Efficient Distributed Training in Heterogeneous Mobile Networks with Active Sampling

Yunhui Guo Xiaofan Yu Kamalika Chaudhuri Tajana Rosing
University of California, San Diego, CA
yug185@eng.ucsd.edu, x1yu@eng.ucsd.edu, kamalika@cs.ucsd.edu, tajana@ucsd.edu

***Abstract*—Mobile edge computing is an emerging research topic which aims at pushing the computation from the cloud to the edge devices. Most of the current machine learning (ML) algorithms, such as federated learning, are designed for homogeneous mobile networks, that is, all the devices collect the same type of data. In this paper, we address distributed training of ML algorithms in heterogeneous mobile networks where the *features*, rather than the *samples*, are distributed across multiple heterogeneous mobile devices. Training ML models in heterogeneous mobile networks incurs a large communication cost due to the necessity to deliver the local data to a central server. Inspired by active learning, which is traditionally used to reduce the labeling cost for training ML models, we propose an active sampling method to reduce the communication cost of learning in heterogeneous mobile networks. Instead of sending all the local data, the proposed active sampling method identifies and sends only informative data from each device to the central server. Extensive experiments on four real datasets, both with numerical simulation and on a networked mobile system, show that the proposed method can reduce the communication cost by up to 53% and energy consumption by up to 67% without accuracy degradation compared with the conventional approaches.**

I. INTRODUCTION

Devices such as mobile phones, tablets and mobile sensors are generating a huge amount of data each day, enabling everything from remotely monitoring heart rate to tracking the location of smartphone [1], [2]. Machine learning (ML) algorithms have become a core component for building data analytic systems [3], [4]. Most ML algorithms are server-based and designed for handling centralized data, that is, all the training examples are generated in one place [5]. However, mobile networks are distributed in nature. Each device only gathers a subset of the data and works collaboratively with a central server. Thus, learning in mobile networks requires extensive data communication which is challenging for training conventional ML algorithms.

Combining information from multiple heterogeneous mobile devices is a new research direction for mobile edge computing [6]–[8]. One characteristic of learning with heterogeneous mobile devices is that the *feature vectors*, rather than the *examples*, are distributed across devices [6], [7]. Most of the works on mobile edge computing, such as *federated learning* [5] and *distributed gradient algorithms* [9], [10], mainly focus on learning in homogeneous mobile networks where the devices have different subsets of the dataset that share common feature space. In contrast, learning in heterogeneous mobile network has the following key properties which make it a more challenging task and has only been addressed by some recent works [7], [11].

- **Distributed features**: Conventional ML algorithms need access to the full feature vectors to make predictions. However, in heterogeneous mobile network the feature vectors are distributed across devices. To gather together the distributed features in the central server incurs a large communication cost. Excessive data communication also dominates the energy consumption of the mobile devices [12], [13] and leads to network congestion.
- **Asymmetrical network bandwidth**: Many telecom companies provide Internet plans with much faster download speeds than upload speeds [14]. For example, AT&T top download and upload speeds can have as much as 5x difference. On the other hand, the devices typically have high sampling rates (100 Hz or higher) and deliver continuous data to the central server. The limited upload speed poses a challenge to communicate the local data which increases the network latency and power usage.
- **Online update**: The user behavior might change over time. It is thus important to have an online update mechanism to adjust the trained ML model in the central server with incoming data with minimal communication cost.

While learning in heterogeneous mobile networks is generally difficult, in this paper we leverage two key facts and propose an active sampling algorithm for training in heterogeneous mobile networks to reduce the communication cost and energy consumption. First, we observe that not all the data are equally important. For example, the data that identify the transition from one activity to another are more informative. On each round of the online update, instead of sending all the data to the central server, the proposed active sampling method sends only the most *informative* data. Our key insight is that these informative data can be chosen by using ideas from the active learning literature [15].

To summarize, our work makes three main contributions,

- To our knowledge this is the first attempt to systematically show that the idea of active learning can be used to greatly reduce the communication cost and energy consumption for training in heterogeneous mobile networks.
- We propose active sampling methods for communication and energy efficient training in heterogeneous mobile networks.
- We validate the proposed approaches on four real-world datasets by both numerical simulation and practical de-

ployment. The results show that we can achieve a reduction in communication cost (in bits) by up to 53% and in energy consumption by up to 67% without accuracy loss compared with the conventional methods.

II. Related Work

A fundamental problem in learning in distributed mobile networks is how to make the right tradeoff between communication and computation [2], [5], [16]. Based on the characteristics of the devices, previous work can be classified into two categories: learning with a homogeneous set of devices [2], [5], [10] and learning with a heterogeneous set of devices [6], [17], [18]. Conventional distributed gradient descent algorithms [9], [10], [19] and federated learning [5] fall under the first category where the examples are distributed across devices. The main communication cost in this setting comes from sending the model parameters between devices and the central server [20]. The methods for reducing communication cost in homogeneous mobile networks can be classified into two threads. One is model compression and quantization, which tries to reduce the communication cost by compressing or quantizing the model parameters [20]–[22]. Another is from an algorithmic perspective, which focuses on developing communication efficient optimization algorithms [23].

In contrast, in heterogeneous mobile networks the feature vectors, rather than the examples, are distributed across different devices [6], [7], [18]. The distributed features further increase the communication cost between devices and the central server [17], [18]. In [6], the authors assume the devices can communicate with each other which is unrealistic in real-world mobile applications. In [7], [11], the authors proposed to distribute the model in the central server onto the devices and conduct computation locally to reduce the dimension of the data. However, these works mainly focus on inference rather than training in heterogeneous mobile networks.

III. Background

In this section, we formulate the problem of learning in heterogeneous mobile networks. Given a heterogeneous mobile network which consists of one central server C and K mobile devices. The central server and the mobile devices are connected via wireless network. We consider an online learning setting where the examples arrive sequentially $\{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_T, \mathbf{y}_T)\}$, where $\mathbf{x}_t \in \mathbb{R}^m$ is an m-dimensional feature vector and $\mathbf{y}_t$ is the one-hot representation of the ground-truth label. The maximum class index is denoted as I. Consider learning a neural network $f_\theta(\mathbf{x})$ with parameter θ for classification in the central server. The prediction of the model can be computed as $\hat{\mathbf{y}} = \text{softmax}(\mathbf{z}) = \frac{\exp(\mathbf{z})}{\sum_{i=1}^{I}\exp(\mathbf{z}_i)}$, where $\mathbf{z} = f_\theta(\mathbf{x})$. Denote the cross-entropy loss function as ℓ, the loss of the example $\mathbf{x}$ can be computed as $\ell(\hat{\mathbf{y}}; \mathbf{y}) = -\frac{1}{I}\sum_{i=1}^{I} y_i \log \hat{y}_i$, where $\hat{y}_i$ and y_i is the i-th component of $\hat{\mathbf{y}}$ and $\mathbf{y}$, respectively. In heterogeneous mobile network, each device k contains a partial feature vector $\mathbf{x}^k \in \mathbb{R}^{m_1}$. The data from all K devices can be concatenated to form

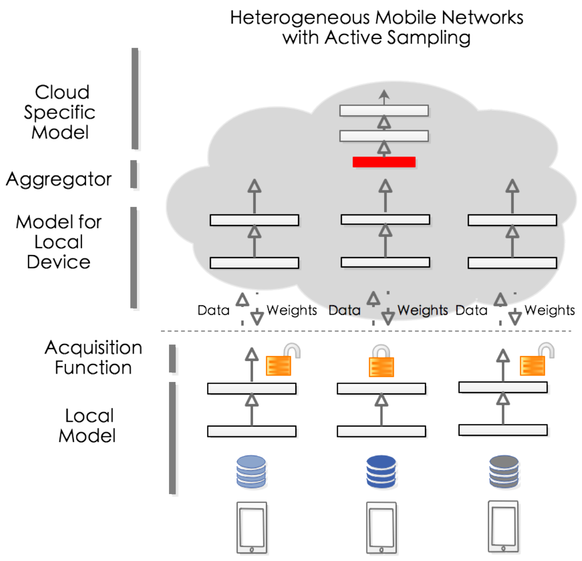


Fig. 1: **The overview of the proposed active sampling approach for training in heterogeneous mobile networks**.

a single feature vector $\mathbf{x}$, i.e., $m = \sum_{k=1}^{K} m_1$. Typically, $\mathbf{x}^k$ itself does not contain enough information to learn a predictive model locally, which indicates that all the mobile devices need to send the local data to the central server [7], [11], [24]. The communication process thus incurs prohibitively large communication and energy costs [7], [11].

IV. Active Sampling in Heterogeneous Mobile Networks

A. Active Learning

Label acquisition for unlabeled data is expensive since it requires the participation of domain experts. One important topic in machine learning research is how to train an accurate predictive model based on as few labeled examples as possible. Active learning [15], [25], [26], which reduces the labeling cost by querying the labels of the most informative examples, has attracted a lot of attention recently. Assume we have a dataset $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$. The set of the examples is denoted as $D_\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$. In active learning, we aim to learn a probabilistic model $p(y|\mathbf{x}; \theta)$ using as few labeled examples as possible. To achieve this, we start with a subset $L \in D$ of the dataset and a large pool of unlabeled example $U_\mathbf{x} = D_\mathbf{x} \setminus L_\mathbf{x}$. We first train a model M_0 based on L. In each iteration of active learning, we query the label for an example $\mathbf{x} \in U_\mathbf{x}$ which can most improve the generalization ability of the current model. Once we have identified the sample to query, we add it to L and its label is given by an oracle at a cost. The oracle is typically a human annotator. We retrain the model again on L with the additional example $\mathbf{x}$. The above process is repeated until a predefined accuracy is met.

One way to evaluate the informativeness of the example is called *uncertainty sampling* [15]. In uncertainty sampling, an active learner queries the label of the example about which it is most uncertain how to label based on an acquisition function $U(\mathbf{x};\theta)$. Commonly used acquisition functions include least confident and confidence margin [15],

- Least confident: $\mathbf{x}^*_{LC} = 1 - \arg\min_{\mathbf{x}\in U_{\mathbf{x}}} p(y^*|\mathbf{x};\theta)$, where $y^* = \arg\max_{\mathbf{x}\in U_{\mathbf{x}}} p(y^*|\mathbf{x};\theta)$ is the most likely labeling.
- Confidence margin: $\mathbf{x}^*_{CM} = 1 - \arg\min_{\mathbf{x}\in U_{\mathbf{x}}} [p(y^*|\mathbf{x};\theta) - p(y^{**}|\mathbf{x};\theta)]$, where y^{**} is the second most likely labeling.

B. Proposed approach

In the context of online learning, it is necessary to update the initial model in the central server to address the dynamics of the environment. However, naively sending all the local data to the central server incurs a large communication cost. In this paper, we propose a communication and energy efficient active sampling algorithm for training in heterogeneous mobile networks. The overview of the proposed architecture is shown in Figure 1. In the central server for each device k of all the K devices, we construct a model f_{θ_k}. The outputs of all the models are concatenated via an aggregate function and are used as input for a cloud specific model f_{θ_c}. The weights of all models in cloud are denoted as θ. On each device k, we construct a local model $\hat{f}_{\theta_k}$ with the same architecture as f_{θ_k}. During the online update process, the local devices send the collected data to the cloud. The cloud updates the models based on received data and sends the updated weights to each local device. On each round, the proposed algorithm identifies informative samples based on $\hat{f}_{\theta_k}$. This means that only a subset of the devices need to communicate with the central server. For example, in Figure 1, only the first device and the third device need to send data to the cloud. On the other hand, in order to be updated, the central server models needs data from all the devices to evaluate the loss function. For those devices which do not send data, we *synthesize* their data in the central server based on the available data from other devices. After the models are updated in the central server, the weights of model f_{θ_k} is sent to device k for synchronization. Finally, in order to prevent one device from dominating the data communication, we propose a lightweight load balancing mechanism to promote each device to communicate roughly the same which can increase the lifetime of the overall system.

1) Design of Acquisition function: We assume that there is an initial model M_0 in the central server, we aim to update M_0 based on the data from all the devices. At each timestamp t, device k collects data $\mathbf{x}^k_t$. The data from all the devices are denoted as $\mathbf{x}_t$. The collection of data without $\mathbf{x}^k_t$ is denoted as $\mathbf{x}_t \setminus \mathbf{x}^k_t$. Our goal is to update the central server model with as few as communication rounds as possible. To achieve this, we propose active sampling to measure the informativeness of $\mathbf{x}^k_t$ based on the local model on each device. Instead of sending all the local measurements to the central server, we only consider the informative ones. We propose two acquisition functions, local uncertainty and delayed global uncertainty, for applying active sampling in heterogeneous mobile network.

Local Uncertainty. In local uncertainty (LU), we assume the devices are independent with each other when evaluating the uncertainty locally, that is, the acquisition function $U(\mathbf{x};\theta)$ only depends on $\mathbf{x}_k$ and θ_k. Although we assume the devices are independent, it is worth noting that since the cloud specific model and the models for local devices are trained in an end to end manner via backpropagation, the correlations between different models can be learned in the cloud [27].

On each round t, the central server sends the weight θ_k to the corresponding device k. Device k updates the local model and evaluates the informativeness of $\mathbf{x}^k_t$ via an uncertainty function $U(\mathbf{x}^k_t;\theta_k)$ which can be the entropy function, least confident function or confidence margin function [15]. In the experiments, we evaluate the least confident function and the confidence margin function due to their effectiveness [15]. Device k sends $\mathbf{x}^k_t$ to the central server only if the local uncertainty $U(\mathbf{x}^k_t;\theta_k)$ is above a given threshold γ. The independence assumption leads to a great reduction in communication cost – the overhead is that on each round the central server needs to send θ_k to each device k which is tolerable since the dimension of θ_k is usually small.

Delayed Global Uncertainty. In delayed global uncertainty (DGU), at each timestamp t, the central server sends the weights of all the models θ and $\mathbf{x}_{t-1} \setminus \mathbf{x}_{t-1,k}$ to device k. Together with $\mathbf{x}^k_t$, we form a new feature vector, denoted by $[\mathbf{x}^k_t, \mathbf{x}_{t-1} \setminus \mathbf{x}^k_{t-1}]$, which is $\mathbf{x}_{t-1}$ with $\mathbf{x}^k_{t-1}$ replaced by $\mathbf{x}^k_t$. In DGU, device k evaluates the informativeness of $\mathbf{x}_{t,k}$ via the function $U([\mathbf{x}^k_t, \mathbf{x}_{t-1} \setminus \mathbf{x}^k_{t-1}];\theta)$. Compared with the true global uncertainty $U(\mathbf{x}_t;\theta)$, it is called delayed global uncertainty since all the measurements have a lag of one timestamp except $\mathbf{x}^k_t$. The delayed global uncertainty can be seem as a better approximation to the true global uncertainty, however it increases the downlink communication cost.

The proposed active sampling approaches trade downlink communication cost for uplink communication cost. Although on each round the central server needs to communicate with the devices, the devices send data to the central server only if the data is informative. Examples of informative data include readings that indicate the device is misbehaving or the user activity is changing – both cases are only a small portion of the overall data stream. Due to the fact that the download speed is typically 5x faster than the upload speed, we can leverage the extra download bandwidth to reduce the upload congestion and increase the battery life of the devices.

2) Synthesized measurements: On each round t, if the central server did not receive the data from device k, we use a machine learning model $S(\mathbf{x}^k_t|\mathbf{x}^j_t, y_t;\beta)$ with parameters β, typically a neural network, to predict the missing data of device k with the measurement received from device j. In the experiments, we use a two-layer neural network for computational efficiency. Note that $\mathbf{x}^k_t$ is not only modeled as a function of $\mathbf{x}^j_t$, but is also conditioned on the label y_t of the data. The label information provides extra supervision for training the synthesized model S to better recover the missing data. The choice of device j depends on the actual deployment and characteristics of the datasets. For example, we can choose the device which has the highest sampling rate, since it collects most fine-grained measurements. With the synthesized measurements, we can update the central server model by relying on a small subset of informative local data.

3) Load balance: Excessive data communication is the main cause of the energy consumption in mobile and sensor networks [2]. It is thus critical to prevent one device from dominating the data communication which can shorten the lifetime of the overall system. To achieve this, we propose a lightweight load balancing mechanism to promote each device to communicate the same number of bits during the online update. The proposed load balancing mechanism adjusts the uncertainty of the data collected from a particular device based on the number of times this device has communicated before.

Suppose there are K devices in the mobile network. In the central server we maintain a vector $M \in \mathbb{R}^K$ which is initialized to be zeros. On each round t as the central server receives data from device k, $M[k]$ increases its stored value by 1. We normalize L via a softmax function to obtain M_{norm}. $M_{norm}[k]$ can be regarded as the proportion of data sent from device k. Then the central server sends $L_{norm}[k]$ to device k. With local uncertainty the informativeness of $\mathbf{x}_t^k$ is calculated as $(\frac{1/K}{M_{norm}[k]})^{\lambda_k} U(\mathbf{x}_t^k; \theta_k)$, where λ_k is a hyperparameter which can be used to adjust the strength of the load balancing. Intuitively, if $M_{norm}[k]$ is greater than $1/K$, which means that device k communicates more bits than the average, we scale down the informativeness of $\mathbf{x}_t^k$. With the proposed load balancing mechanism, the communication cost is distributed evenly across all the devices.

V. Analysis and Experiments

A. Datasets

- **MNIST** [28]: In mobile networks, we may have multiple devices which record different angles or parts of an object. To identify the object in the image, we need to combine recordings from all the devices to form a single feature vector. We use the MNIST dataset, which is the canonical dataset used for digit classification, as a proof of concept. The MNIST dataset consists of hand-written digits from 0 to 9. Each image has 784 dimensions. We assume there are 7 devices, and each records 122 dimensions of the original image. The task is to identify the digit in the image in the test dataset by utilizing the recordings from all the 7 devices.
- **PAMAP2** [29]: This is a physical activity recognition dataset which contains 18 different physical activities performed by 9 subjects. There are a total of 4 sensors, 3 inertial measurement units and a heart rate monitor. We consider the 5 main activities {lying, sitting, standing, walking and running} and use the data from subject 1 in the experiments.
- **HAR** [30]: This is a recently introduced human activity recognition dataset. It contains data collecting from 17 mobile sensors. There are a total of 30 subject performing 6 activities {walking, walking_upstairs, walking_downstairs, sitting, standing, laying}. For consistency, we use the first 7 sensors in the experiments. The data partition is the same as in [30].
- **Google Glass** (GLEAM) [31]: The GLEAM dataset consists of two hours of high resolution sensor data collected using Google Glass. There are 6 sensors and 38 subjects. The subjects conduct 6 activities {eating, talking, drinking, walking, going climbing stairs, computer_work} in a controlled environment. We use the data from the first subject as a case study. 40% of the examples are used as the training set and the rest is used as the test set.

B. Evaluation Protocol

In the experiments, we use 30% of the original training samples in each dataset as validation set and the rest is used as the actual training set. 10% of the training examples are used to train the initial model M_0 in the central server and the rest is used for online update. We run the online update for a total of 1000 rounds. One each round, we use the proposed active sampling methods to decide whether or not to send the corresponding data. The central server model is updated every 50 rounds, also called a batch. The total number of batches is 200. We report results of different threshold values for completeness. In order to find the best threshold value for each method, we test the trained model on the validation set and choose the threshold value which achieves the highest validation accuracy. In the experiments, we consider the cases with and without the proposed load balancing mechanism. We consider the following three metrics for comparing different methods,

- Communication cost: the total number of bits communicated by all the devices. It is the sum of the bits communicated by each individual device during the online update.
- Test accuracy: the accuracy of model on the test dataset after the online update.
- Energy consumption: the energy consumption of the all the devices during the online update. The energy consumption consists of the energy consumption for both computation and communication.

C. Baselines

We consider the following baselines in the experiments,

- Local Prediction (LP): we only utilize the data on each device to predict the labels of the test dataset. The total communication cost is minimal in this case, however the accuracy suffers since we cannot use the information from all the devices.
- Passive Learning (PL): Each device sends all the data to the central server which incurs a large communication cost.
- Random Sampling (RS): We consider sending γ% (γ = 50) of the local data collected by each device to the central server. On each round, we sample from a Bernoulli($\gamma/100$) and decide whether or not to send the measurement based on the sampling output.

D. Implementation details

The neural network for each device consists of three layers with one layer for computing the local probability. The cloud specific model consists of two layers. We adopt Adam as the optimizer. The learning rate is set to be 0.01 and decays exponentially with a factor of 0.9. We train the initial models

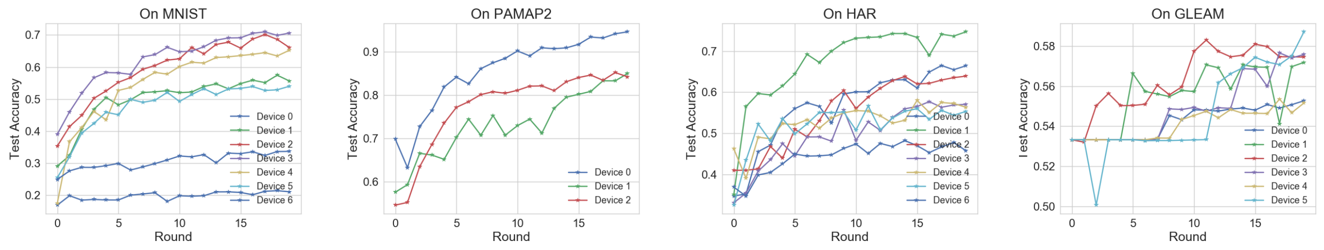


Fig. 2: **Results of local prediction on the four datasets during the online update**.

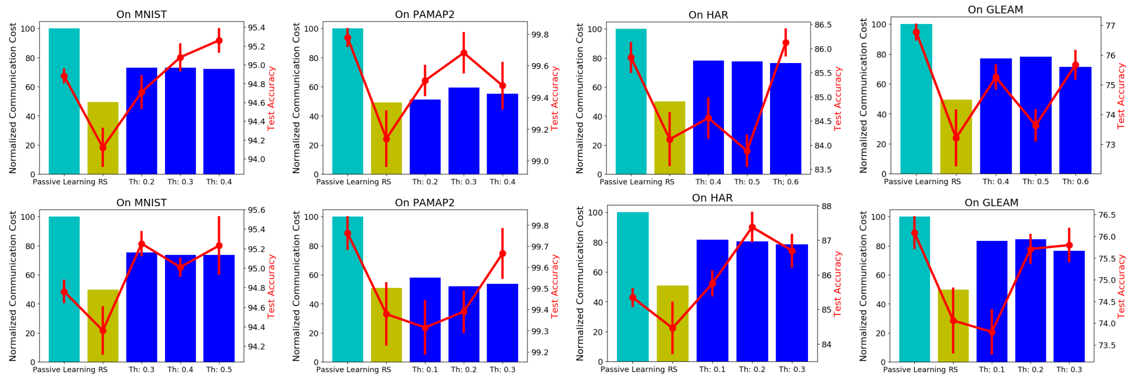


Fig. 3: **Normalized communication cost and final test accuracy with local uncertainty and load balance. The bars show the communication cost of different methods. Blue bars show the results of the proposed methods with different thresholds (Th). The red line is the test accuracy. First row: local uncertainty with least confident function. Second row: local uncertainty with confidence margin function. Yellow bar: random sampling (RS). The results were averaged over 5 runs.**

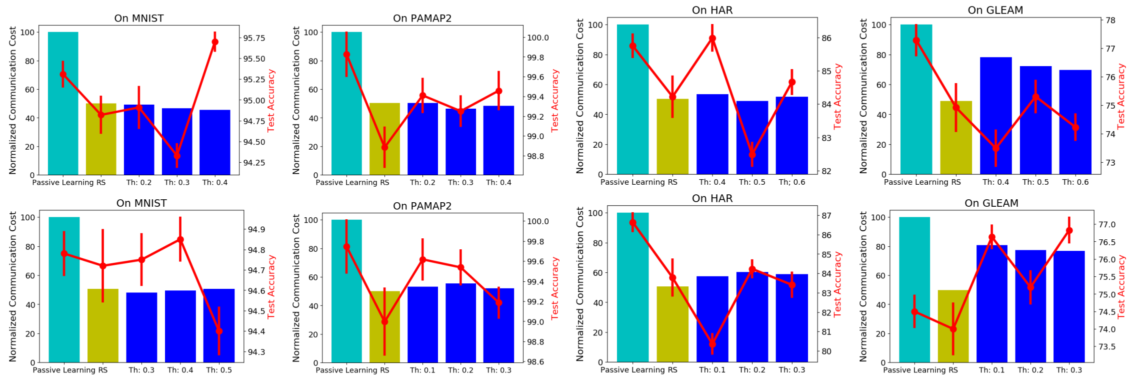


Fig. 4: **Normalized communication cost and final test accuracy of delayed global uncertainty and load balance. The bars show the communication cost of different methods. Blue bars show the results of the proposed methods with different thresholds (Th). The red line is the test accuracy. First row: delayed global uncertainty with least confident function. Second row: delayed global uncertainty with confidence margin function. Yellow bar: random sampling (RS). The results were averaged over 5 runs.**

for a total of 10 epochs. No regularization is used in the experiments. All the models are implemented in Tensorflow [3]. We train the model $S(\mathbf{x}_k|\mathbf{x}_j, y; \beta)$ which is a two-layer neural network for synthesizing measurements on the training dataset using mean squared error. For simplicity, we use the measurements from device 1 to predict the measurements for all other devices. During the online update, if the data of device 1 is missing, we impute the missing value with the historical average.

E. Numerical Results

Do we need measurement from all sensors to learn an accurate predictive model in the central server? In Fig 2, we show the results of local predictions on the four benchmarks. We make two observations based on the results. First, if we only train models locally using the data from one device, the models suffer great loss in accuracy. This is due to the fact that

Dataset \ Method	LU	DGU	RS
MNIST [28]	95.24±0.18	94.90±0.22	94.30±0.21
PAMAP2 [29]	99.52±0.23	99.34±0.13	98.59±0.29
HAR [30]	84.69±0.48	84.76±0.40	83.47±0.76
GLEAM [31]	76.93±0.48	77.34±0.57	74.55±0.86

TABLE I: Test accuracy (%) without load balance mechanism. For each method (LU, DGU), the best threshold value and acquisition function is found on the validation set.

Dataset \ Method	LU	DGU	RS
MNIST [28]	95.26±0.13	95.70±0.12	94.30±0.21
PAMAP2 [29]	99.68±0.13	99.61±0.21	98.59±0.29
HAR [30]	87.37±0.44	85.98±0.41	83.47±0.76
GLEAM [31]	75.79±0.40	76.82±0.38	74.55±0.86

TABLE II: Test accuracy (%) with load balance mechanism. For each method (LU, DGU), the best threshold value and acquisition function is found on the validation set.

each device only captures a subset of the full feature vector which cannot approximate the underlying distribution between the input and the label. This also indicates the necessity to communicate the local data to a central server and train models based on data from all the devices. Second, the accuracy consistently improves with more and more incoming data. This implies the importance to adjust the trained model with newly collected measurements.

How does different active sampling methods work for distributed training in heterogeneous sensor networks? In Figure 3 - Figure 4, we show the results of the proposed active sampling methods on the four datasets with the load balancing mechanism. For the bar plots, the total communication round is normalized against the passive learning. The normalized communication cost and test accuracy is calculated after the online update. We show the results with different threshold values for comprehensiveness. In practice, we can use the best threshold value found on the validation set.

Active Sampling vs. Passive Learning As shown in Figure 3 and Figure 4, we note that, as expected, compared with passive learning the proposed active learning methods typically only suffer about 1% test accuracy loss while reducing the communication cost by 50%. This is because in each round the proposed active sampling methods only send informative data about which the current central server model are most uncertain how to label. Thus the central server model can be better adapted to the incoming user data with the proposed approach. Thus we can greatly reduce the communication cost with the proposed approaches without accuracy loss for training in heterogeneous mobile networks.

Active Sampling vs. Random Sampling In Table I and II, we report the test accuracy with the best threshold value and acquisition function found on the validation set. The results are averaged over 5 runs and the standard deviation is reported. From the tables, we can see that the proposed methods outperform Random Sample (RS) on all the datasets, in some cases with a large margin. Unlike the proposed active sampling method, naive random sampling fails to capture the important data and cannot identify the change points.

With Load Balance Mechanism vs. Without Load Balance Mechanism The results in Table I and Table II show the best accuracy with and without the load balancing mechanism. We found that the accuracy is typically lower without the load balancing mechanism except on the *GLEAM* dataset. For the *GLEAM* dataset, we found that for LU the best acquisition function is confidence margin function with threshold value $\gamma = 0.2$. For DGU the best acquisition function is confidence margin function with threshold value $\gamma = 0.1$. We further observe that the total communication cost on the *GLEAM* dataset with the best acquisition function and the threshold value is much higher than the case of with load balancing mechanism, the extra data communication can explain the slight higher accuracy in the case of without load balance. In summary, the proposed load balance mechanism can reduce the sensitivity of the model to different threshold values and avoid the situation that the model will bias towards a small subset of the devices.

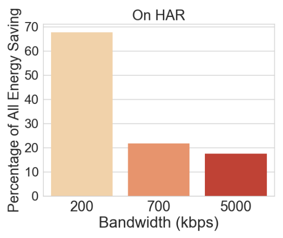
Acquisition Function Interestingly, we also observe that different acquisition functions yield similar performance on different datasets. This implies that the proposed active sampling methods are robust to the choices of acquisition functions, though the thresholds need to be adjusted to balance the trade-off between communication cost and test accuracy. The insensitivity of proposed method to the choice of acquisition functions is of great importance for practical applications since it reduces the time for searching the optimal configurations.

LU vs. DGU Finally when we compare local uncertainty (LU) with delayed global uncertainty (DGU), we note that at the end of the online update, LU typically sends more data to the central server than DGU which leads to a higher test accuracy. We conjecture that this is because in DUG the data from other devices decrease the uncertainty of the newly collected data which results in a lower upload communication cost and test accuracy compared with LU. These results suggest that, we can select the methods based on the actual network condition and the test accuracy requirement. If the download bandwidth is high enough to communicate the sensor data from the central server to the local sensors, DGU allows us to trade for the excessive download bandwidth for a lower upload communication cost. Otherwise, LU can achieve a better test accuracy with a slightly increase in upload communication.

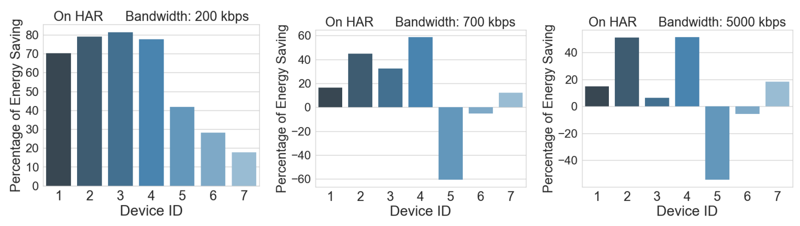
VI. Practical Deployment

A. Setup

To demonstrate the benefits the proposed active sampling method for real-world applications, we deploy a prototype network in our lab and measure the energy and communication saving of the proposed active learning approach compared with the passive learning approach. We consider a common topology setting in various Internet-of-Things (IoT) or mobile applications such as Smart Home, Smart Building, etc. As shown in Figure 7, the prototype network locates in our 6m×10m lab, consisting of seven Raspberry Pi 3B (RPi 3B) devices and one conventional laptop as a central server. All devices are connected to a router and form a local heterogeneous mobile network. The RPis communicate with the central

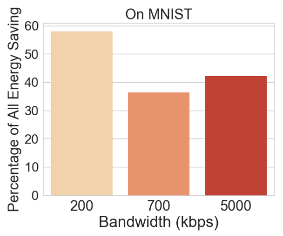


(a)

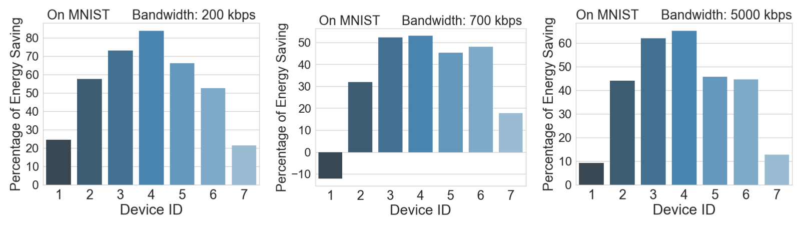


(b)

Fig. 5: **Energy saving for human activity recognition. Left: The energy saving of all devices. Right: the energy saving of each individual device.**



(a)



(b)

Fig. 6: **Energy saving for image classification. Left: The energy saving of all devices. Right: the energy saving of each individual device.**

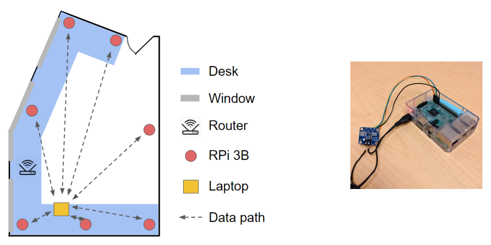


Fig. 7: **Left: deployment topology of the prototype network. Right: configuration of each RPi 3B, which has a current sensor INA219 attached to it.**

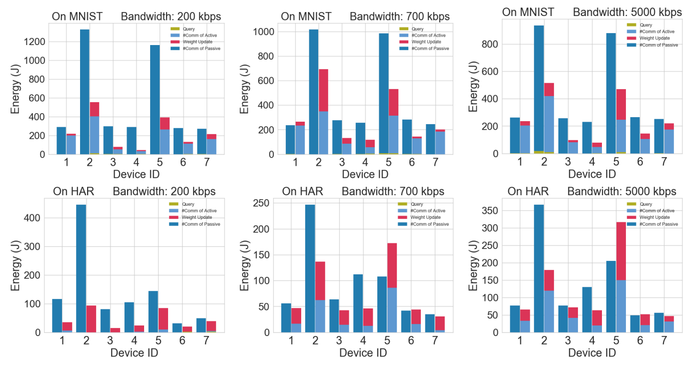


Fig. 8: **Energy consumption breakdown of the passive learning approach and the proposed active sampling approach**.

server using the MQTT protocol which is commonly used in IoT and mobile applications. The network bandwidth is controlled by the *wondershaper* tool [32]. We experiment with communication bandwidths of 200, 700 and 5000 kbps, which correspond to typical bandwidths of constrained Bluetooth, Bluetooth, and WiFi, respectively. For energy measurement, we attach a high side current sensor INA219 [33] to each RPi 3B, as depicted in Figure 7. INA219 reports high side voltage and DC current draw with 1% precision.

The experiments are conducted based on two applications, image classification and human activity recognition with the MNIST dataset and the HAR dataset, respectively. For image classification, we evenly distribute the feature vector on each device. For human activity recognition, each device have the measurements of one particular sensor. The neural network architecture is consistent with the setup described in Section V-D. We repeat the online update round for ten epochs and compare the active sampling approach with the passive learning approach in terms of energy consumption. For the active sampling approach, we use LU for simplicity as it has been shown that LU can achieve similar accuracy as DGU in Section V-E. Entropy is used as the acquisition function for calculating the informativeness of the samples.

B. *Results and Discussion*

All energy saving The results of energy saving of all the devices are shown in Figure 5a and 6a. It is clear that the proposed active sampling approach can greatly reduce the energy consumption of the system. For human activity recognition, the energy saving is 67.68% , 21.55%, and 17.34% with a bandwidth of 200 kbps, 700 kbps and 5000 kbps, respectively. For image classification, the energy saving is 58.01%, 36.37%, and 42.04% with a bandwidth of 200 kbps, 700 kbps and 5000 kbps, respectively. Notably the proposed active sampling approach achieves a large energy consumption reduction than the passive learning approach under a low bandwidth (200 kbps). This is particularly important for real-world mobile applications due to the network bandwidth is often limited.

Energy saving of each device We further show the energy saving of each individual device in Figure 5b and Figure 6b. It can be seen that due to the heterogeneity of the devices (since they have different features), different devices have different patterns of energy saving. Some devices even have a higher energy consumption with the active sampling approach due to the additional computation. This indicates that not all the devices are equally important for the task. With the proposed active sampling approach, we can leverage this fact to reduce the data communication of the unimportant devices to save the energy of the whole system.

Energy consumption breakdown We show the energy consumption breakdown of the proposed active sampling approach and the passive learning approach in Figure 8 under different network conditions. For the passive learning approach, the energy consumption consists of sample query and data communication. For the active sampling approach, the energy consumption consists of sample query, local weight update and data communication. We can see that for both approaches, the energy consumed by sample query is negligible. The total energy consumption of local weight update and data communication of the active sampling approach is usually less than the energy consumption of the passive learning approach.

VII. Conclusion

In this paper, we propose an active sampling method for communication efficient training in heterogeneous sensor networks. Instead of sending all the local data to the central server, the proposed method identifies and sends only informative informative measurements. We also propose a synthesized measurement approach to accurately recover the missing measurements which allows the central server model can be updated based on a small set of informative data. Finally, we propose a load balance mechanism to distribute the communication cost across all the devices. Experimental results show that the proposed active sampling methods can reduce the communication cost by up to 53% and energy consumption by up to 67% without accuracy degradation.

VIII. Acknowledgement

This work was supported in part by CRISP, one of six centers in JUMP, an SRC program sponsored by DARPA. This work is also supported by NSF CHASE-CI #1730158, NSF FET #1911095, NSF CC* NPEO #1826967. The paper was also funded in part by SRC AIHW grants.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, "Distributed regression: an efficient framework for modeling sensor network data," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*. ACM, 2004, pp. 1–10.

[3] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[4] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets."

[5] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[6] B. Ying, K. Yuan, and A. H. Sayed, "Supervised learning under distributed features," *IEEE Transactions on Signal Processing*, vol. 67, no. 4, pp. 977–992, 2018.

[7] A. Thomas, Y. Guo, Y. Kim, B. Aksanli, A. Kumar, and T. S. Rosing, "Hierarchical and distributed machine learning inference beyond the edge," in *2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC)*. IEEE, 2019, pp. 18–23.

[8] N. AlDuaij, A. Van't Hof, and J. Nieh, "Heterogeneous multi-mobile computing," in *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2019.

[9] Y. Zhang, M. J. Wainwright, and J. C. Duchi, "Communication-efficient algorithms for statistical optimization," in *Advances in Neural Information Processing Systems*, 2012, pp. 1502–1510.

[10] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018.

[11] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 328–339.

[12] S. R. Madden, "The design and evaluation of a query processing architecture for sensor networks," Ph.D. dissertation, University of California, Berkeley, 2003.

[13] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu, "Deep learning for sensor-based activity recognition: A survey," *Pattern Recognition Letters*, vol. 119, pp. 3–11, 2019.

[14] S. Bauer, D. D. Clark, and W. Lehr, "Understanding broadband speed measurements." Tprc, 2010.

[15] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[16] A. Jain and E. Y. Chang, "Adaptive sampling for sensor networks," in *Proceedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004*. ACM, 2004.

[17] J. Chen, Z. J. Towfic, and A. H. Sayed, "Dictionary learning over distributed models," *IEEE Transactions on Signal Processing*, vol. 63, no. 4, pp. 1001–1016, 2014.

[18] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation: numerical methods*. Prentice hall Englewood Cliffs, NJ, 1989, vol. 23.

[19] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le *et al.*, "Large scale distributed deep networks," in *Advances in neural information processing systems*, 2012, pp. 1223–1231.

[20] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[21] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

[22] Y. Guo, "A survey on methods and theories of quantized neural networks," *arXiv preprint arXiv:1808.04752*, 2018.

[23] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *International conference on machine learning*, 2014, pp. 1000–1008.

[24] 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.

[25] S. Dasgupta, "Two faces of active learning," *Theoretical computer science*, vol. 412, no. 19, pp. 1767–1781, 2011.

[26] K. Konyushkova, R. Sznitman, and P. Fua, "Learning active learning from data," in *Advances in Neural Information Processing Systems*, 2017, pp. 4225–4235.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, 2016.

[28] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[29] A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," in *2012 16th International Symposium on Wearable Computers*. IEEE, 2012, pp. 108–109.

[30] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones."

[31] S. A. Rahman, C. Merck, Y. Huang, and S. Kleinberg, "Unintrusive eating recognition using google glass," in *2015 9th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*. IEEE, 2015, pp. 108–111.

[32] "Wondershaper," https://github.com/magnific0/wondershaper, [Online].

[33] "INA219 High Side DC Current Sensor Breakout," https://www.adafruit.com/product/904, [Online].