has more location ID numbers than others. As the relation between locations is much more diverse with more features for the trajectory data, the larger the number of location ID numbers, the less ambiguous the trajectory is. When $\epsilon$ is greater than 40, the accuracy of FedADMP is less sensitive to the DP, i.e., the marginal improvement is smaller with the change of $\epsilon$.

We further study the security of our model based on our differential privacy (DP) and attack model described in Section 3.2. In particular, we define the attack risk as $\|ls_{attack} \cap ls_{truth}\|/ls_{truth}$, where ls stands for the location set, and $ls_{attack}$ is the estimated location set based on differences between the downloaded model and the uploaded model. Table 1 gives the attack risk of FedADMP on different DP parameter $\epsilon$. Furthermore, Figure 13 and Table 1 together show that the more noises are added, the better privacy protection is provided at the cost of model accuracy. Thus, there exists a tradeoff between the privacy and the accuracy

| DP $\epsilon$ | 10 | 20 | 40 | 60 | 80 | 100 | $\infty$ |
|---|---|---|---|---|---|---|---|
| Beijing | 0.08 | 0.313 | 0.401 | 0.439 | 0.454 | 0.462 | 0.503 |
| KSR | 0.101 | 0.392 | 0.508 | 0.556 | 0.576 | 0.585 | 0.637 |
| Seattle | 0.106 | 0.413 | 0.535 | 0.586 | 0.606 | 0.616 | 0.670 |
| NYC | 0.144 | 0.559 | 0.723 | 0.792 | 0.0819 | 0.8292 | 0.906 |
| SF | 0.121 | 0.534 | 0.627 | 0.685 | 0.706 | 0.724 | 0.771 |

**Table 1.** The variation of attack risk with different DP parameters $\epsilon$.

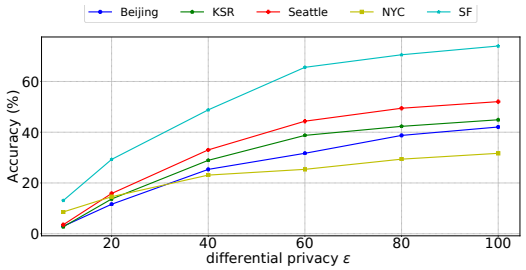


**Figure 13.** The impact of different differential privacy parameters.

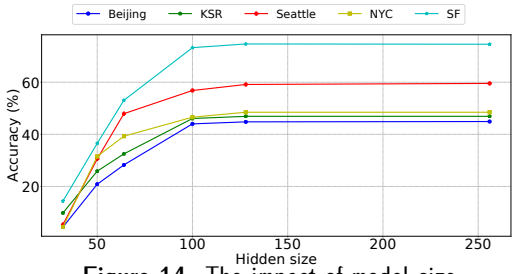


**Figure 14.** The impact of model size.

for specific conditions, which provides us a control knob based on different application requirements.

**The impact of model size:** Since the client devices in ubiquitous computing systems are usually limited in storage, energy, and computational resources, we hope to design a model that has small number of parameters. However, models with fewer parameters usually have worse performance. As a result, it is important to achieve the desirable tradeoff between the number of parameters and the performance in ubiquitous computing systems. We conduct experiments to evaluate the impact of the model size on the accuracy of the model. In particular, we focus on the impact of hidden layers. For simplicity, we set the dimension $h_r$ of the hidden state in LSTM the same as the dimension $h_l$ of the input and output state, which denotes the size of hidden state (i.e., hidden size). Figure 14 shows that the model accuracy varies with the hidden size in all datasets. In particular, the smaller the hidden size, the worse the performance of model. Therefore, we choose 128 as the default value of the hidden size for all datasets.

**Joint model vs. individual model:** A natural question is what is the advantage of FedADMP, a joint model for anomaly detection and mobility prediction, over two separate models (one for each task)? Below, we address this question by comparing FedADMP against executing two tasks separately in terms of communication costs, time consumption ,and energy cost. As shown in Figure 15, FedADMP achieves a better performance in all performance metrics. There are two major reasons. First, FedADMP reduces the training time, which results in less energy consumption on client devices. At the same time, the communication cost is reduced since less epoch is taken to achieve the same performance accuracy. Secondly, FedADMP allows to process two tasks simultaneously while having separate models for each task requires to process two models one after the other, which incurs additional overheads.

**Robustness of FedADMP:** Different from traditional machine learning models that run on servers in well-managed data centers, client devices in ubiquitous computing systems often have various amount of computing powers. This makes it challenging for the coordinator to efficiently identify and manage valuable participants. Furthermore, client devices often vary in system performance and may slow down or drop out of the network. To that end, we evaluate the robustness of our FedADMP on scenario where not all client devices are available for training and testing, and characterize the performance tradeoff in terms of model accuracy, energy consumption and communication costs.

We consider a scenario with 100 client devices, and randomly select some client devices for training and testing under the assumption that the remaining client devices are not available. For the ease of exposition, we only present results for three cases: (i) FedADMP-40: 40 client devices are used for training and testing; (ii) FedADMP-70: 70 client devices participate in training and testing; and (iii) FedADMP-100: all client devices participate in training and testing. Figure 16 presents the results of FedADMP in terms of convergence. The figure shows that when the number of devices involved in the training is small (e.g., FedADMP-40), it takes more epochs but less time to converge and the accuracy rate is lower. This is because, when fewer client devices are involved in training, less data is used for training. Hence the length of each epoch is smaller and it takes more epoch to converge at the cost of a lower accuracy. More importantly, we observe that when a relatively large number of client devices participate in the training (e.g., FedADMP-70), FedADMP achieves almost the same accuracy

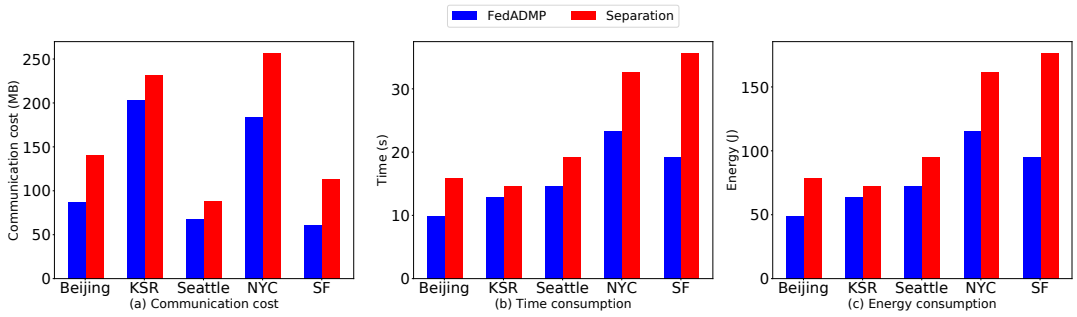


**Figure 15.** Performance comparison between FedADMP and separate models.

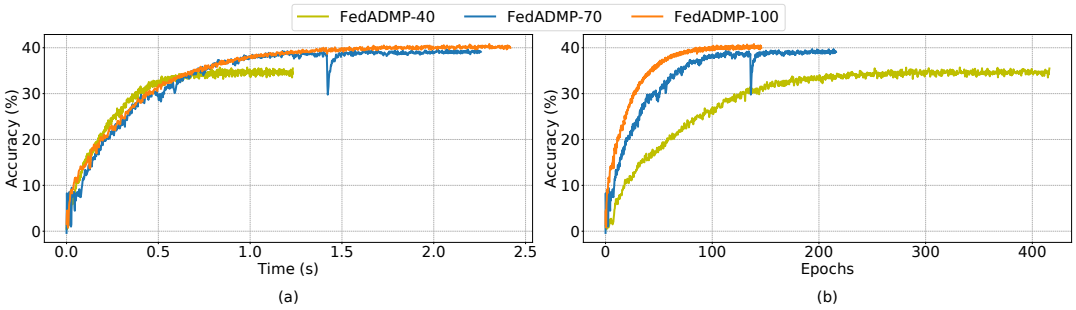


**Figure 16.** Convergence and accuracy performance of FedADMP when partial client devices are available in Beijing trajectory.

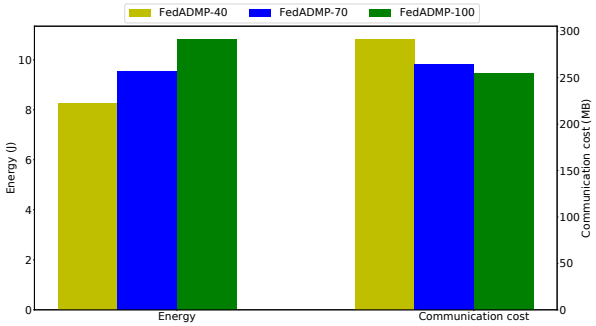


**Figure 17.** Energy consumption and communication cost of FedADMP when partial client devices are available in Beijing trajectory.

with relatively small convergence time compared to FedADMP-100. One insight is that some client devices may not provide enough meaningful information for the whole system model and the information provided by client devices may overlap, and hence when enough client devices are involved, we obtain desirable results. Our observations coincide with Figure 17, where FedADMP-40 consumes the least energy (as it takes the least time to converge), but has the highest communication costs (as it takes more epochs to converge). Similar trends are observed in other trajectories, and hence are relegated to the Appendix A.

## 5. Related work

**Anomaly detection and mobility prediction** is an emerging topic in ubiquitous computing systems, which has recently been considered by a large number of researchers. Markov models are widely used in this area to cluster the location information from the trajectories [9] to cluster the location information from the trajectories. Performance guaranteed algorithms have been proposed to capture the tradeoff between detection latency and accuracy [45–47]. To deal with the issue of data sparsity, matrix factorization has been introduced and applied (e.g., [48]). With the tremendous success of deep neural networks in various domains, deep learning based models have been proposed to solve this problem (e.g., [7, 8, 10, 24, 49]). While these state-of-the-art methods achieve reasonable performance in domain-specific applications, none of them protect the privacy of the client data.

Furthermore, existing works on target recognition focused on synthesize photos from sketches. For example, Tang et al. [50] proposed techniques to automatically match hand-drawn sketches of human faces with photos. Kumar et al. [51] were inspired by the characteristics displayed by attribute-based representations in other pattern recognition problems, and the desire to perform a semantic search on face images, and designed the solution to achieve significant recognition accuracy. Klare et al. [52] developed an algorithm to perform automated extraction. The proposed algorithm operated by performing facial component positioning

and alignment, followed by texture descriptor encoding and support vector regression. For target tracking, Brooks et al. [53] described a prediction-based sensor collaboration that uses estimation of target velocities to activate regions of sensors. Estrin et al. [54] developed the directed diffusion approach to move sensor data in a network that seeks to minimize communication distance between data sources and data sinks.

**Privacy-preserving methods** such as differential privacy [21] and k-anonymity [55] were proposed to address the increasing demand of privacy regulations. However, these methods protect data at the cost of destroying the data structure, which impacts the performance of ADMP. Federated learning (FL) was recently proposed to protect privacy of mobile devices. Many optimizations (e.g., [56, 57]) have been developed to reduce the communication and computation costs of FL. However, existing FL-based models perform training heavily on client devices, which becomes a critical bottleneck for resource-constraint client devices in ubiquitous computing environments. Some other works (e.g., [58]) augmented FL with privacy through differential privacy techniques by adding artificial noise to the whole model in the aggregation stage, which imposes overhead on the model. Another line of works focuses on the split learning (SL) [29, 30, 59–62], a collaborative deep learning technique that splits a deep learning network into two parts: a client-side network and a server-side network. The training of the network is done in a sequential manner where the server trains with one client and then moves to another client. However, such a sequential training engages only one client at a time and hence is not efficient. Vertical federated learning (VFL) enables multiple parties that own different attributes (e.g., features and labels) of the same data entity (e.g., a person) to jointly train a model [63–65]. In contrast to VFL where clients transfer the whole model to the server, our FedADMP transfers only the activation from clients to the server.

The work that is most close to ours is PMF [23], which is a FL-based method for mobility prediction. However, PMF has a low efficiency for ADMP with relatively high computation and communication costs on client devices. In departure from the above works, FedADMP performs anomaly detection and mobility prediction simultaneously with privacy guarantees.

## 6. Conclusion

In this paper, we considered the joint anomaly detection and mobility prediction (ADMP) problem in ubiquitous computing systems. We proposed FedADMP, a federated learning-based framework, for ADMP. To reduce the computation loads on resource-constrained client devices, client devices and the cloud server work collaboratively on the training process. To protect the privacy of user data, each client device uploads only the activation, instead of the raw data or the whole local model, to the cloud server. We also developed a differential privacy method to further protect the privacy and strengthened the robustness of FedADMP when only partial clients devices are available for training, which is a situation often occurring for emerging applications in ubiquitous computing systems. Our experimental results show that FedADMP consistently outperforms state-of-the-art methods in terms of model accuracy with dramatically reduced energy consumption and computation costs at client devices.

## 7. Acknowledgements

We would like to thank the anonymous reviewers for their constructive feedback. We also thank Jerome Dinal Herath for his help in the initial phases of the project. This work was supported in part by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Solar Energy Technologies Office Award Number DE-EE0009341 and the National Science Foundation under grant OAC-1738929. This work was also supported in part by the Research Foundation of State University of New York through the Transdiscipinary Areas of Excellence Seed Grant Program Award Number 61476.

## Appendix A. Additional Experimental Results

Complementary to the results presented in Section 4, we provide additional experimental results when using the Fashion-MNIST dataset as the object recognition database. The time-to-accuracy and epoch-to-accuracy performance are presented in Figures .1 and .2, respectively. Again, we observe that FedADMP consistently outperforms other four systems although the accuracy improvement over PMF is limited in some traces. More importantly, FedADMP dramatically reduces the convergence time to achieve a particular accuracy. This property is highly desirable for the joint anomaly detection and mobility prediction task in ubiquitous computing systems since a timely decision is critical in many emerging applications as motivated in the introduction.

Similarly, we compare the energy consumed by client devices in different models for achieving the same accuracy. We assume an ideal cloud server which always has enough computation resources. Figure A.3 *(Left)* compares the energy consumed by client devices in different models when achieving 40% accuracy. Again, we observe that FedADMP consumes the least energy without incurring additional communication costs which are measured by the size of the data transmitted between the server and the clients, as illustrated in Figure A.3 *(Right)*.

Finally, we provide additional results to evaluate the robustness of FedADMP using NYC and SF trajectories.

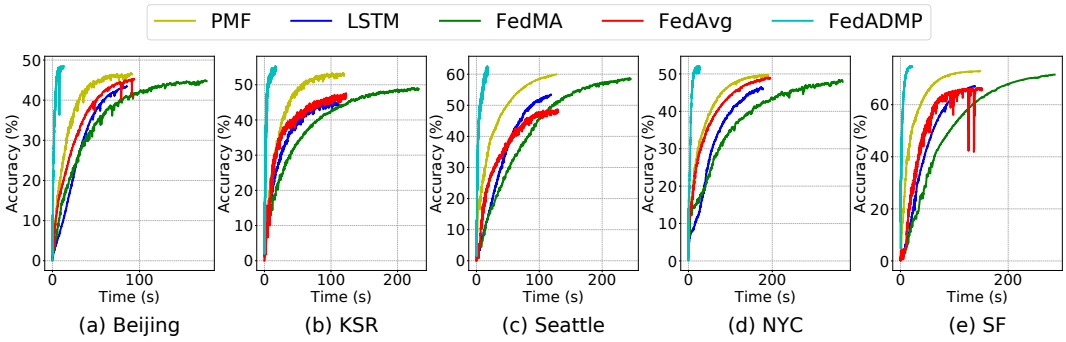


**Figure .1.** Convergence and accuracy performance measure in time (i.e., time-to-accuracy) when using Fashion-MNIST dataset as the object recognition database.

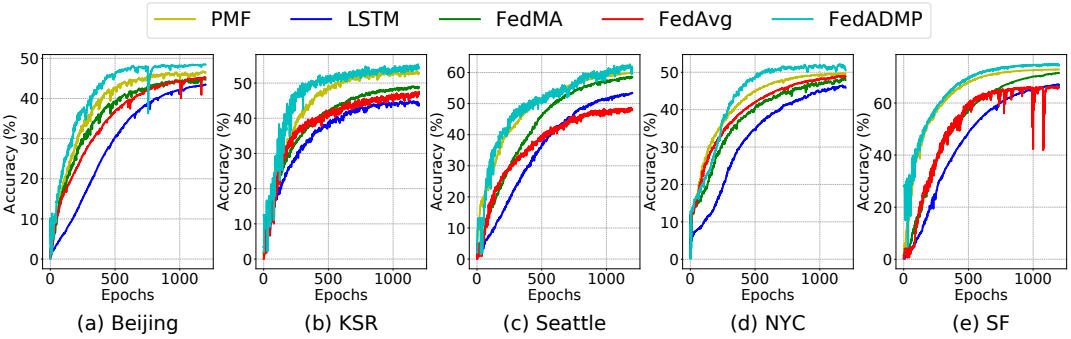


**Figure .2.** Convergence and accuracy performance measure in epochs (i.e., epoch-to-accuracy) when using Fashion-MNIST dataset as the object recognition database.

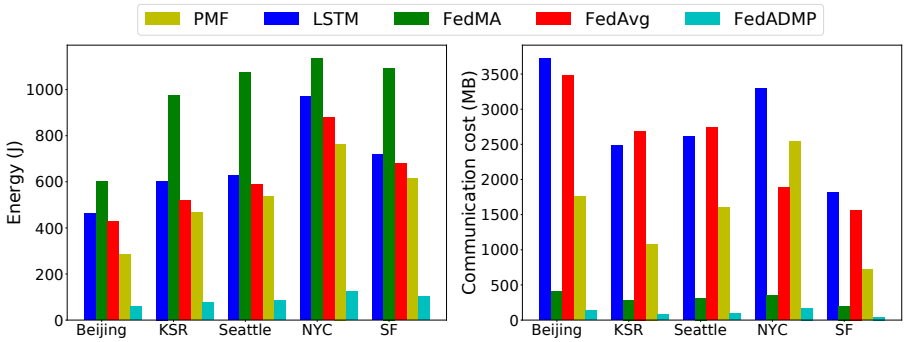


**Figure A.3.** *(Left):* Energy consumption when using Fashion-MNIST dataset as the object recognition database. *(Right):* Communication cost when using Fashion-MNIST dataset as the object recognition database.

As shown in Figures A.4 - A.7, we make the same observations as in Figures 16 and 17.

## References

[1] S. V. Nath, "Crime Pattern Detection Using Data Mining," in *2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology Workshops*. IEEE, 2006, pp. 41–44.

[2] I. Golan and R. El-Yaniv, "Deep Anomaly Detection Using Geometric Transformations," in *Proc. of NeurIPS*, 2018.

[3] A. Kumagai, T. Iwata, and Y. Fujiwara, "Transfer Anomaly Detection by Inferring Latent Domain Representations," *Proc. of NeurIPS*, 2019.

[4] M. Zhang, T. Li, H. Shi, Y. Li, and P. Hui, "A Decomposition Approach for Urban Anomaly Detection Across Spatiotemporal Data," in *IJCAI*, 2019.

[5] C. Huang, C. Zhang, P. Dai, and L. Bo, "Cross-Interaction Hierarchical Attention Networks for Urban Anomaly Prediction," in *IJCAI*, 2020.

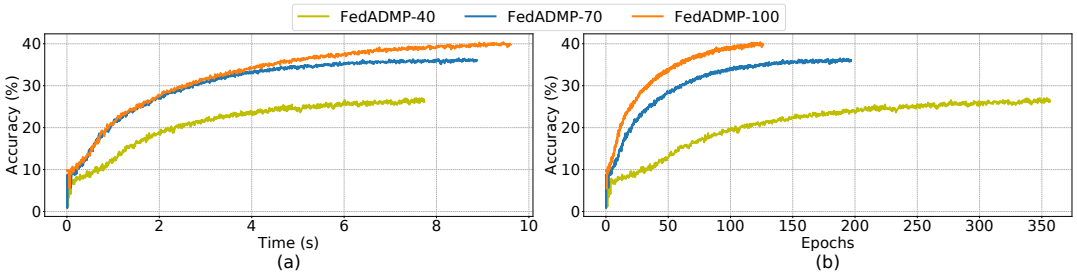


**Figure A.4.** Convergence and accuracy performance of FedADMP when partial client devices are available in NYC trajectory.

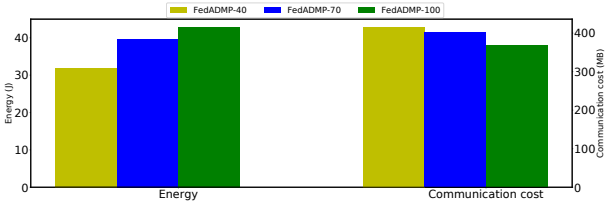


**Figure A.5.** Energy consumption and communication cost of FedADMP when partial client devices are available in NYC trajectory.


[6] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller, and M. Kloft, "Deep Semi-Supervised Anomaly Detection," *arXiv preprint arXiv:1906.02694*, 2020.

[7] D. Kong and F. Wu, "HST-LSTM: A Hierarchical Spatial-Temporal Long-Short Term Memory Network for Location Prediction," in *IJCAI*, 2018.

[8] D. Liao, W. Liu, Y. Zhong, J. Li, and G. Wang, "Predicting Activity and Location with Multi-task Context Aware Recurrent Neural Network," in *IJCAI*, 2018.

[9] W. Mathew, R. Raposo, and B. Martins, "Predicting Future Locations with Hidden Markov Models," in *Proc. of ACM UbiComp*, 2012.

[10] P. Zhao, H. Zhu, Y. Liu, J. Xu, Z. Li, F. Zhuang, V. S. Sheng, and X. Zhou, "Where to Go Next: A Spatio-Temporal Gated Network for Next POI Recommendation," in *Proc. of AAAI*, 2019.

[11] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proc. of AISTATS*, 2017.

[12] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and Open Problems in Federated Learning," *arXiv preprint arXiv:1912.04977*, 2019.

[13] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A Survey on Federated Learning for Resource-Constrained IoT Devices," *IEEE Internet of Things Journal*, 2021.

[14] G. Yan, H. Wang, and J. Li, "Critical Learning Periods in Federated Learning," *arXiv preprint arXiv:2109.05613*, 2021.

[15] F. Hartmann, S. Suh, A. Komarzewski, T. D. Smith, and I. Segall, "Federated Learning for Ranking Browser History Suggestions," *arXiv preprint arXiv:1911.11807*, 2019.

[16] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied Federated Learning: Improving Google Keyboard Query Suggestions," *arXiv preprint arXiv:1812.02903*, 2018.

[17] W. Li, F. Milletarì, D. Xu, N. Rieke, J. Hancox, W. Zhu, M. Baust, Y. Cheng, S. Ourselin, M. J. Cardoso *et al.*, "Privacy-Preserving Federated Brain Tumour Segmentation," in *International Workshop on Machine Learning in Medical Imaging*. Springer, 2019, pp. 133–141.

[18] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein, "Learning in situ: A Randomized Experiment in Video Streaming," in *Proc. of USENIX NSDI*, 2020.

[19] K. Hsieh, G. Ananthanarayanan, P. Bodik, S. Venkataraman, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu, "Focus: Querying Large Video Datasets with Low Latency and Low Cost," in *Proc. of USENIX OSDI*, 2018.

[20] J. Luo, X. Wu, Y. Luo, A. Huang, Y. Huang, Y. Liu, and Q. Yang, "Real-World Image Datasets for Federated Learning," *arXiv preprint arXiv:1910.11089*, 2019.

[21] C. Dwork, "Differential Privacy: A Survey of Results," in *Proc. of TAMC*, 2008.

[22] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan *et al.*, "Towards Federated Learning at Scale: System Design," *arXiv preprint arXiv:1902.01046*, 2019.

[23] J. Feng, C. Rong, F. Sun, D. Guo, and Y. Li, "PMF: A Privacy-Preserving Human Mobility Prediction Framework via Federated Learning," *Proc. of ACM UbiComp*, 2020.

[24] J. Feng, Y. Li, C. Zhang, F. Sun, F. Meng, A. Guo, and D. Jin, "Deepmove: Predicting Human Mobility with Attentional Recurrent Networks," in *Proc. of WWW*, 2018.

[25] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, "Understanding Mobility Based on GPS Data," in *Proc. of ACM UbiComp*, 2008.

[26] D. Yang, D. Zhang, Z. Yu, and Z. Yu, "Fine-Grained Preference-Aware Location Search Leveraging Crowdsourced Digital Footprints from LBSNs," in *Proc. of ACM UbiComp*, 2013.

[27] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "A Parsimonious Model of Mobile Partitioned

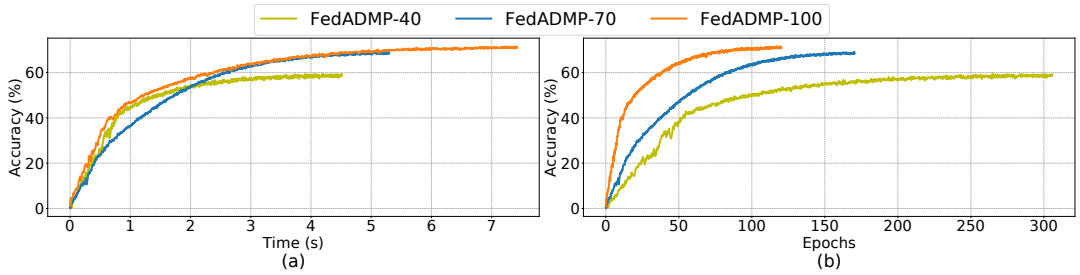


**Figure A.6.** Convergence and accuracy performance of FedADMP when partial client devices are available in SF trajectory.

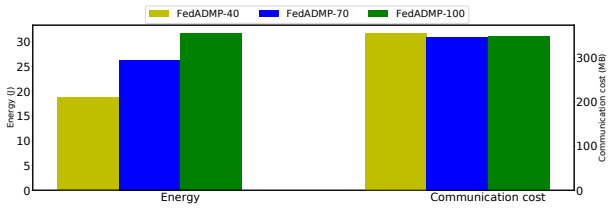


**Figure A.7.** Energy consumption and communication cost of FedADMP when partial client devices are available in SF trajectory.

Networks with Clustering," in *Proc. of IEEE COMSNETS*, 2009.

[28] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[29] A. Singh, P. Vepakomma, O. Gupta, and R. Raskar, "Detailed Comparison of Communication Efficiency of Split Learning and Federated Learning," *arXiv preprint arXiv:1909.09145*, 2019.

[30] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split Learning for Health: Distributed Deep Learning Without Sharing Raw Patient Data," *arXiv preprint arXiv:1812.00564*, 2018.

[31] J. Wang, N. Wu, W. X. Zhao, F. Peng, and X. Lin, "Empowering A* Search Algorithms with Neural Networks for Personalized Route Recommendation," in *Proc. of ACM SIGKDD*, 2019.

[32] H. Wu, Z. Chen, W. Sun, B. Zheng, and W. Wang, "Modeling Trajectories with Recurrent Neural Networks." IJCAI, 2017.

[33] R. Shokri and V. Shmatikov, "Privacy-Preserving Deep Learning," in *Proc. of ACM CCS*, 2015.

[34] C. Ma, J. Li, M. Ding, H. H. Yang, F. Shu, T. Q. Quek, and H. V. Poor, "On Safeguarding Privacy and Security in the Framework of Federated Learning," *IEEE Network*, 2020.

[35] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential Privacy for Location-Based Systems," in *Proc. of ACM CCS*, 2013.

[36] D. P. Kingma and J. Ba, "ADAM: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[37] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, "Data Poisoning Attacks Against Federated Learning Systems," in *European Symposium on Research in Computer Security*. Springer, 2020, pp. 480–501.

[38] A. Huang, "Dynamic Backdoor Attacks Against Federated Learning," *arXiv preprint arXiv:2011.07429*, 2020.

[39] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 603–618.

[40] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning," in *Proc. of IEEE INFOCOM*, 2019.

[41] M. Song, Z. Wang, Z. Zhang, Y. Song, Q. Wang, J. Ren, and H. Qi, "Analyzing user-level privacy attack against federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2430–2444, 2020.

[42] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[43] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[44] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated Learning with Matched Averaging," *arXiv preprint arXiv:2002.06440*, 2020.

[45] S. Zou, V. V. Veeravalli, J. Li, and D. Towsley, "Quickest Detection of Dynamic Events in Networks," *IEEE Transactions on Information Theory*, vol. 66, no. 4, pp. 2280–2295, 2019.

[46] J. Li, D. Towsley, S. Zou, V. V. Veeravalli, and G. Ciocarlie, "A Consensus-based Approach for Distributed Quickest Detection of Significant Events in Networks," in *Proc. of Asilomar*, 2019.

[47] S. Zou, V. V. Veeravalli, J. Li, D. Towsley, and A. Swami, "Distributed Quickest Detection of Significant Events in Networks," in *Proc. of IEEE ICASSP*, 2019.

[48] C. Cheng, H. Yang, I. King, and M. Lyu, "Fused Matrix Factorization with Geographical and Social Influence in Location-Based Social Networks," in *Proc. of AAAI*, 2012.

[49] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song, "Recurrent Marked Temporal Point Processes: Embedding Event History To Vector," in *Proc. of ACM SIGKDD*, 2016.

[50] X. Wang and X. Tang, "Face Photo-Sketch Synthesis and Recognition," *IEEE Transactions on Pattern Analysis*

*and Machine Intelligence*, vol. 31, no. 11, pp. 1955–1967, 2008.

[51] N. Kumar, A. Berg, P. N. Belhumeur, and S. Nayar, "Describable Visual Attributes for Face Verification and Image Search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 10, pp. 1962–1977, 2011.

[52] B. F. Klare, S. Klum, J. C. Klontz, E. Taborsky, T. Akgul, and A. K. Jain, "Suspect Identification Based on Descriptive Facial Attributes," in *IEEE International Joint Conference on Biometrics*. IEEE, 2014, pp. 1–8.

[53] R. R. Brooks, C. Griffin, and D. Friedlander, "Self-Organized Distributed Sensor Network Entity Tracking," *The International Journal of High Performance Computing Applications*, vol. 16, no. 3, pp. 207–219, 2002.

[54] "Next Century Challenges: Scalable Coordination in Sensor Networks, author=Estrin, Deborah and Govindan, Ramesh and Heidemann, John and Kumar, Satish, booktitle=Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, pages=263–270, year=1999."

[55] M. Gramaglia and M. Fiore, "Hiding Mobile Traffic Fingerprints With Glove," in *Proc. of ACM CoNEXT*, 2015.

[56] N. Agarwal, A. T. Suresh, F. X. X. Yu, S. Kumar, and B. McMahan, "cpSGD: Communication-Efficient and Differentially-Private Distributed SGD," in *Proc. of NeurIPS*, 2018.

[57] G. Xiong, G. Yan, and J. Li, "Straggler-Resilient Distributed Machine Learning with Dynamic Backup Workers," *arXiv preprint arXiv:2102.06280*, 2021.

[58] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning Differentially Private Recurrent Language Models," in *Proc. of ICLR*, 2018.

[59] O. Gupta and R. Raskar, "Distributed Learning of Deep Neural Network over Multiple Agents," *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, 2018.

[60] S. Abuadbba, K. Kim, M. Kim, C. Thapa, S. A. Camtepe, Y. Gao, H. Kim, and S. Nepal, "Can We Use Split Learning on 1D CNN Models for Privacy Preserving Training?" in *Proc. of ACM AsiaCCS*, 2020.

[61] Y. Gao, M. Kim, S. Abuadbba, Y. Kim, C. Thapa, K. Kim, S. A. Camtepe, H. Kim, and S. Nepal, "End-to-End Evaluation of Federated Learning and Split Learning for Internet of Things," *arXiv preprint arXiv:2003.13376*, 2020.

[62] R. Khincha, U. Sarawgi, W. Zulfikar, and P. Maes, "Robustness to Missing Features using Hierarchical Clustering with Split Neural Networks," *arXiv preprint arXiv:2011.09596*, 2020.

[63] T. Chen, X. Jin, Y. Sun, and W. Yin, "Vafl: A Method of Vertical Asynchronous Federated Learning," *arXiv preprint arXiv:2007.06081*, 2020.

[64] D. Romanini, A. J. Hall, P. Papadopoulos, T. Titcombe, A. Ismail, T. Cebere, R. Sandmann, R. Roehm, and M. A. Hoeh, "Pyvertical: A Vertical Federated Learning Framework for Multi-Headed Splitnn," *arXiv preprint arXiv:2104.00489*, 2021.

[65] I. Ceballos, V. Sharma, E. Mugica, A. Singh, A. Roman, P. Vepakomma, and R. Raskar, "Splitnn-Driven Vertical Partitioning," *arXiv preprint arXiv:2008.04137*, 2020.