Crowd-ML: A Privacy-Preserving Learning Framework for a Crowd of Smart Devices

Jihun Hamm, Adam C. Champion, Guoxing Chen, Mikhail Belkin, Dong Xuan
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210

***Abstract*—Smart devices with built-in sensors, computational capabilities, and network connectivity have become increasingly pervasive. Crowds of smart devices offer opportunities to collectively sense and perform computing tasks at an unprecedented scale. This paper presents Crowd-ML, a privacy-preserving machine learning framework for a crowd of smart devices, which can solve a wide range of learning problems for crowdsensing data with differential privacy guarantees. Crowd-ML endows a crowdsensing system with the ability to learn classifiers or predictors online from crowdsensing data privately with minimal computational overhead on devices and servers, suitable for practical large-scale use of the framework. We analyze the performance and scalability of Crowd-ML and implement the system with off-the-shelf smartphones as a proof of concept. We demonstrate the advantages of Crowd-ML with real and simulated experiments under various conditions.**

I. Introduction

A. Crowdsensing

Smart devices are increasingly pervasive in daily life. These devices are characterized by their built-in sensors (e.g., accelerometers, cameras, and, microphones), programmable computation ability, and Internet connectivity via wireless or cellular networks. These include stationary devices such as smart thermostats and mobile devices such as smartphones or in-vehicle systems. More and more devices are also being interconnected, often referred to as the "Internet of Things." Inter-connectivity offers opportunities for crowds of smart devices to collectively sense and compute at an unprecedented scale. Various applications of crowdsensing have been proposed, including personal health/fitness monitoring, environmental sensing, and monitoring road/traffic conditions (see Section II-A), and the list is currently expanding.

Crowdsensing is used primarily for collecting and analyzing aggregate data from a population of participants. However, more complex and useful tasks can be performed beyond calculation of aggregate statistics, by using machine learning algorithms on crowdsensing data. Examples of such tasks include: learning optimal settings of room temperatures for smart thermostats; predicting user activity for context-aware services and physical monitoring; suggesting the best driving routes; and recognizing audio events from microphone sensors. Specific algorithms and data types for these tasks are different, but they can all be trained in standard unsupervised or supervised learning settings: given sensory features (time, location, motion, environmental measures, etc.), train an algorithm or model that can accurately predict a variable of interest (temperature setting, current user activity, amount of traffic, audio events, etc.). Conventionally, crowdsensing and machine learning are performed as two separate processes: devices passively collect and send data to a central location and analyses or learning procedures are performed at the remote location. However, current generations of smart devices have computing capabilities in addition to sensing. In this paper, we propose to use computing capabilities of smart devices and integrate sensing and learning processes together into a crowdsensing system. As we will show, the integration allows us to design a system with better privacy and scalability.

B. Privacy

Privacy is an important issue for crowdsensing applications. By assuring participants' privacy, a crowdsensing system can appeal to a larger population of potential participants, which increases the utility of such a system. However, many crowdsensing systems in the literature do not employ any privacy-preserving mechanism (see Section II-B), and existing mechanisms used in crowdsensing (see [1]) are often difficult to compare qualitatively across different systems or data types. In the last decade, differential privacy has gained popularity as a formal quantifiable measure of privacy risk in data publishing [2]–[4]. Briefly, differential privacy measures how much the outcome of a procedure changes probabilistically by the presence or absence of any single subject in the original data. The measure provides an upper bound on privacy loss *regardless of* any prior knowledge an adversary might have. While differential privacy has been applied in data publishing and machine learning, (see Section II-B), it has not been broadly adopted in crowdsensing systems. In this paper, we integrate differentially private mechanisms into the crowdsensing system as well, which can provide strong protection against various types of possible attacks (see Section III-C).

C. Proposed work

This paper presents Crowd-ML, a privacy-preserving machine learning framework for crowdsensing system that consists of a server and smart devices (see Fig. 1). Crowd-ML is a distributed learning framework that integrates sensing, learning, and privacy mechanisms together and can build classifiers or predictors of interest from crowdsensing data using computing capabilities of devices with formal privacy guarantees.

Algorithmically, Crowd-ML learns a classifier or predictor via distributed incremental optimization. Optimal parameters

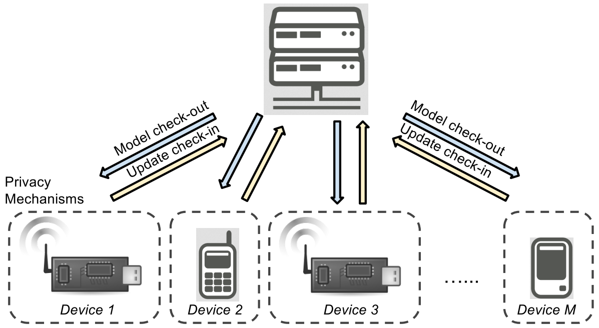

Fig. 1: Crowd-ML consists of a server and a number of smart devices. The system integrates sensing, learning, and privacy mechanisms together to learn a classifier or predictor from device-generated data in an online and distributed way with formal privacy guarantees.

of a classifier or predictor are found by minimizing the risk function associated with a given task [5] (see Section III-A for details). Specifically, the framework finds optimal parameters by incrementally minimizing the risk function using a variant of stochastic (sub)gradient descent (SGD) [6]. Unlike batch learning, SGD requires only the gradient information to be communicated between devices and a server, which has two important consequences: 1) computation load can be distributed among the devices, enhancing scalability of the system; 2) private data of the devices need not be communicated directly, enhancing privacy. By exploiting these two properties, Crowd-ML efficiently learns a classifier or predictor from a crowd of devices with a guarantee of ϵ-differential privacy. The differential privacy mechanism is applied locally on each device using Laplace noise for the gradients and exponential mechanisms for other information (see Section III-C).

We show advantages of Crowd-ML by analyzing its scalability and privacy-performance trade-offs (Section IV), and by testing the framework with demonstrative tasks implemented on Android smartphones and in simulated environments under various conditions (see Section V).

In summary, we make the following contributions:

- We present Crowd-ML, a general framework for machine learning with smart devices from crowdsensing data with many potential applications.
- We show differential privacy guarantees of Crowd-ML that provide a strong privacy mechanism against various types of attacks in crowdsensing. To the best of our knowledge, Crowd-ML is the first general framework that integrates sensing, learning, and differentially private mechanisms for crowdsensing.
- We analyze the framework to show that the cost of privacy preservation can be minimized and that the computational and communication overheads on devices are only moderate, allowing a large-scale deployment of the framework.
- We implement a prototype and evaluate the framework with a demonstrative task in a real environment as well as large-scale experiments in a simulated environment.

The remainder of this paper is organized as follows. We first review related work in Section II. Section III describes the Crowd-ML framework. Section IV analyzes Crowd-ML in terms of privacy-performance trade-off, computation, and communication loads. Section V presents an implementation of Crowd-ML and experimental evaluations. We discuss remaining issues and conclude in Section VI.

II. Related Work

Crowd-ML integrates distributed learning algorithms and differential privacy mechanisms into a crowdsensing system. In this section, we review related work in crowdsensing and learning systems, and privacy-preserving mechanisms.

A. Crowdsensing and learning

There is a vast amount of work in crowdsensing, and we focus on the systems aspect of previous work with representative papers (we refer the reader to survey papers [7] and [1]). Crowdsensing systems aim to achieve mass collection and mining of environmental and human-centric data such as social interactions, political issues of interest, exercise patterns, and people's impact on the environment [8]. Examples of such systems include Micro-Blog [9], PoolView [10], BikeNet [11], and PEIR [12]. Data collected by crowdsensing can also be used to mine high-level patterns or to predict variables of interest using machine learning. Applications of learning applied to crowdsensing include learning bus waiting times [13] and recognizing user activities (see [14] for a review). Jigsaw [15] and Lifestreams [16] also use pattern recognition in sensed data from mobile devices. From the systems perspective, these work use devices to passively sense and send data to a central server on which analyses take place, which we will refer to as the *centralized* approach. In contrast, sensing and learning can be performed purely inside each device without a server, which we call the *decentralized* approach. For example, SoundSense [17] learns a classifier on a smartphone to recognized various audio events without communicating with the back-end. Mixed centralized and decentralized approaches are also used in [18], [19], where a portion of the computation is performed offline on a server. CQue [18] provides a query interface for privacy-aware probabilistic learning of users' contexts, and ACE [19] uses static association rules to learn users' contexts. System-wise, our work differs from these centralized or decentralized approaches as we use a *distributed* approach to perform learning via the devices and the server together, which improves privacy and scalability of the system. We are not aware of any other crowdsensing system that takes a similar approach. Also, the cited papers are oriented towards novel applications, but our work focuses on a general framework for learning a wide range of algorithms and applications.

Crowd-ML also builds on recent advances in incremental distributed learning [20], [21], which show that a near-optimal convergence rate is achievable despite communication delays. A privacy-preserving stochastic gradient descent method is presented briefly in [22]. Unlike the latter, we present a complete framework for privacy-preserving multi-device learning with performance analyses and demonstrations in real environments.

B. Privacy-preserving mechanisms

Privacy is an important issue in data collection and analysis. In particular, preserving privacy of users' locations has been studied by many researchers (see [23] for a survey). To preserve privacy of general data types formally, several mechanisms such as k-anonymity [24] and secure multiparty computation [25] have been proposed for data publishing [26] and also for participatory sensing [1]. Recently, differential privacy [2]–[4] has addressed several weaknesses of k-anonymity [27], and gained popularity as a quantifiable measure of privacy risk. Differential privacy has been used for a privacy-preserving data analysis platform [28], for sanitization of learned model parameters from data [29], and for privacy-preserving data mining from distributed time-series data [17]. So far, formal and general privacy mechanisms have not been adopted broadly in crowdsensing. Among the crowdsensing systems cited in the previous section ([9]–[13], [15]–[19], [30], [31]), only [10], [12], [18] provide privacy mechanisms, of which only [10] addresses the privacy more formally. To our best knowledge, Crowd-ML is the first framework to provide formal privacy guarantees in general crowd-based learning with smart devices.

III. Crowd-ML

In this section, we describe our Crowd-ML framework in detail: system, algorithms, and privacy mechanisms.

A. System and workflow

The Crowd-ML system consists of a server and multiple smart devices that are capable of sensory data collection, numerical computation, and communication over a public network with the server (see Fig. 1). The goal of Crowd-ML is to learn a classifier or predictor of interest from crowdsensing data collectively by multiple devices. A wide-range of classifiers or predictors can be learned by minimizing an empirical risk associated with a given task, a common method in statistical learning [5]. Formally, let $x \in \mathbb{R}^D$ be a feature vector from preprocessing sensory input such as audio, video, accelerometer, etc, and y be a target variable we aim to predict from x, such as user activity. For regression, y can be a real number and for classification, y is a discrete label $y \in \{1, ..., C\}$ with C classes. We define data as N pairs of (feature vector, target variable) generated i.i.d. from an unknown distribution by all participating devices up to the present:

$$\mathcal{D} = \{(x_1, y_1), ..., (x_N, y_N)\}. \tag{1}$$

Suppose we use a classifier/predictor $h(x; w)$ with a tunable parameter vector w, and a loss function $l(y, h(x; w))$ to measure the performance of the classifier/predictor with respect to the true target y. A wide range of learning algorithms can be represented by h and l, e.g., regression, logistic regression, and Support Vector Machine (see [32] for more examples). If there are M smart devices, we find the optimal parameters w of the classifier/predictor by minimizing the empirical risk over all M devices:

$$\mathcal{R}(w) = \sum_{m=1}^{M} \frac{1}{|\mathcal{D}_m|} \sum_{(x,y)\in\mathcal{D}_m} l(h(x; w), y) + \frac{\lambda}{2}\|w\|^2, \tag{2}$$

where $\mathcal{D}_m$ is a set of samples generated from device m only, and $\frac{\lambda}{2}\|w\|^2$ is a regularization term. This risk function (2) can be minimized by many optimization methods. In this work we use stochastic (sub)gradient descent (SGD) [33] which is one of the simplest optimization methods and is also suitable for large-scale learning [32], [34]. SGD minimizes the risk by updating w sequentially

$$w(t + 1) \leftarrow \Pi_{\mathcal{W}} \left[w(t) - \eta(t)g(t)\right], \tag{3}$$

where $\eta(t)$ is the learning rate, and $g(t)$ is the gradient of the loss function

$$g = \nabla_w l(h(x; w), y), \tag{4}$$

evaluated with the sample (x, y) and the current parameter $w(t)$. We assume the parameter domain $\mathcal{W}$ is a d-dimensional ball of some large radius R, and the projection is $\Pi_{\mathcal{W}} = \min(1, R/\|w\|)w$. By default, we use the learning rate

$$\eta^{(t)} = \frac{c}{\sqrt{t}}, \tag{5}$$

where c is a constant hyperparameter. When computing gradients, we use a 'minibatch' of b samples to compute the averaged gradient

$$\tilde{g} = \frac{1}{b} \sum_i \nabla_w l(h(x_i; w), y_i), \tag{6}$$

which plays an important role in the performance-privacy trade-off and the scalability (Section IV). In Crowd-ML, risk minimization by SGD is performed by distributing the main workload (=computation of averaged gradients) to M devices. Note that each device generates data and compute gradients using its own data. The workflow is described in Fig. 2.

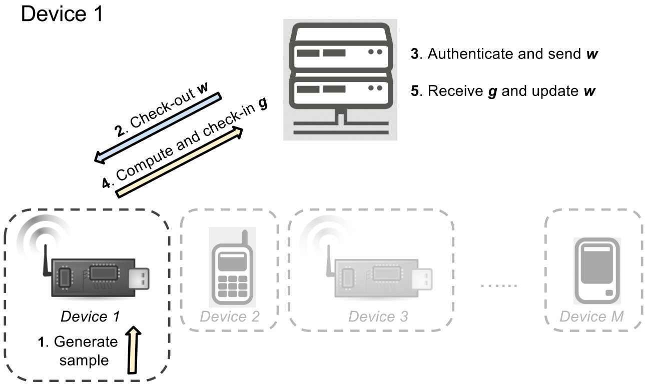

Fig. 2: Crowd-ML workflow. 1. A device preprocesses sensory data and generates a sample(s). 2. When the number of samples $\{(x, y)\}$ exceeds a certain number, the device requests current model parameters w from the server. 3. The server authenticate the device and sends w. 4. Using w and $\{(x, y)\}$, the device computes the gradient g and send it to the server using privacy mechanisms. 5. The server receives the gradient g and updates w. While one device is performing routines 1-5, another device(s) are allowed to perform the same routines asynchronously. Devices can join or leave the task at any time.

Algorithm 1 Device side

```
Input: privacy levels $\epsilon_g, \epsilon_e, \epsilon_{y^k}$, minibatch size $b$, max buffer size $B$, classifier model ($C$, $h$, $l$, $\lambda$ from Eq. (2))
Init: set $n_s = 0,\ n_e = 0,\ n_y^k = 0,\ k = 1, ..., C$
Communication to server: $\hat{g}, n_s, \hat{n}_e, \hat{n}_y^k$
Communication from server: $w$
Device Routine 1
  if $n_s \geq B$ then
    stop collection to prevent resource outage
  else
    receive a sample $(x, y)$ (in a regular interval or triggered by events), and add to the secure local buffer
    $n_s = n_s + 1$
  end if
  if $n_s \geq b$ then
    checkout $w$ from the server via HTTPS
    call Device Routine 2.
  end if
Device Routine 2
  Using $w$ from the server and $\{(x, y)\}$ from the local buffer,
  for $i = 1, ..., n_s$ do
    make a prediction $y^{\text{pred}} = h(x_i; w)$
    $n_y^{(y_i)} = n_y^{(y_i)} + 1$
    $n_e = n_e + I[y_i^{\text{pred}} \neq y_i]$
    Incur a loss $l(y^{\text{pred}}, y_i)$
    Compute a subgradient $g_i = \nabla_w l(h(x_i; w))$
  end for
  Compute the average $\tilde{g} = \frac{1}{n_s} \sum_i g_i + \lambda w$
  Sanitize data with Device Routine 3
  Checkin $\hat{g},\ n_s,\ \hat{n}_e\ \hat{n}_y^k,\ k = 1, ..., C$ with server via https
  Reset $n_s = 0,\ n_e = 0,\ n_y^k = 0,\ k = 1, ..., C$
Device Routine 3
  Sample $\hat{g} = \tilde{g} + z$ from Eq. (10)
  Sample $\hat{n}_e = n_e + z$ from Eq. (11)
  Sample $\hat{n}_y^k = n_y^k + z,\ k = 1, ..., C$ from Eq. (12)
```

B. Algorithms

Crowd-ML algorithms are presented in Algorithms 1 and 2. Device Routine 1 collects samples. When the number of samples reaches the minibatch size b, the routine tries to checks out the current model parameters w from the server and calls Device Routine 2. Device Routine 2 computes the averaged gradient from the stored samples and w received from the server, sanitizes information by Device Routine 3, and sends the sanitized information to the server. Device Routine 3 uses Laplace noise and exponential mechanisms (described in the next section) to sanitize the averaged gradient $\hat{g}$, the number of misclassified samples $\hat{n}_e$ and the label counts $\hat{n}_y^k$. Device Routines 1-3 are performed independently and asynchronously by multiple devices.

Server Routine 1 sends out current parameters w when requested and Server Routine 2 receives checkins ($\hat{g}$, n_s, $\hat{n}_e$,$\hat{n}_y^k$) from devices when requested. The whole procedure ends when the total number of iteration exceeds a maximum value $T_{\max}$, or the overall error is below a threshold ρ.

Remark 1: In Device Routine 1, if check-out fails, the device keeps collecting samples and retries check-out later.

Algorithm 2 Server side

```
Input: number of devices $M$, learning rate schedule $\eta(t),\ t = 1, 2, ..., T_{\max}$, desired error $\rho$, classifier model ($C$, $h$, $l$, $\lambda$ from Eq. (2))
Init: $t = 0$, randomized $w$, $N_s^m = 0,\ N_e^m = 0,\ N_y^{k,m},\ m = 1, ..., M,\ k = 1, ..., C$
Stopping criteria: $t \geq T_{\max}$ or $\frac{\sum_m^M N_e^m}{\sum_m^M N_s^m} \leq \rho$
Server Routine 1
  while Stopping criteria not met do
    Listen to and accept checkout requests
    Authenticate device
    Send current parameters $w$ to device
  end while
Server Routine 2
  while Stopping criteria not met do
    Listen to and accept checkin requests
    Authenticate device (suppose it is device $m$)
    Receive $\hat{g},\ n_s,\ \hat{n}_e,\ \hat{n}_y^k,\ k = 1, ..., C.$
    $N_s^m = N_s^m + n_s$
    $N_e^m = N_e^m + \hat{n}_e$
    $N_y^{k,m} = N_y^{k,m} + \hat{n}_y^k$
    $w = w - \eta(t)\hat{g}$
    $t = t + 1$
  end while
```

A prolonged period of network outage for a device can make the parameter outdated for the device, but it does not affect the overall learning critically. Similarly, failure to check-in information with server in Device Routine 2 is non-critical.

Remark 2: In Device Routine 2, we can randomly set aside a small portion of samples as test data. In this case, the misclassification error is computed only from these held-out samples, and their gradients will not be used in the average $\hat{g}$.

Remark 3: In Server Routine 2, more recent update methods [35], [36] can be used in place of the simple update rule (3) without affecting differential privacy nor changing device routines. Similarly, adaptive learning rates [37], [38] can be used in place of (5), which can provide a robustness to large gradients from outlying or malignant devices.

C. Privacy mechanism

In crowdsensing systems, users' private data can be leaked in many ways. System administrators/analysts can violate privacy intentionally, or they may leak private information unintentionally when publishing data analytics. There are also more hostile types of attacks: by malicious devices posing as legitimate devices, by hackers poaching data stored on the server or eavesdropping on communication between devices and servers. Instead of preserving privacy separately for each attack type, we can preserve privacy from all these attacks by a *local* privacy-preserving mechanism that is implemented on each device and sanitizes any information before it leaves the device. A local mechanism assumes that an adversary can potentially access all communication between devices and the server, which subsumes other types of attacks. This is because the other forms of data that are 1) visible to malicious devices, 2) stored in the server, or 3) released in public, are all derived from what is communicated between devices and the server. We adopt a local ϵ-differential privacy as a quantifiable measure of privacy in Crowd-ML. Formally, a (randomized) algorithm that takes data $\mathcal{D}$ as input and outputs f is called ϵ-differentially private if

$$\frac{P(f(\mathcal{D}) \in \mathcal{S})}{P(f(\mathcal{D}') \in \mathcal{S})} \leq e^{\epsilon} \tag{7}$$

for all measurable $\mathcal{S} \subset \mathcal{T}$ of the output range and for all data sets $\mathcal{D}$ and $\mathcal{D}'$ differing in a single item. That is, even if an adversary has the whole data $\mathcal{D}$ except a single item, it cannot infer much more about that item from the output of the algorithm f. A smaller ϵ makes such an inference more difficult, and therefore makes the algorithm more private-preserving. When the algorithm outputs a real-valued vector $f \in \mathbb{R}^D$, its global sensitivity can be defined by

$$S(f) = \max_{\mathcal{D},\mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D}')\|_1. \tag{8}$$

where $\|\cdot\|_1$ is the L_1 norm. A basic result from the definition of differential privacy is that a vector-valued function f with sensitivity $S(f)$ can be made ϵ-differentially private [3] by adding an independent Laplace noise vector z where

$$P(z) \propto e^{-\frac{\epsilon}{S(f)}\|z\|_1}. \tag{9}$$

In Crowd-ML, we consider ϵ-differential privacy of any single (feature,label)-sample[1] revealed by communications from all devices to the server, which are the gradients $\tilde{g}$, the numbers of samples n_s, the number of misclassified samples n_e, and the labels counts n_y^k [2]. The amount of noise required depends on the choice of loss functions. We compute this value for multiclass logistic regression (Table I), but it can be computed similarly for other loss functions as well. By adding element-wise independent Laplace noise z to averaged gradients $\tilde{g}$

$$\hat{g} = \frac{1}{b}\sum_i g_i + z, \;\; P(z) \propto e^{-\frac{\epsilon_g b}{4}|z|}, \tag{10}$$

we have the following privacy guarantee:

TABLE I: Multiclass logistic regression

Prediction	$\arg\max_k w_k' x$
Risk	$\mathcal{R}(w) = \frac{1}{N}\sum_i[-w_{y_i}' x_i + \log \sum_l e^{w_l' x_i}] + \quad \frac{\lambda}{2}\sum_k \|w_k\|^2$
Gradient	$\nabla_{w_k}\mathcal{R} = \frac{1}{N}\sum_i x_i[-I[y_i = k] + P(y = k\|x_i)] + \quad \lambda w_k$

Theorem 1 (Averaged gradient perturbation)**.** *The transmission of $\tilde{g}$ by Eq. (10) is ϵ_g-differentially private.*

See Appendix A for proof.

To sanitize n_e and n_y^k, we add discrete Laplace noise [41] as follows:

$$\begin{aligned} \hat{n}_e &= n_e + z, \; P(z) \propto e^{-\frac{\epsilon_e}{2}|z|}, & (11) \\ \hat{n}_y^k &= n_y^k + z, \; P(z) \propto e^{-\frac{\epsilon_{y^k}}{2}|z|}, & (12) \end{aligned}$$

where $z = 0, \pm 1, \pm 2, \ldots$. These mechanisms has the following privacy guarantees:

Theorem 2 (Error and label counts)**.** *The transmission of n_e and n_y^k by Eqs. (11) and (12) is ϵ_e- and ϵ_{y^k}- differentially private, respectively.*

See Appendix B for proof.

Practically, a system administrator chooses ϵ depending on the desired level of privacy for the data collected. A small ϵ ($\to 0$) may be used for data that users deem highly private such as current location, and a large ϵ ($\to \infty$) may be used for less private data such as ambient temperature.

IV. Analysis

In this section, we analyze the privacy-performance trade-off and the scalability of Crowd-ML in comparison with purely centralized or decentralized approaches of the existing crowd-sensing systems. By design, Crowd-ML achieves differential privacy with little loss of performance ($O(1/b)$), only moderate computation load due to its simple optimization method, and reduced communication load and delay ($O(1/b)$), where b is the minibatch size.

A. Privacy vs Performance

Privacy costs performance – the more private we make the system, the less accurate the outcome of analysis/learning. From Theorem 1, Crowd-ML is ϵ-differentially private by perturbing averaged gradients. The centralized approach can also be made ϵ-differentially private by feature and label perturbation (see Appendix C). Below we compare the impact of privacy on performance between the centralized approach and Crowd-ML. The performance of an SGD-based learning can be represented by its rate of convergence to the optimal value/parameters $\mathbb{E}[l(w(t)) - l(w^*)]$ at iteration t, which in turn depends on the properties of the loss $l(\cdot)$ (such as Lipschitz-continuity and strong-convexity) and the step size $\eta(t)$, with the best known rate being $O(1/t)$ [42]. When other conditions are the same, the convergence rate is roughly proportional $\mathbb{E}[l(w(t)) - l(w^*)] \propto G^2$ to the amount of noise in the estimated gradient $G^2 = \sup_t \mathbb{E}[\|\hat{g}(t)\|^2]$ [43]. For Crowd-ML, we have from (10)

$$\mathbb{E}[\|\hat{g}\|^2] = \mathbb{E}[\|\tilde{g}\|^2] + \mathbb{E}[\|z\|^2] = \frac{1}{b}\mathbb{E}[\|g\|^2] + \frac{32D}{(b\epsilon_g)^2}, \tag{13}$$

where the first term is the amount of noise due to sampling, and the latter is due to Laplace noise mechanism with D-dimensional features. By choosing a large enough minibatch size b, the impact of sampling noise and Laplace noise can be made arbitrarily small[3]. In contrast, the centralized approach has to add Laplace noise of *constant* variance $\frac{8}{\epsilon^2}$ to each feature and perturb labels with a constant probability (Appendix C). Regardless of which optimization method is used (SGD or not), the centralized approach has no means of mitigating the negative impact of constant noise on the accuracy of learned model, which will be especially problematic with a small ϵ.

[1] This event-level privacy is weaker than the privacy of all (possibly infinite) samples of a subject over the course of time. We will consider only the former privacy in this paper and refer the readers to [39] for a discussion of the difficulty of the latter and also to [40] for a possible solution to the repeated measurement problem.

[2] The communication from the server to devices $\{w(t)\}$ can be reconstructed by (3) from $\{g(t)\}$, and therefore is redundant to consider.

[3] Although a larger batch size means fewer updates given the same number of samples N, and too large a batch size can negatively affect the convergence rate (see [44] for discussion).

In the decentralized approach, a device need not interact with a server, and is almost free of privacy concerns. However, the increased privacy comes at the cost of performance. In Crowd-ML and the centralized approach, samples pooled from all devices are used in the learning process, whereas in the decentralized approach, each device can use only a fraction ($\sim 1/M$) of samples. This undermines the accuracy of a model learned by the decentralized approach. For example, it is known from the VC-theory for binary classification problems that the upper-bound of the estimation error with a $1/M$-times smaller sample size is $\sqrt{M/\log M}$-times larger [45].

B. Scalability

Scalability is determined by computation and communication loads and latencies on both device and server sides. We compare these factors between centralized, crowd, and decentralized learning approaches.

1) Computation load: For all three approaches, we assume the same preprocessing is performed on each device to compute features from raw sensory input or metadata. On the device side, the centralized learning approach requires generation of Laplace noise per sample on the device. The crowd and the decentralized approaches perform partial and full learning on the device, respectively, and requires more processing. Specifically, Crowd-ML requires computation of a gradient per sample, a vector summation (for averaging) per sample, and generation of Laplace random noise per minibatch. A low-end smart device capable of floating-point computation can perform these operations. The decentralized learning approach can use any learning algorithms including SGD (similar to Crowd-ML). However, if the decentralized approach is to make up for the smaller sample size ($1/M$) compared to Crowd-ML, it may require more complex optimization methods which results in higher computation load. For all three approaches, the number of devices M do not affect per-device computation load. Computational load on the server is also different for these approaches. The centralized approach puts the highest load on the server, as all computations take place on the server. In contrast, Crowd-ML puts minimal load on the server which is the SGD update (3), since the main computation is performed distributed by the devices.

2) Communication load: To process incoming streams of data from the device in time, the network and the server should have enough throughput. The centralized learning approach requires N number of samples to be sent over the network to the server. For Crowd-ML with a minibatch size of b, devices send N/b gradients altogether, and receives the same number of current parameters, both of the same dimension as a feature vector. Therefore, the data transmission is reduced by a factor of $b/2$ compared to the centralized approach.

3) Communication latency: When using a public (and mobile) network, latency is non-negligible. In the centralized approach, latency may not be an issue, since the server need not required to send any real-time feedback to the devices. In Crowd-ML, latency is an issue that can affect its performance. There are three possible delays that add up to the overall latency of communication:

- Request delay (τ_{req}): time since the check-out request from a device until the receipt of the request at the server
- Check-out delay (τ_{co}): time since the receipt of a request at the server and the receipt of the parameter at the device
- Check-in delay (τ_{ci}): time since the receipt of the parameters at the device until the receipt of the check-ins at the server.

Due to delays, if a device checks out the parameter w at time t_0 and checks in the gradient $\hat{g}$ and the server receives $\hat{g}$ at time $t_0 + \tau_{\text{co}} + \tau_{\text{ci}}$, the server may have already updated the parameters w multiple times using the gradients from other devices received during this time period. This number of updates is roughly $(\tau_{\text{co}} + \tau_{\text{ci}}) \times MF_s/b$, where M is the number of devices, F_s is the data sampling rate per device, and $1/b$ is the reduction factor due to minibatch. Again, choosing a large batch size b relative to MF_s can reduce the latency. While exact analysis of impact of latency is difficult, there are several related results known in the literature without considering privacy. Nedić et al. proved that delayed asynchronous incremental update converges with probability 1 to an optimal value, assuming a finite maximum latency. Recent work in distributed incremental update [20], [21] also shows that a near-optimal convergence rate is achievable despite delays. In particular, Dekel et al. [21] shows that delayed incremental updates are scalable with M by adapting the minibatch size.

V. Evaluation

In this section, we describe a prototype of Crowd-ML implemented on off-the-shelf Android phones and report results of experiments in real and simulated environments.

A. Implementation

We implement a Crowd-ML prototype with three components: a Web portal, commercial off-the-shelf smart devices, and a central server. On the device side, we implement Algorithm 1 on commercial off-the-shelf smartphones as an app using Android OS 4.3+. Our prototype uses smartphones, but will be easily ported to other smart device platforms. On the server side, we implement Algorithm 2 on a Lenovo ThinkCentre M82 machine with a quad-core 3.2 GHz Intel Core i5-3470 CPU and 4 GB RAM running Ubuntu Linux 14.04. The server runs the Apache Web server (version 2.4) and a MySQL database (version 5.5).

Also on the server side, our Crowd-ML prototype provides a Web portal over HTTPS where users can browse ongoing crowd-learning tasks and join them by downloading the app to their smart devices. To enhance transparency, details of tasks (objective, sensory data collected, labels collected, and learning algorithms used) and our privacy mechanisms is explained. It also displays timely statistics about crowd-learning applications such as error rates and activity label distributions, which are differentially private. We implement the portal in Python using the Django[4] Web application framework and Matplotlib[5] for statistical visualization.

[4] http://www.djangoproject.com

[5] http://matplotlib.org

B. Activity Recognition in Real Environments

In this experiment, we perform activity recognition on smart devices. The purpose of this demonstration is to show Crowd-ML working in a real environment, so we choose a simple task of recognizing three types of user activities ("Still", "On Foot", and "In Vehicle"). We install a prototype Crowd-ML application on 7 smartphones (Galaxy Nexus, Nexus S, and Galaxy S3) running Android 4.3 or 4.4. The seven smartphones are carried by college students and faculty over a period of a few days. The devices' triaxial accelerometers are sampled at 20 Hz. In this demonstration, we avoid manual annotation of activity labels to facilitate data acquisition, and instead use Google's activity recognition service to obtain ground truth labels. Acceleration magnitudes $|a| = \sqrt{a_x^2 + a_y^2 + a_z^2}$ are computed continuously over 3.2 s sliding windows. Feature extraction is performed by computing the 64-bin FFT of the acceleration magnitudes. We set the sampling rate $F_s = 1/30$ Hz, that is, a feature vector x and its label y is generated every 30 s. However, to avoid getting highly correlated samples and to increase diversity of features, we collect a sample only when its label has changed from its previous value. For example, samples acquired during sleeping are discard automatically as they all have "Still" labels. This lowers the actual sampling rate to about $F_s = 1/352$ Hz (or about once every six minutes). With this low rate, no battery problem was observed.

We use 3-class logistic regression (Table I) with $\lambda = 0, b = 1, \epsilon^{-1} = 0$ and a range of η values. Repeated experiments with different parameters are time-consuming, and we leave the full investigation to the second experiment in a simulated environment. In Fig. 3, we shows the collective error curves for the first 300 samples from the 7 devices. The error is a time-averaged misclassification error as the learning progresses: $\mathrm{Err}(t) = \frac{1}{t}\sum_{i=1}^{t} I[y_i \neq y_i^{\mathrm{pred}}(w_i)]$. The error curves for different learning rates (5) are very similar, and virtually converge after only 50 samples (=7 samples per device). This experiment is a proof-of-concept that Crowd-ML can learn a common classifier fast from only a small number of samples per user.

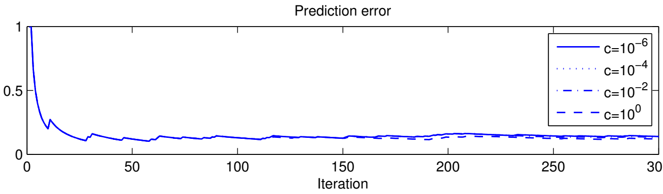

Fig. 3: Time-averaged error across all devices for activity recognition task.

C. Digit/Object Recognition in Simulated Environments

To evaluate Crowd-ML under various conditions, we perform a series of experiments on handwritten digit recognition and visual object recognition. Since the two results are quite similar, we only describe the digit recognition results (object recognition result is in Appendix D). The MNIST dataset[6] consists of 60,000 training and 10,000 test images of handwritten digits (0 to 9), which is a standard benchmark dataset for learning algorithms. The task is to classify a test image as one of the 10 digit classes. The images from MNIST data are preprocessed with PCA to have a reduced dimension of 50 and L_1 normalized. In this experiment, we compare the performance of centralized, Crowd-ML, and decentralized learning approaches using the same data and classifier (multiclass logistic regression), under different conditions such as privacy level ϵ, minibatch size b, and delays. To test the algorithms with a full control of parameters, we run the algorithms in a simulated environment instead of on a real network. We can therefore choose the number of devices and maximum delays arbitrarily. For simplicity, we set $\tau = \tau_{\mathrm{req}} = \tau_{\mathrm{co}} = \tau_{\mathrm{ci}}$ (Section IV-B3). The τ is the maximum delay, and the actually delays are sampled randomly and uniformly from $[0, \tau]$ for each communication instance.[7]

All results in this section are averaged test errors from 10 trials. For each trial, assignment of samples, order of devices, perturbation noise, and amounts of delay are randomized. Test errors are computed as functions of the iteration (the number of samples used) up to five passes through the data. Hyperparameters λ (Table I) and c (5) are selected from the averaged test error from 10 trials. We set the number of devices $M = 1000$. Consequently, each device has 60 training and 10 test samples on average.

Fig. 4 compares the performance of the centralized, crowd, and decentralized learning approaches, without privacy or delay ($\epsilon^{-1} = 0,\ b = 1,\ \tau = 0$). The error of centralized batch training is the smallest (0.1), in a tie with Crowd-ML. The error curve of Crowd-ML converges to the same low value as the centralized approach. It shows that incremental update by SGD in Crowd-ML is as accurate as batch learning when privacy and delay are not considered. In contrast, the error curve of the decentralized approach converges at a slower rate and also converges to a high error (~ 0.5), despite using the same overall number of samples as other algorithms, due to the lack of data sharing.

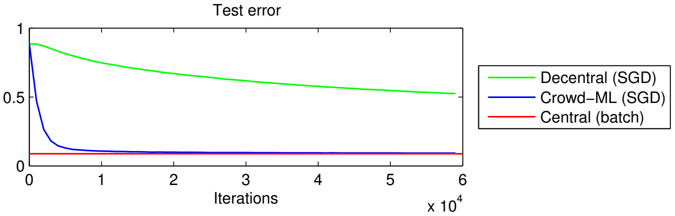

Fig. 4: Comparison of test error for centralized, crowd, and decentralized learning approaches, without delay or privacy consideration. The curves show how error decreases as the number of iterations (number of samples used) increases over time. The batch algorithm is not incremental and therefore is a constant.

We perform tests with varying levels of privacy ϵ. The privacy impacts the centralized approach via (15) and (16)[8]

[6] http://yann.lecun.com/exdb/mnist/

[7] We can test with any distribution other than the uniform distribution as well.

[8] The features and labels for test data are not perturbed.

and also Crowd-ML via (10). With low privacy ($\epsilon^{-1} \to 0$), the performance of both centralized and crowd approaches are almost the same as Fig. 4, and we omit the result. With high privacy ($\epsilon \to 0$), the performance of both approaches degrades to a unusable level. Here we show their performances at $\epsilon^{-1} = 0.1$ in Fig. 5, where the performance is in a transition state between high and low privacy regions. Firstly, the centralized and crowd approaches both perform worse than they did in Fig. 4, which is the price of privacy preservation. Among these results, Crowd-ML with a minibatch size $b = 20$ has the smallest asymptotic error, much below the centralized (batch). Crowd-ML with $b = 1$ and 10 still achieves similar or better asymptotic error compared to Central (batch). As predicted from Section IV, increasing the minibatch size improves the performance of Crowd-ML. When SGD is used for the centralized approach (Central SGD) with perturbed features and labels, its performance is very poor (~ 0.9) regardless of the minibatch size, due to the larger noise required to provide the same level ϵ of privacy as Crowd-ML.

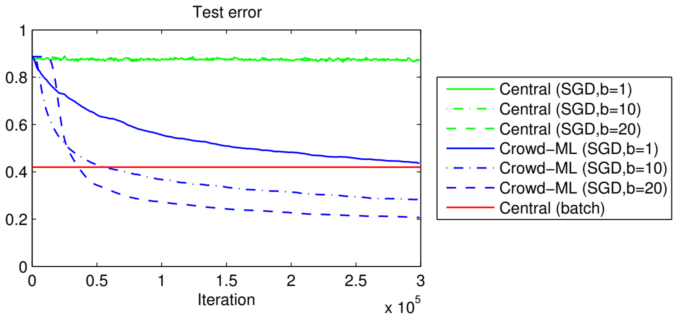

Fig. 5: Comparison of test error for centralized and crowd learning approaches with privacy ($\epsilon^{-1} = 0.1$), varying minibatch sizes (b), and no delay.

Lastly, we look at the impact of delays on Crowd-ML with privacy $\epsilon^{-1} = 0.1$. We test with different delays in the unit of $\Delta = \tau/(MF_s)$, that is, the number of samples generated by all device during the delay of size τ. In Fig. 6, we show the results with two minibatch sizes ($b = 1, 20$) and varying delays ($1\Delta, 10\Delta, 100\Delta, 1000\Delta$). The delay of 1000Δ means that a maximum of 3×1000 samples are generated among the devices, between the time a single device requests a check-out from the server and the time the server received the check-in from that device, which is quite large. Fig. 6 shows that the increase in the delay somewhat slows down the convergence with a minibatch size of 1, and the converged value of error is similar to or worse than Central (batch). However, it also shows that with a minibatch size of 20, delay has little effect on the convergence, and the error is much lower than Central (batch). Note that with the minibatch size of 20, there is a small plateau in the beginning of error curves, reflecting the devices' initial waiting time for their minibatches to be filled before computing begins. After this initial waiting time, the error starts to decrease at a fast rate.

VI. Conclusion

In this paper, we proposed Crowd-ML, a machine learning framework for a crowd of smart devices. Compared to previous crowdsensing systems, Crowd-ML is a framework that integrates sensing, learning, and privacy mechanisms together. Algorithmically, Crowd-ML uses recent advances in distributed and incremental learning and implements strong differentially private mechanisms. We analyzed Crowd-ML to show its advantages over purely centralized or decentralized approaches. We implemented a prototype and evaluated the framework with a simple activity recognition task in a real environment as well as larger-scale experiments in simulated environments that demonstrate the advantages of Crowd-ML's design. Crowd-ML is a general framework for learning with crowdsensing data, and is open to further refinements for specific applications.

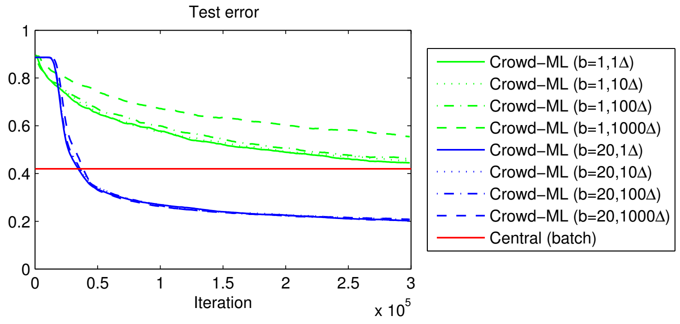

Fig. 6: Impact of delays on Crowd-ML with privacy ($\epsilon^{-1} = 0.1$), varying minibatch sizes, and varying delays.

Appendix

A. Proof of Theorem 1

In our algorithms, a device receives w from the server and sends averaged gradients $\hat{g}$ along with other information. We assume $\|x\|_1 \leq 1$, which can be easily achieved by normalizing the data. The sensitivity of an averaged gradient for logistic regression is $4/b$ as shown below. There are C parameter vectors $w_1, ..., w_C$ for multiclass logistic regression. Let the matrix of gradient vectors corresponding to C parameter vectors be

$$\begin{aligned} g &= [g_1\ g_2\ \cdots\ g_C] = x[P_1\ \cdots\ P_y{-}1\ \cdots\ P_C] \\ &\quad +\lambda[w_1\ \cdots\ w_C] = xM + \lambda[w_1\ \cdots\ w_C], \end{aligned}$$

where $P_j = P(y = j|x; w)$ is the posterior probability, and M is a row vector of P_j's. Without loss of generality, consider two minibatches $\mathcal{D}$ and $\mathcal{D}'$ that differ in only the first sample x_1. The difference of averaged gradients $\tilde{g}(\mathcal{D})$ and $\tilde{g}'(\mathcal{D}')$ is

$$\|\tilde{g} - \tilde{g}'\|_1 \leq \frac{1}{b}(\|x_1 M_1\|_1 + \|x_1' M_1'\|_1) \leq \frac{4}{b},$$

To see $\|M_1\|_1 \leq 2$, note that the absolute sum of the entries of M_1 is $2(1 - P_{y_1}) \leq 2$. The sensitivity of multiple minibatches $\hat{g}(1), ..., \hat{g}(T)$ is the same as the sensitivity of a single $\hat{g}(t)$, and the ϵ-differential privacy follows from Proposition 1 of [3].

B. Proof of Theorem 2

In addition to the averaged gradients, a device sends to the server the numbers of samples n_s, the number of misclassified samples n_e, and the labels counts n_y^k. Perturbation by adding discrete Laplace noise is equivalent to random sampling by the exponential mechanism [46] with $P(\hat{n}_e|n_e) \propto e^{-\frac{\epsilon_e}{2}|\hat{n}_e - n_e|}$, $\hat{n}_e \in \mathbb{Z}$. If two datasets $\mathcal{D}$ and $\mathcal{D}'$ are different in only one item, then the score function $d = -|\hat{n}_e - n_e|$ changes at most by 1. That is, $\max_{\mathcal{D},\mathcal{D}'} \; |d(\hat{n}_e, n_e(\mathcal{D})) - d(\hat{n}_e, n_e(\mathcal{D}'))| = 1$. As with multiple gradients, the sensitivity of multiples sets of $(\hat{n}_e,\, \hat{n}_y^k)$ is the same as the sensitivity of a single set, and ϵ_e-differential privacy follows from Theorem 6 of [46]. Proof of ϵ_{y^k}-differential privacy of n_y^k is similar.

Remark 1: Unlike the gradient $\hat{g}$, the information $(n_s,\, \hat{n}_e,\, \hat{n}_y^k)$ is not required for learning itself, but for monitoring the progress of each device on the server side. Therefore, ϵ_e and ϵ_{y^k} can be set to very small values without affecting the learning performance so that $\epsilon = \epsilon_g + \epsilon_e + C\epsilon_{y^k} \approx \epsilon_g$.

Remark 2: $\hat{n}_e$ and $\hat{n}_y^k$ can be negative with a small probability, which have a limited effect on the estimates of the error rate and the prior at the server. After receiving T minibatches, the error rate and the prior estimates are

$$\mathrm{Err}^{\mathrm{est}} = \frac{\sum_i^T \hat{n}_e(i)}{\sum_i^T n_s(i)} \;\text{ and }\; P^{\mathrm{est}}(y=k) = \frac{\sum_i^T \hat{n}_y^k(i)}{\sum_i^T n_s(i)}. \quad (14)$$

Since $\hat{n}_e(i) - n_e(i)$ is independent for $i = 1, 2, ...$ and has zero-mean and constant variance $\frac{2e^{-\epsilon_e/2}}{(1-e^{-\epsilon_e/2})^2}$ [41], the estimate of error rate converge almost surely to the true error rate with vanishing variances as T increases. The same can be said of the estimate of prior $P(y)$.

C. Differential Privacy in Centralized Approach

For completeness of the paper, we also describe the ϵ-differential privacy mechanisms for the centralized approach. In the centralized approach, data are directly sent to the server. Without a privacy mechanism, an adversary can potentially observe all data. To prevent this, ϵ-differential privacy can be enforced by perturbing the features

$$f(x) = x + z, \;, \;\; P(z) \propto e^{-\frac{\epsilon_x}{2}|z|}, \quad (15)$$

and also perturbing the labels. To perturb labels, we use the exponential mechanism to sample a noisy label $\hat{y}$ given a true label y from

$$P(\hat{y}|y) \propto e^{\frac{\epsilon_y}{2} d(y,\hat{y})}, \;\; y, \hat{y} \in \{1, ..., C\} \quad (16)$$

where we use the score function $d(y, \hat{y}) = I[y = \hat{y}]$.

Theorem 3 (Feature and label perturbation)**.** *The transmission of x and y by feature perturbation (15) and the exponential mechanism (16) is ϵ_x- and ϵ_y-differentially private.*

Proof: Assume $\|x\|_1 \leq 1$. Feature transmission is an identity operation and therefore has sensitivity 2. For label transmission, the score function $d(\hat{y}, y) = I[\hat{y} = y]$ changes at most by 1 by changing y. From Proposition 1 of [3] and Theorem 6 of [46], respectively, we achieve ϵ_x- and ϵ_y-differential privacy of data. ■

Note that the sensitivity is independent of the number of features and labels sent, and that we have to add the same level of independent noise to the features and apply the same amount of label perturbation. An overall ϵ-differential privacy is achieved by $\epsilon = \epsilon_x + \epsilon_y$. The required privacy levels ϵ_x and ϵ_y can be chosen differently, and we use $\epsilon_x = \epsilon_y = \epsilon/2$ in the experiments.

D. Experiments with Visual Object Recognition Task

We repeat the experiments in Section V-C for an object recognition task using CIFAR-10 dataset, which consists of images of 10 types of objects (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck) collected by [47]. We use 50,000 training and 10,000 test images from CIFAR-10. To compute features, we use a convolutional neural network[9] trained using ImageNet ILSVRC2010 dataset[10], which consists of 1.2 million images of 1000 categories. We apply CIFAR-10 images to the network and use the 4096-dimensional output from the last hidden layer of the network as features. Those features are preprocessed with PCA to have a reduced dimension of 100 and are L_1 normalized. We use the same setting in Section V-C to test Crowd-ML on this object recognition task. The results are given in Figs. 7, 8, and 9. The figures are very similar to the handwritten digit recognition task (Figs. 4, 5, 6), except that the error is larger (e.g., 0.3 in Fig. 7) than the error for digit recognition (0.1 in Fig. 4). This is because the CIFAR dataset is more challenging than MNIST due to variations in color, pose, viewpoint, and background of object images.

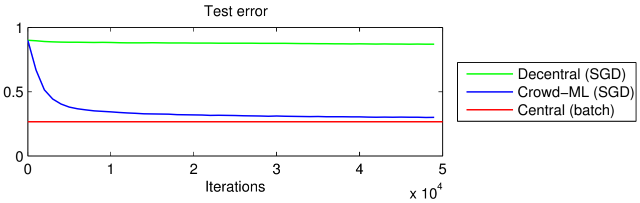

Fig. 7: Comparison of test error for centralized, crowd, and decentralized learning approaches, without delay or privacy consideration.

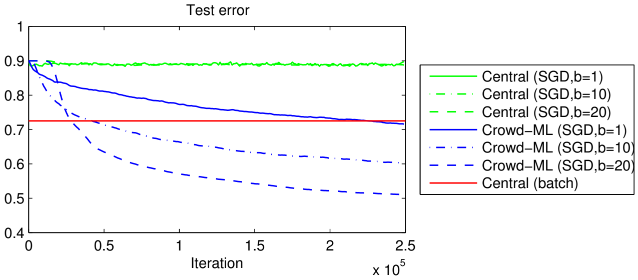

Fig. 8: Comparison of test error for centralized and crowd learning approaches with privacy ($\epsilon^{-1} = 0.1$), varying minibatch sizes (b), and no delay.

References

[1] D. Christin, A. Reinhardt, S. S. Kanhere, and M. Hollick, "A Survey on Privacy in Mobile Participatory Sensing Applications," *J. Syst. Softw.*, vol. 84, pp. 1928–1946, 2011.

[2] C. Dwork and K. Nissim, "Privacy-Preserving Data Mining on Vertically Partitioned Databases," in *Proc. CRYPTO*. Springer, 2004.

[3] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography*. Springer, 2006, pp. 265–284.

[9] https://github.com/jetpacapp/DeepBeliefSDK

[10] http://www.image-net.org/challenges/LSVRC

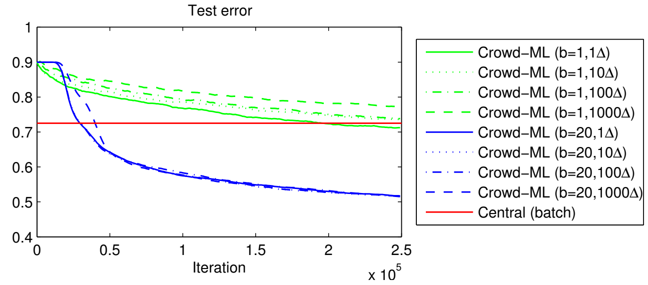

Fig. 9: Impact of delays on Crowd-ML with privacy ($\epsilon^{-1} = 0.1$), varying minibatch sizes, and varying delays.

[4] C. Dwork, “Differential privacy,” in *Automata, languages and programming*. Springer, 2006, pp. 1–12.

[5] V. Vapnik, *The nature of statistical learning theory*. springer, 2000.

[6] H. Robbins and S. Monro, “A stochastic approximation method,” *The annals of mathematical statistics*, pp. 400–407, 1951.

[7] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, “A survey of mobile phone sensing,” *Comm. Mag.*, vol. 48, pp. 140–150, September 2010.

[8] M. Srivastava, T. Abdelzaher, and B. Szymanski, “Human-Centric Sensing,” *Phil. Trans. R. Soc. A*, vol. 370, pp. 176–197, 2012.

[9] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt, “Micro-Blog: Sharing and Querying Content Through Mobile Phones and Social Participation,” in *Proc. ACM MobiSys*, 2008.

[10] R. K. Ganti, N. Pham, Y.-E. Tsai, and T. F. Abdelzaher, “PoolView: Stream Privacy for Grassroots Participatory Sensing,” in *Proc. ACM SenSys*, 2008.

[11] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell, “BikeNet: A Mobile Sensing System for Cyclist Experience Mapping,” *ACM Trans. Sensor Networks*, vol. 6, no. 1, 2009.

[12] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda, “PEIR, the Personal Environmental Impact Report, as a Platform for Participatory Sensing Systems Research,” in *Proc. ACM MobiSys*, 2009.

[13] P. Zhou, Y. Zheng, and M. Li, “How Long to Wait? Predicting Bus Arrival Time with Mobile Phone based Participatory Sensing,” in *Proc. ACM MobiSys*, 2012.

[14] O. D. Lara and M. A. Labrador, “A survey on human activity recognition using wearable sensors,” *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 3, pp. 1192–1209, 2013.

[15] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell, “The Jigsaw Continuous Sensing Engine for Mobile Phone Applications,” in *Proc. ACM SenSys*, 2010.

[16] C.-K. Hsieh, H. Tangmunarunkit, F. Alquaddoomi, J. Jenkins, J. Kang, C. Ketcham, B. Longstaff, J. Selsky, B. Dawson, D. Swendeman, D. Estrin, and N. Ramanathan, “Lifestreams: A Modular Sense-Making Toolset for Identifying Important Patterns from Everyday Life,” in *Proc. ACM SenSys*, 2013.

[17] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, “Soundsense: scalable sound sensing for people-centric applications on mobile phones,” in *Proceedings of the 7th international conference on Mobile systems, applications, and services*. ACM, 2009, pp. 165–178.

[18] A. Parate, M.-C. Chiu, D. Ganesan, and B. M. Marlin, “Leveraging Graphical Models to Improve Accuracy and Reduce Privacy Risks of Mobile Sensing,” in *Proc. ACM MobiSys*, 2013.

[19] S. Nath, “ACE: Exploiting Correlation for Energy-Efficient and Continuous Context Sensing,” in *Proc. ACM MobiSys*, 2012.

[20] A. Agarwal and J. C. Duchi, “Distributed delayed stochastic optimization.” in *Proc. NIPS*, 2011, pp. 873–881.

[21] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao, “Optimal distributed online prediction,” in *Proc. ICML*, 2011.

[22] S. Song, K. Chaudhuri, and A. D. Sarwate, “Stochastic gradient descent with differentially private updates,” in *Proc. IEEE GlobalSIP*, 2013.

[23] J. Krumm, “A survey of computational location privacy,” *Personal and Ubiquitous Computing*, vol. 13, no. 6, pp. 391–399, 2009.

[24] L. Sweeney, “k-Anonymity: A Model for Protecting Privacy,” *Int. J. Uncertainty, Fuzziness, Knowl. Syst.*, vol. 10, no. 5, pp. 557–570, 2002.

[25] A. C. Yao, “Protocols for secure computations,” in *2013 IEEE Symp. Found. Comp. Sci.* IEEE, 1982, pp. 160–164.

[26] B. Fung, K. Wang, R. Chen, and P. S. Yu, “Privacy-preserving data publishing: A survey of recent developments,” *ACM Comp. Surveys (CSUR)*, vol. 42, no. 4, p. 14, 2010.

[27] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith, “Composition attacks and auxiliary information in data privacy,” in *Proc. ACM SIGKDD*. ACM, 2008, pp. 265–273.

[28] F. D. McSherry, “Privacy integrated queries: an extensible platform for privacy-preserving data analysis,” in *Proc. ACM SIGMOD*. ACM, 2009, pp. 19–30.

[29] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, “Differentially private empirical risk minimization,” *JMLR*, vol. 12, pp. 1069–1109, 2011.

[30] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra, and K. Aherer, “Energy-Efficient Continuous Activity Recognition on Mobile Phones,” in *Proc. ISWC*, 2012.

[31] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “A Public Domain Dataset for Human Activity Recognition Using Smartphones,” in *Proc. ESANN*, 2013.

[32] L. Bottou, “Stochastic gradient descent tricks,” in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 421–436.

[33] H. Robbins and S. Monro, “A stochastic approximation method,” *Annal Math. Stat.*, pp. 400–407, 1951.

[34] L. Bottou, “Online learning and stochastic approximations,” *On-line learning in neural networks*, vol. 17, p. 9.

[35] Y. Nesterov, “Primal-dual subgradient methods for convex problems,” *Math. Program.*, vol. 120, no. 1, pp. 221–259, Apr. 2009.

[36] N. L. Roux, M. Schmidt, and F. R. Bach, “A stochastic gradient method with an exponential convergence rate for finite training sets,” in *Proc. NIPS*, 2012, pp. 2663–2671.

[37] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *COLT 2010*, p. 257, 2010.

[38] T. Schaul, S. Zhang, and Y. LeCun, “No more pesky learning rates,” in *Proceedings of The 30th International Conference on Machine Learning*, 2013, pp. 343–351.

[39] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum, “Differential privacy under continual observation,” in *Proceedings of the forty-second ACM symposium on Theory of computing*. ACM, 2010, pp. 715–724.

[40] Ú. Erlingsson, V. Pihur, and A. Korolova, “Rappor: Randomized aggregatable privacy-preserving ordinal response,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 1054–1067.

[41] S. Inusah and T. J. Kozubowski, “A Discrete Analogue of the Laplace distribution,” *J. Stat. Plan. Inf.*, vol. 136, no. 3, pp. 1090–1102, 2006.

[42] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro, “Robust stochastic approximation approach to stochastic programming,” *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1574–1609, 2009.

[43] O. Shamir and T. Zhang, “Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes,” in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, S. Dasgupta and D. Mcallester, Eds., vol. 28, no. 1, 2013, pp. 71–79.

[44] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan, “Better mini-batch algorithms via accelerated gradient methods,” in *Advances in Neural Information Processing Systems*, 2011, pp. 1647–1655.

[45] M. Anthony and P. L. Bartlett, *Neural network learning: Theoretical foundations*. Cambridge University Press, 1999.

[46] F. McSherry and K. Talwar, “Mechanism design via differential privacy,” in *Proc. IEEE FOCS*, 2007.

[47] A. Krizhevsky, “Learning multiple layers of features from tiny images,” Tech. Rep., 2009.