Private, yet Practical, Multiparty Deep Learning

Xinyang Zhang
Lehigh University
xizc15@lehigh.edu

Shouling Ji
Zhejiang University
sji@zju.edu.cn

Hui Wang
Stevens Institute of Technology
hui.wang@stevens.edu

Ting Wang
Lehigh University
inbox.ting@gmail.com

***Abstract*—In this paper, we consider the problem of multiparty deep learning (MDL), wherein autonomous data owners jointly train accurate deep neural network models without sharing their private data. We design, implement, and evaluate $\propto$MDL, a new MDL paradigm built upon three primitives: asynchronous optimization, lightweight homomorphic encryption, and threshold secret sharing. Compared with prior work, $\propto$MDL departs in significant ways: a) besides providing explicit privacy guarantee, it retains desirable model utility, which is paramount for accuracy-critical domains; b) it provides an intuitive handle for the operator to gracefully balance model utility and training efficiency; c) moreover, it supports delicate control over communication and computational costs by offering two variants, $\propto$MDL^{c} and $\propto$MDL^{d}, operating under loose and tight coordination respectively, thus optimizable for given system settings (e.g., limited versus sufficient network bandwidth). Through extensive empirical evaluation using benchmark datasets and deep learning architectures, we demonstrate the efficacy of $\propto$MDL.**

I. Introduction

The recent advances in deep learning (DL) [1] have led to breakthroughs in long-standing artificial intelligence tasks (e.g., image recognition, language translation, and even playing Go [2]), enabling use cases previously considered strictly experimental. Such success is premised on the availability of massive training data. Most deployed DL systems are built upon large, centralized data repositories. This paradigm abounds with privacy risks: often, data owners neither understand nor have control over how their private data is being used. Moreover, in a range of domains, notably those related to healthcare, the sharing of personal information is forbidden by regulations. Thus, following the centralized-training paradigm, clinical sites and institutions can only perform analysis and modeling over their own data, resulting in largely suboptimal modeling and decision-making. In these cases, privacy and confidentiality restrictions significantly impede the full exploitation of data owned by these parties.

There thus emerges a critical need for multiparty deep learning (MDL), wherein autonomous data owners jointly train deep neural network (DNN) models without sharing their private data yet benefitting from others' data. Existing MDL solutions [3], [4], [5] follow the model-sharing paradigm: each party trains on its private data and shares its local model. The global model is built by aggregating local models. As local models may be reverse-engineered to reveal private data of contributing parties [3], an oft-used remedy is to apply differentially private randomization [6] to local models. However, as each party operates under the local privacy context [7], the injected randomization is often overly conservative, resulting in significant utility loss in the global model.

Motivated by the observations above, we present $\propto$MDL, a new MDL paradigm that, besides providing strong privacy guarantee, retains desirable model utility. At a high level, $\propto$MDL ensures that participating parties learn the global model (necessary for MDL to function) only if a sufficient number of local models are aggregated, while no extra information pertaining to individual local models is leaked in this process. We show both empirically and analytically that this design not only adds a new layer of privacy protection, but also significantly improves the utility of the global model.

A series of unique features distinguish $\propto$MDL from prior art. First, it achieves explicit privacy assurance and desirable model utility simultaneously, which is paramount for accuracy-critical domains (e.g., healthcare predictive modeling). Further, it provides an intuitive handle for the operators to gracefully balance model utility and training efficiency. Moreover, $\propto$MDL supports delicate control over communication and computational costs. In specific, it offers two variants, coordinated- and dynamic-$\propto$MDL, operating under tight and loose coordination respectively. Therefore, the operators may choose the variant optimized for given system settings (e.g., limited versus sufficient network bandwidth).

We empirically evaluate $\propto$MDL on varied benchmark datasets and DNN architectures. In all the cases, $\propto$MDL achieves model utility close to the standalone, privacy-violating training, wherein a single party trains on the entire dataset. For instance, over the SVHN dataset, $\propto$MDL obtains 91.4% accuracy (92.1% accuracy by the standalone training).

The remainder of the paper proceeds as follows. Section II introduces the building blocks of $\propto$MDL; Section III presents the high-level design of $\propto$MDL; Section IV and V elaborate the two variants of $\propto$MDL, followed by their empirical evaluation in Section VI; Section VII surveys relevant literature; the paper is concluded in Section VIII.

II. Preliminaries

A. Deep Learning

Deep learning (DL) represents a class of machine learning algorithms which learn high-level abstraction of complex data using multiple processing layers and non-linear transformations. We primarily focus on supervised learning, wherein the training inputs are associated with "labels" while the goal is to learn models to predict the labels for future inputs. Our discussion is applicable to unsupervised learning as well.

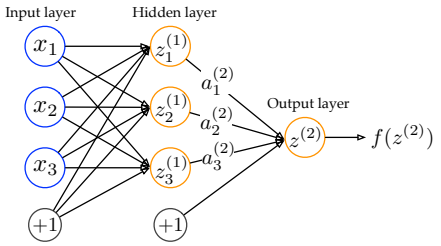

Figure 1: A schematic example of neural network.

Training a DNN model (see Figure 1) is to optimally configure its parameters. Due to the model complexity, stochastic gradient descent and its variants are often used [8]. During each "epoch", a set of "mini-batches" are sampled from the training set. The gradient of each parameter w with respect to the objective function (e.g., the cross entropy of the training labels and the model's outputs) is computed for every mini-batch via a back-propagation procedure. The update rule for w is $w := w - \lambda \nabla_w$, where λ is the learning rate and ∇_w is the gradient computed from the current mini-batch. This process repeats until the objective function converges.

This procedure can be generalized to the case of multiple parties: each party performs training using its own data; at the end of each epoch, they share with each other their "local" gradients; the local gradients of a parameter w are aggregated to determine its global descent. The update rule for w is:

$$w := w - \lambda \frac{\sum_{A_i \in \mathcal{A}} \nabla_w^i}{|\mathcal{A}|} \tag{1}$$

where $\mathcal{A}$ denotes the set of parties, A_i is the i-th party, and ∇_w^i is w's gradient computed using the data owned by A_i.

Despite its efficiency [8], [9], this construction is potentially privacy-violating: the local gradients are exploitable to reveal sensitive information of contributing parties [3], [4]. We thus consider local gradients as private information to be protected.

B. Attack Model

Like most other privacy-preserving machine learning frameworks (e.g., [10], [11], [12], [5]), we assume a semi-honest attack model: each party strictly follows the predefined protocol but is curious about the private information of other parties; further, different parties may collude with each other, attempting to infer a victim's private information.

We remark that this threat model is realistic. Theoretically, it is impossible to prevent malicious parties from invading on a victim's privacy at the cost of sabotaging the tasks. For instance, all the malicious parties may empty their training data to expose the victim's private information in the training outcome. Practically, in healthcare domains, clinical sites are strongly incentivized to collaboratively train predictive models with better demographical coverage and diagnosis accuracy than models trained on their own data; yet, they may also be interested to learn other parities' data ownerships.

C. Cryptographic Building Blocks

Below we introduce two cryptographic primitives that serve as the building blocks of $\propto$MDL.

a) Threshold Secret Sharing: Secret sharing represents a set of cryptographic primitives that split and distribute a secret amongst multiple parties, each allocated a share. The secret is reconstructable only if a sufficient number of shares (e.g., m out of n) are combined together, while individual shares are of no use on their own.

We use the Shamir's secret sharing protocol [13] as a concrete instance. This protocol is built upon the polynomial interpolation construction. Denote by $\theta_0, \theta_1, \ldots, \theta_{m-1}$ the integer coefficients of a degree-$(m-1)$ polynomial f, i.e., $f(x) = \theta_0 + \theta_1 x + \ldots + \theta_{m-1} x^{m-1}$. With θ_0 being the secret, it can be reconstructed as: $\theta_0 = \sum_{i=0}^{m-1} f(x_i) \prod_{0 \le j \le m-1, j \ne i} \frac{-x_j}{x_i - x_j}$, if any m distinct points: $\{(x_i, f(x_i))\}_{i=0}^{m-1}$ are collected; with less than m such points, it is impossible to compute θ_0.

b) Homomorphic Encryption: Homomorphic encryption allows the computation to be performed on the ciphertext and generate an encrypted result which, when decrypted, matches the result of the computation performed over the plaintext. In the following, we use the ElGamal cryptosystem [14] as a concrete instance, which supports multiplicative homomorphism: $\mathsf{Enc}(m) \cdot \mathsf{Enc}(m') = \mathsf{Enc}(m \cdot m')$, where $\mathsf{Enc}(\cdot)$ denotes the encryption algorithm, $\cdot$ represents multiplication, and m, m' are two plaintext messages.

Unlike fully homomorphic encryption [15] that supports homomorphism for arbitrary functions, the cryptosystem above only supports multiplicative homomorphism via modular exponentiation, thereby being practically efficient.

Also note that the cryptographic primitives above operate in the integer space, while DL systems use floating numbers. A simple solution is to multiply each number by a constant (e.g. 252 for IEEE 754 doubles) to support finite precision. Fortunately, in most DL systems, due to regularization, parameters are usually small (e.g., at the magnitude of 10^{-3}). The type transformation thus incurs fairly limited accuracy loss.

III. Overview of $\propto$MDL

A. System Architecture

To build $\propto$MDL, we begin with the strawman construction in Section II, which supports MDL, albeit in a privacy-violating manner. We then overcome this limitation by building proper privacy enhancing mechanisms (PEMs) into this construction. Below we refer to each participating party as a "worker".

At the core of this construction lies in the protocol of sharing local gradients among the workers. Possible design options include: the workers exchange local gradients directly, via a centralized server, or by using secure multiparty computation (SMC) primitives [16], [17] in an oblivious manner. All these protocols can be unified by the abstraction of a virtual coordinator (which we refer to as the "manager"), that aggregates the local gradients uploaded by the workers and distributes the computed global gradients back to them.

As shown in Figure 2, each worker A maintains a local model while the manager M maintains a global model. During the training, A trains its local model using the SGD algorithm using its private data and influences others' training indirectly

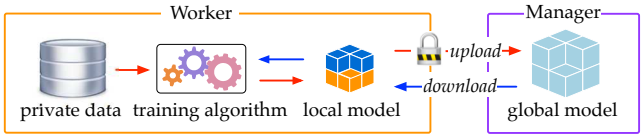

Figure 2: Illustration of multiparty deep learning architecture.

via the global model at M. It is noted that once the training is complete (i.e., the global model converges), each worker can independently and privately evaluate the trained model over new data, without interacting with other workers.

B. Basic Design

Now we incorporate proper PEMs into the basic construction. One straightforward solution is to perform differentially private randomization [6] to the local gradients, with the hope that releasing such local gradients does not overly leak the private information of the local data [18], [5], which we refer to as the local differential privacy (LDP) solution.

Intuitively, a function f is differentially private if its probability of producing a particular output is not sensitive to whether a specific data entry is included in its input: given any two datasets $\mathcal{D}$, $\mathcal{D}'$ differing by a single entry and any output $\mathcal{O}$, it holds that $\Pr(f(\mathcal{D}) \in \mathcal{O}) \leq \exp(\epsilon) \cdot \Pr(f(\mathcal{D}') \in \mathcal{O})$, where ϵ is the "budget" controlling the tolerable privacy loss. We may specify such budget for each parameter and randomize its local gradient using the Laplacian or Gaussian mechanisms [6].

Despite its simplicity, this mechanism suffers low model utility. Specifically, applying DP requires estimating the *global sensitivity*: how the randomization of the input impacts the global output. However, in MDL, due to lacking the knowledge about others' data, it is extremely obscure for a worker to estimate the influence of its input to the global model. Therefore, overly conservative noise needs to be injected, resulting in significant utility loss in the global model. Such utility loss is consequential for accuracy-demanding domains [19].

C. Enhanced Design

To mitigate this issue, we integrate the LDP mechanism with a coordination mechanism. Our design is motivated by the following observations. Most DP mechanisms employ random noise sampled from symmetric distributions (e.g., Laplace and Gaussian); as more workers aggregate their local gradients, a larger part of the injected noise can cancel out. Formally,

Theorem 1. *Let ∇^i denote the local gradient of the i-th worker and r^i be the differentially private noise: $r^i \sim \mathsf{Lap}(\frac{1}{\epsilon})$. The estimated global gradient $\frac{1}{|\mathcal{A}|}\sum_{A_i \in \mathcal{A}}(\nabla^i + r^i)$ is a random variable with mean $\frac{1}{|\mathcal{A}|}\sum_{A_i \in \mathcal{A}} \nabla^i$ and variance $\frac{2}{\epsilon^2 \cdot |\mathcal{A}|}$.*

Proof. It follows from the definition of global gradient in Eqn. (1) and the independence of random noise $\{r^i\}_{A_i \in \mathcal{A}}$. □

Intuitively, as more workers aggregate local gradients, we obtain better estimates of global gradients. We formalize this intuition with the concept of *ρ-visibility*: an n-worker MDL system is ρ-visible, if the global gradient of each parameter is visible (available for download) only after it aggregates the local gradients of at least $m = \rho \cdot n$ workers. The subset of workers contributing to parameter w is called the active set of w, denoted by $\mathcal{A}_w$. Clearly, ρ-visibility enforces $|\mathcal{A}_w| \geq \rho \cdot |\mathcal{A}|$ for any w at any time.

We may consider ρ-visibility as another layer of privacy protection, because it essentially enforces a generalized abstraction of secure aggregation [20]: the global information is revealed only when it has included the local information from a sufficient number of individuals. It is also noted that ρ-visibility does not compromise the protection offered by the LDP mechanism, as it is applied in the post-processing stage of LDP. Therefore, they can be integrated in synergy, which we name as the $\propto$MDL mechanism.

Next we discuss a few instantiations of ρ-visibility to show the tradeoff between model utility and training efficiency.

D. Implications of Rho-Visibility

As more workers aggregate local gradients, we obtain more accurate global gradients, leading to better utility of the global model. However, this improvement is not free: the synchronization among more workers at every epoch implies higher computational and communication costs per epoch. Given the assumption that the global model converges roughly within a fixed number of epochs (which is empirically validated in Section VI), more synchronization results in higher overall training cost. Thus, there exists inherent tradeoff between model utility and training efficiency, as shown in Figure 3.

Concretely, $\rho = 1$ corresponds to a synchronous protocol. After each training epoch, the workers contribute their local gradients to estimating the global gradient. The global gradient is computed in a privacy-preserving manner, which ensures that each worker only learns the global gradient without knowing the local gradients of other workers.

Meanwhile, $\rho = \frac{1}{n}$ entails an asynchronous protocol [5], wherein the workers perform the training in a collaborative yet uncoordinated manner. In specific, after finishing an epoch, each worker asynchronously uploads its local gradient to the manager, downloads the gradients contributed by other workers, and updates its local model.

Finally, $\frac{1}{n} < \rho < 1$ corresponds to a hybrid protocol, which features better model utility than asynchronous protocols and lower training cost than synchronous protocols. Therefore, the operator is able to balance model utility and training efficiency by adjusting ρ.

E. $\propto$MDL: A Nutshell View

At the core of $\propto$MDL is a novel use of lightweight homomorphic encryption and threshold secret sharing to construct a gradient exchange protocol. In a nutshell, each worker A_i is assigned a private key sk_i and all the workers share a public key pk. For each parameter w, A_i encrypts its local gradient ∇_w^i as $\mathsf{Enc}_{\mathsf{pk}}\left[\exp\left(\nabla_w^i\right)\right]$. The multiplicative homomorphism entails: $\prod_i \mathsf{Enc}_{\mathsf{pk}}\left[\exp\left(\nabla_w^i\right)\right] = \mathsf{Enc}_{\mathsf{pk}}\left[\exp\left(\sum_i \nabla_w^i\right)\right]$. The aggregated ciphertext thus encodes the summation of local gradients of active workers, which is equivalent to the global

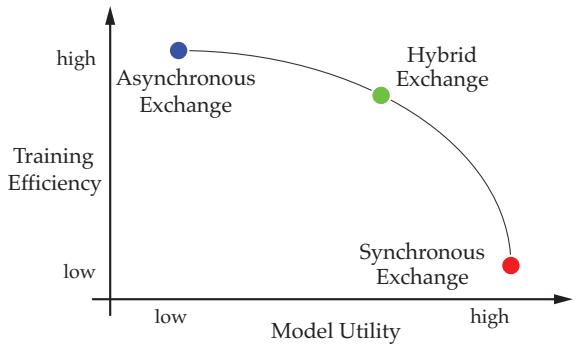

Figure 3: Tradeoff of model utility and training efficiency.

gradient. If at least m ($m = \rho \cdot n$) workers have contributed local gradients, the global gradient is automatically decryptable, thereby available for all the workers to access; otherwise, the global gradient is invisible.

Although existing cryptosystems (e.g. [21], [22], [23]) also meet the requirement that more than a threshold number of workers are required to decrypt the ciphertext, our setting differs in significant ways. First, existing schemes implicitly assume that the set of active workers are known in advance. In MDL setting, the active workers are determined on the fly; each worker may contribute to different sets of parameters during different epochs. Second, existing schemes require multiple rounds of communication among the active workers, which is costly for the setting of a large number of parameters and active workers. Finally, exiting schemes are designed to encrypt and decrypt a fixed secret message, while in our target setting, the secret message (i.e., the summation of local gradients) is dynamically constructed by the active workers.

Next we detail two variants of $\propto$MDL operating under tight and loose coordination respectively. The symbols used in the following are summarized in Table I.

IV. $\propto$MDL$^{\mathrm{c}}$

A. Overview

In $\propto$MDL$^{\mathrm{c}}$, the workers are shepherded by the manager to perform training in a semi-synchronous manner. Specifically, $\propto$MDL$^{\mathrm{c}}$ involves the following operations (without loss of generality, we consider a specific parameter w)

- `Setup` - A trusted authority T (e.g., certificate authority) generates cryptographic keys, announces the public keys, and distributes the private keys to the workers.
- `Training` - Each worker A_i $(i = 1, 2, \ldots, n)$ independently trains the model over its private data and derives the local gradient ∇_w^i for each parameter w.
- `Register` - A_i selects a set of parameters $\mathcal{R}_i$ to contribute, and registers its requests with the manager M.
- `Callback` - If at least m workers register for w, M invokes them to upload their local gradients; meanwhile, once the global model is updated, M notifies these workers to download the latest value of w (i.e., w^g).
- `Upload` - Invoked by M, A_i uploads its local gradient ∇_w^i to M in a privacy-preserving manner.

Symbol	Definition
n	total number of workers in MDL$^{\propto}$
ρ	visibility threshold
m	minimum size of active set $\rho \cdot n$
t	total number of parameters in DNN
k	number of parameters registered by each worker
k^*	minimum k required for MDL$^{\propto}$ to function
$\mathcal{A}_w$	active set of workers with respect to w
∇_w^i	local gradient of w by worker A_i
$\mathcal{R}_i$	set of parameters registered by A_i
c_i	ciphertext of ∇_w^i
z_i	Lagrange coefficient of A_i: $z_i = \prod_{A_j \in \mathcal{A}_w, j \neq i} \frac{-x_j}{x_i - x_j}$
∇_w^g	global gradient of w
r_i	random nonce of A_i
r	sum of random nonce $\sum_{A_i \in \mathcal{A}_w} r_i$

Table I. Symbols and notations.

- `Download` - Notified by M, A_i downloads the global version of w (w^g) and applies it to its local model.

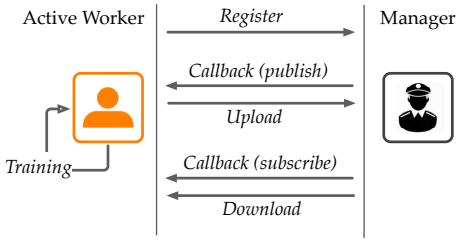

Figure 4: Protocols of worker-manager interaction.

Note that the `Setup` operation only executes once, since all keys are reusable in the following stages. The other operations are performed during each epoch, constituting the worker-manager interactions, as shown in Figure 4. Also note that we assume a simple contribution-based download policy, that is, each worker is only allowed to download the global parameters which it contributes to. We consider developing more advanced download policies as our ongoing research.

Algorithm 1: Trusted Authority

Input: generator g, primes p, q

`// generate keys`

1 $s \xleftarrow{\$} \mathbb{Z}_p$;
2 public key $h \leftarrow g^s$;
3 $\{(x_1, s_1)\}_{i=1}^n \leftarrow$ `Shamir's Protocol` with s as secret;

`// distribute keys`

4 announce h and $\{(x_i, g^{s_i})\}_{i=1}^n$;
5 **for** $i = 1, 2, \ldots, n$ **do** assign s_i to A_i;

B. Protocols

Below we elaborate the operations from the perspective of each key player of $\propto$MDL$^{\mathrm{c}}$.

a) Trusted Authority: The trusted authority T generates cryptographic keys. In specific, let p, q be two large primes satisfying $p = 2kq + 1$ for some constant integer k. With a random number $s \in \mathbb{Z}_p$ being the secret, each worker A_i is assigned a share (x_i, s_i) generated by the Shamir's protocol, where x_i is public while s_i is private to A_i. To set up the ElGamal cryptosystem, let $\mathbb{G}$ be a cyclic group of order q (i.e., a subgroup of $\mathbb{Z}_p^*$) with g as its generator; then $h = g^s$ will be the public key. In order not to clutter the notations, we omit the modulo operation in the following presentation.

Algorithm 2: Manager M

Input: public key h, generator g, public shares $\{g^{s_i}, x_i\}_{i=1}^n$

/* initialization */
1 initialize global parameters;
2 **while** *approximate optimum is not achieved yet* **do**
/* request (register) */
3 **event** *register by* A_i *to contribute to* w
4 add A_i to $\mathcal{A}_w$;
/* callback (upload) */
5 **event** *callback* $\mathcal{A}_w$ *to upload* w
6 receive c_i, g^{r_i} from A_i;
// (i) $z_i \triangleq \prod_{A_j \in \mathcal{A}_w, j \neq i} \frac{-x_j}{x_i - x_j}$ (ii) $r \triangleq \sum_{A_i \in \mathcal{A}_w} r_i$
7 **for** $A_i \in \mathcal{A}_w$ **do**
8 send $g^{z_i r}$ to A_i;
9 receive $d_i \leftarrow (g^{z_i r})^{s_i}$;
// decrypt global gradient
10 compute $\nabla_w^g = \frac{1}{|\mathcal{A}_w|} \ln \left(\frac{\prod_{A_i \in \mathcal{A}_w} c_i}{\prod_{A_i \in \mathcal{A}_w} d_i} \right)$;
// update global parameter
11 update $w^g \leftarrow w^g - \lambda \nabla_w^g$;
/* callback (download) */
12 **event** *callback* $\mathcal{A}_w$ *to download* w
13 **for** $A_i \in \mathcal{A}_w$ **do** send w^g to A_i;

Algorithm 3: Worker A_i

Input: secret share s_i, primes p, q

/* initialization */
1 initialize local parameters;
2 **while** *approximate optimum is not achieved yet* **do**
/* A_i registers to contribute to w */
3 register requests $\mathcal{R}_i$ with M;
/* training */
4 run SGD on local dataset to compute $\{\nabla_w^i\}_w$;
5 **for** $w \notin \mathcal{R}_i$ **do** update $w^i \leftarrow w^i - \lambda \nabla_w^i$;
6 **while** $\mathcal{R}_i \neq \emptyset$ **do**
/* upload */
7 **event** *callback by* M *to upload* w
// local gradient, one-time nonce
8 $c_i \leftarrow \exp(\nabla_w^i) \cdot h^{r_i}$; $r_i \xleftarrow{\$} \mathbb{Z}_q$;
9 send c_i, g^{r_i} to M;
// auxiliary information
10 receive $g^{z_i r}$ from M;
11 compute and send $(g^{z_i r})^{s_i}$ to M;
/* download */
12 **event** *callback by* M *to download* w
13 receive w^g and update $w^i \leftarrow w^g$;
14 remove w from $\mathcal{R}_i$;

b) Manager: After initializing the global model (line 1), M handles all requests and callbacks (line 2-13). It manages an active list $\mathcal{A}_w$ for w, which records the workers registered to contribute to w. On receiving the register request by A_i for w, M adds A_i to $\mathcal{A}_w$ (line 4). Once all the workers finish registration, M calls back the workers in $\mathcal{A}_w$ and receives from them encrypted local gradients, one-time nonces and auxiliary information to update w (line 6-9). In the event that w is ready for download, M notifies the workers in $\mathcal{A}_w$ to receive the latest value of w^g (line 13).

c) Worker: After initializing its local model, A_i enters the training loop. During each training epoch, A_i first decides the parameters to which it intends to contribute ($\mathcal{R}_i$) and registers $\mathcal{R}_i$ with M (line 3). We will discuss the selection of $\mathcal{R}_i$ shortly. Then, A_i runs SGD over its private data and updates local parameters (line 4-5). Next A_i enters an event-driven loop (line 6-14). In the event of callback by M to update w, A_i uploads the encrypted local gradient c_i, an encrypted one-time nonce r_i and auxiliary information to M (line 8-11); in the event of callback to download w, A_i updates w with its global version (line 13-14).

C. Analysis

Next we analyze the correctness, privacy and complexity properties of $\propto$MDL^c.

a) Correctness: We show that during each epoch, the global gradient of parameter w can be correctly decoded, if at least m workers contribute to it.

Theorem 2. *If $|\mathcal{A}_w| \geq m$, then the global gradient of w is correctly decryptable as $\frac{1}{|\mathcal{A}_w|} \sum_{A_i \in \mathcal{A}_w} \nabla_w^i$.*

Proof. From the definitions of encryption procedure and multiplicative homomorphism, we have

$$\prod_{A_i \in \mathcal{A}_w} c_i = g^{sr} \exp\left(\sum_{A_i \in \mathcal{A}_w} \nabla_w^i \right)$$

where $r = \sum_{A_i \in \mathcal{A}_w} r_i$.

Then for $|\mathcal{A}_w| \geq m$, based on the property of Shamir's protocol, we have

$$\prod_{A_i \in \mathcal{A}_w} d_i = \prod_{A_i \in \mathcal{A}_w} g^{s_i z_i r} = g^{r \sum_{A_i \in \mathcal{A}_w} s_i z_i} = g^{sr}$$

Thus, $\nabla_w^g = \frac{1}{|\mathcal{A}_w|} \ln \left(\frac{\prod_{A_i \in \mathcal{A}_w} c_i}{\prod_{A_i \in \mathcal{A}_w} d_i} \right) = \frac{\sum_{A_i \in \mathcal{A}_w} \nabla_w^i}{|\mathcal{A}_w|}$. □

Further, we show the overall correctness of the protocols above. Particularly, we only need to prove:

Theorem 3. *During any training epoch, the parameter w is correctly updated by the worker-manager interaction.*

Proof. (Sketch) During each epoch, from the perspective of either participating worker A_i or manager M, the protocol of updating w is strictly serial, as shown in Figure 4, consisting of `register`, `callback` (upload), `upload`, `callback` (download) and `download`. This serialism entails that A_i and M always send and receive the correct version of w. □

In current implementation, the worker A_i randomly selects k parameters as $\mathcal{R}_i$. It can derived that the probability of a given parameter missing updates (i.e., less than m workers register for it) in an epoch is at most $\binom{n}{n-m+1}(\frac{t-k}{t})^{n-m+1}$, where t is the total number of parameters. In realistic settings, a small k often suffices: for example, with $n = 8$ and $m = 2$, $k = 0.47 \cdot t$ ensures that the missing rate is below 10%. Also note that the minimum k (k^*) necessary for each parameter to be updated is $\lceil m \cdot t/n \rceil$.

b) Privacy: We desire for two types of privacy assurance for global and local gradients respectively.

- Global gradient - If less than m workers have contributions to w, its global gradient is undecryptable.
- Local gradient - The gradient of an individual worker is invisible to other parties under any circumstances.

Theorem 4. *If $|\mathcal{A}_w| < m$, the global gradient ∇_w^g of w is undecryptable.*

Proof. (Sketch) We use the concepts of zero-knowledge and simulatability [24]. The system involves the following players: $\mathcal{A}_w$, the manager M and the rest workers (collectively denoted by $\mathcal{B}_w$). Given $\{\nabla_w^i, c_i, d_i, g^{r_i}\}_{A_i \in \mathcal{A}_w}$, it is straightforward to see that the interaction between $\mathcal{A}_w$ and $\mathcal{B}_w$ can be simulated; more precisely, the probability distribution of the views is simulatable. Thus, no extra information is revealed about $\{s_i, r_i\}_{A_i \in \mathcal{A}_w}$ other than $\{c_i, d_i, g^{r_i}\}_{A_i \in \mathcal{A}_w}$.

To reveal ∇_w^g from $\{c_i\}_{A_i \in \mathcal{A}_w}$, the attacker needs to solve g^{sr}. Given the accessible information, this is equivalent to (i) computing g^{sr} from g^s and g^r (guarded by the hardness of computational Diffie-Hellman problem), (ii) computing g^{sr} from $\{d_i\}_{A_i \in \mathcal{A}_w}$ for $|\mathcal{A}_w| < m$ (guarded by the Shamir's protocol), or (iii) computing s (or r) from g^s (or g^r) (guarded by the hardness of discrete logarithm). In other words, an attacker that reveals ∇_w^g implies an attacker that can efficiently solve one of the problems above. □

Further, we show the privacy guarantee for local gradients. We consider the worst case that all other workers collude with each other, attempting to reveal the local gradient of a victim.

Theorem 5. *If $|\mathcal{A}_w| < m$, the local gradient of an individual worker in $\mathcal{A}_w$ is undecryptable, even if all other workers in $\mathcal{A}_w$ collude with each other.*

Proof. (Sketch) We again use the concept of zero-knowledge proof. Without loss of generality, let A_i be the victim and $\mathcal{A}'_w$ be the remaining parties in the system. Given (∇_w^i, c_i, d_i), it is trivial to see that the interaction between A_i and $\mathcal{A}'_w$ can be simulated. No extra information is leaked about (s_i, r_i) other than (c_i, d_i). Thus, if the computational Diffie-Hellman problem is hard, the computation of ∇_w^i is also hard. □

c) Complexity: Since the local training often dominates each worker's computation load (see our empirical evaluation), we focus our analysis on the communication complexity. In particular, we consider the case that each worker registers for the minimum number of parameters ($k^* = \lceil m \cdot t/n \rceil$). In Algorithm 3, during each epoch, for each $w \in \mathcal{R}_i$, A_i exchanges a constant number of integers with M, which entails the overall communication cost of $O(\lceil m \cdot t/n \rceil)$. Correspondingly, in Algorithm 2, the communication cost of M is the aggregated costs of all workers, i.e., $O(t \cdot m)$.

V. $\propto$MDL$^{\text{D}}$

A. Overview

At a high level, $\propto$MDL$^{\text{d}}$ departs from $\propto$MDL$^{\text{c}}$ in that it requires little coordination among the participating workers, thereby enabling high concurrency. In specific, for given parameter w, $\propto$MDL$^{\text{c}}$ requires fixing the set of active workers $\mathcal{A}_w$ (i.e., the `regsiter` operation) before allowing them to upload the local gradients of w. In contrast, in $\propto$MDL$^{\text{d}}$, $\mathcal{A}_w$ is constructed on the fly; the workers contribute to w on a voluntary and first-come-first-served basis.

In $\propto$MDL$^{\text{c}}$, M updates w only if all the workers in $\mathcal{A}_w$ have committed their local gradients. This restriction results in costly delay in each epoch in cases of (i) large-size $\mathcal{A}_w$ or (ii) the workers with imbalanced computation capacities. Also it may cause the robustness issue, as the malfunction of a single worker in $\mathcal{A}_w$ causes the failure of updating w. In $\propto$MDL$^{\text{d}}$, once any m ($m = \rho \cdot n$) workers have uploaded their local gradients of w, M is able to update w immediately. This difference entails $\propto$MDL$^{\text{d}}$'s significant advantage over $\propto$MDL$^{\text{c}}$ in execution efficiency and fault tolerance.

Algorithm 4: Manager M

Input: public key h, generator g, public shares $\{g^{s_i}, x_i\}_{i=1}^n$

```
/* initialization {...} */
1  while approximate optimum is not achieved yet do
       /* request (upload) */
2      event A_i requests to contribute to w
           // c-value
3          receive c_i and g^{r_i} from A_i;
           // z- and d-values
4          for A_j ∈ A_w do
5              update z_j ← z_j · (−x_i)/(x_j − x_i);
6              send g^{s_j z_j} to A_i and receive g^{s_j z_j r_i};
7              update d_j ← (d_j)^{(−x_i)/(x_j − x_i)} · g^{s_j z_j r_i};
8          add A_i to A_w;
9          assign z_i ← ∏_{A_j ∈ A_w, j ≠ i} (−x_j)/(x_i − x_j);
10         send g^{z_i r} to A_i, receive g^{s_i z_i r} and assign d_i ← g^{s_i z_i r};
11         if |A_w| ≥ m then
               // decrypt global gradient
12             compute ∇_w^g = 1/|A_w| ln( ∏_{A_j ∈ A_w} c_j / ∏_{A_j ∈ A_w} d_j );
               // update global parameter
13             update w^g ← w^g − λ∇_w^g;
       /* callback (download) {...} */
```

B. Protocols

Next we detail the protocols of $\propto$MDL$^{\text{d}}$. For space limitations, we omit the operations identical to $\propto$MDL$^{\text{c}}$.

a) Manager: We focus on the operation of `request (upload)`, with which M handles the requests by active workers to upload their local gradients.

In specific, $\mathcal{A}_w$ (for parameter w) maintains the current set of active workers which have uploaded their encrypted local gradients (c-values) of w. Meanwhile, M maintains two values (z- and d-values) for each worker in $\mathcal{A}_w$. As a new worker A_i comes, besides receiving its local gradient and one-time nonce (line 3), M interacts with A_i and updates z- and d-values for each worker (including A_i) in $\mathcal{A}_w$ (line 4-10). Once m workers have uploaded their local gradients, the global gradient ∇_w^g is immediately decryptable by aggregating c- and d-values of workers in $\mathcal{A}_w$ (line 12). After update according to Eqn.(1), w^g is available for workers in $\mathcal{A}_w$ to download (line 13).

b) Worker: We focus on the operation of `upload`, with which the worker A_i uploads its local gradient to M. Specifically, A_i first encrypts its local gradient and sends the ciphertext (c-value) and one-time nonce to M (line 3). Then A_i interacts with M and helps update the z- and d-values corresponding to the workers (including itself) in the current active list $\mathcal{A}_w$ (line 4-7).

Algorithm 5: Worker A_i

Input: secret share s_i, primes p, q

/* initialization {...} */

while *approximate optimum is not achieved yet* **do**

  /* training {...} */

  /* upload */

  for $w \in \mathcal{R}_i$ **do**

    // c-value, one-time nonce

    $c_i \leftarrow \exp(\nabla_w^i) \cdot h^{r_i}$; $r_i \xleftarrow{\$} \mathbb{Z}_q$; send c_i, g^{r_i} to M;

    // d- and z-values

    receive $\{g^{s_j z_j}\}_{A_j \in \mathcal{A}_w}$ from M;

    compute and send $\{g^{s_j z_j r_i}\}_{A_j \in \mathcal{A}_w}$ to M;

    receive $g^{z_i r}$ from M;

    compute and send $g^{s_i z_i r}$ to M;

  remove from $\mathcal{R}_i$ parameters A_i has no contributes to;

  while $\mathcal{R}_i \neq \emptyset$ **do**

    /* download {...} */

C. Analysis

We now analyze the correctness, privacy and complexity properties of $\propto$MDL$^{\text{d}}$. We also compare the strengths and weaknesses of $\propto$MDL$^{\text{c}}$ and $\propto$MDL$^{\text{d}}$.

a) Correctness: We first introduce the following lemmas.

Lemma 1. *If* $|\mathcal{A}_w| \geq m$, $\prod_{A_i \in \mathcal{A}_w} (g^{s_i})^{z_i} = g^s$.

Proof. It can be verified that for $A_i \in \mathcal{A}_w$, its z-value $z_i = \prod_{A_i \in \mathcal{A}_w, j \neq i} \frac{-x_j}{x_i - x_j}$. Given the property of Shamir's protocol, secret s can be recovered by any subset (of cardinality at least m) of its shares, i.e., $s = \sum_{A_i \in \mathcal{A}_w} s_i z_i$, which leads to $\prod_{A_i \in \mathcal{A}_w} (g^{s_i})^{z_i} = g^s$. □

Lemma 2. *For* $A_i \in \mathcal{A}_w$, $d_i = g^{s_i z_i r}$.

Proof. We prove this lemma using induction. Suppose that it holds for any $\mathcal{A}_w$ with $|\mathcal{A}_w| \leq l - 1$ and that A_i is the l-th worker joining $\mathcal{A}_w$.

After the interaction between A_i and M, for $A_j \in \mathcal{A}_w$ ($j \neq i$), we have (we use x and x' to differentiate object x before and after A_i is included in $\mathcal{A}_w$):

$$d_j' = (d_j)^{\frac{-x_i}{x_j - x_i}} \cdot (g^{s_j})^{z_j' r_i} = g^{s_j z_j \frac{-x_i}{x_j - x_i} r + s_j z_j' r_i} = g^{s_j z_j' r'}$$

Further, by construction, we have $d_i' = g^{s_i z_i' r'}$. □

Based upon these lemmas, we have the following theorem.

Theorem 6. *If* $|\mathcal{A}_w| \geq m$, *the global gradient of* w *is decryptable:* $\nabla_w^g = \frac{1}{|\mathcal{A}_w|} \ln \left(\frac{\prod_{A_i \in \mathcal{A}_w} c_i}{\prod_{A_i \in \mathcal{A}_w} d_i} \right)$.

Proof. From the definition of encryption procedure, we have

$$\prod_{A_i \in \mathcal{A}_w} c_i = \exp \left(\sum_{A_i \in \mathcal{A}_w} \nabla_w^i \right) g^{sr}$$

Then for $|\mathcal{A}_w| \geq m$, given Lemma 1 and 2 and the property of Shamir's protocol, we have

$$\prod_{A_i \in \mathcal{A}_w} d_i = \prod_{A_i \in \mathcal{A}_w} g^{s_i z_i r} = g^{\sum_{A_i \in \mathcal{A}_w} s_i z_i r} = g^{sr}$$

leading to $\nabla_w^g = \frac{1}{|\mathcal{A}_w|} \ln \left(\frac{\prod_{A_i \in \mathcal{A}_w} c_i}{\prod_{A_i \in \mathcal{A}_w} d_i} \right) = \frac{\sum_{A_i \in \mathcal{A}_w} \nabla_w^i}{|\mathcal{A}_w|}$.

□

The proof of the overall correctness of the protocols is similar to Theorem 3, which we omit here.

b) Privacy: Similar to the analysis in Section IV, we demand privacy assurance for both global and local gradients. First, we have the following theorem that guards the privacy of global gradients.

Theorem 7. *If* $|\mathcal{A}_w| < m$ *(i.e., less than* m *workers contribute local gradients), the global gradient* ∇_w^g *is undecryptable.*

To show the privacy protection for local gradients, again, we assume all other parties collude with each other, attempting to reveal the local gradient of a victim worker.

Theorem 8. *In* $\propto$*MDL*^{d}, *if* $|\mathcal{A}_w| < m$, *the local gradient of any individual worker* $A_i \in \mathcal{A}_w$ *is undecryptable, even if all other workers in* $\mathcal{A}_w$ *collude with each other.*

Proof. The proofs of Theorem 7 and 8 are respectively similar to Theorem 4 and 5. □

c) Complexity: We consider the case that each worker contributes to a minimum number of parameters (i.e., $k^* = \lceil m \cdot t/n \rceil$); each parameter has exactly m contributors. In Algorithm 5, during each epoch, to update given parameter w, the worker A_i, assumed to arrive as the l-th worker in $\mathcal{A}_w$, needs to exchange and update $O(l)$ (i.e., on average, $O(m/2)$) integers with M. Thus, the average communication complexity for each worker is $O(\lceil m^2 \cdot t/n \rceil)$. In Algorithm 4, the communication cost of M is the aggregated communication cost across all participating workers, i.e., $O(m^2 \cdot t)$.

d) Comparison of $\propto$*MDL*^{c} *and* $\propto$*MDL*^{d}*:* Compared with $\propto$MDL$^{\text{c}}$, $\propto$MDL$^{\text{d}}$ requires less coordination, thereby achieving higher execution efficiency. Nevertheless, this advantage is not free. $\propto$MDL$^{\text{d}}$ features the communication complexity of $O(m^2 \cdot t)$, in contrast of $O(m \cdot t)$ in $\propto$MDL$^{\text{c}}$, which can be expensive for the settings of limited network bandwidth or a large number of participating workers.

VI. Empirical Evaluation

Next using benchmark datasets and DNN architectures, we empirically assess the performance of individual components and overall system of $\propto$MDL; we also explore the strengths and limitations of variants of $\propto$MDL.

A. Experiment Setting

Our experiments used two benchmark datasets, the MNIST[1] and SVHN[2] datasets. The former comprises 60K training and 10K testing (28×28 grayscale) images, while the latter constitutes 73K training and 26K testing (32×32 RGB) images. We consider MNIST and SVHN respectively representing "simple" and "hard" DL tasks. Using two disparate datasets, we intend to capture the impact of data dimensionality over the system performance. The private data of each worker comprises 60% random samples from the training dataset.

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

[2] http://ufldl.stanford.edu/housenumbers

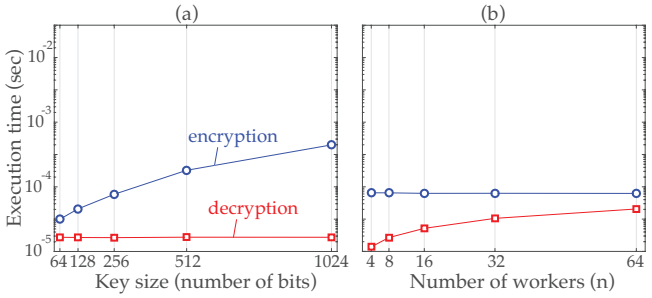

Figure 5: Execution time of encryption and decryption.

We implemented all alternative MDL solutions on Theano[3], one of the most popular DL platforms. We considered two major DNN architectures adapted from [5], multi-layer perception (MLP) and convolutional neural network (CNN), both of which have been widely used in image recognition tasks. We regard the CNN and MLP models respectively representing "strong" and "weak" DL models.

We contrasted the performance of alternative MDL frameworks using three metrics, *model utility*, *convergence rate* and *training efficiency*. Specifically, the model utility is measured by its classification accuracy over the testing data; the convergence rate is captured by the number of epochs necessary for the model to converge; and the training efficiency is measured by the end-to-end training time of the model.

The default parameter setting is as follows: the number of workers $n = 8$, the visibility threshold $\rho = 0.5$, the learning rate $\lambda = 0.01$, and the mini-batch size 128. Each worker is running a 3.40 GHz processor and 16 GB RAM.

B. Experiment Results

a) Encryption and Decryption: We start by assessing the overhead of the cryptographic machinery used by $\propto$MDL.

Figure 5 (a) shows how the execution time of the encryption and decryption operations grows with the key size of the cryptosystem. Observe that the encryption cost is proportional to the key size, whereas the key size has minimal impact on the decryption cost, as it only involves modular multiplication and multiplicative inverse calculations.

Further, Figure 5 (b) shows the execution time of encryption and decryption as a function of the number of workers n, where we fix the key size = 256 bits and $\rho = 0.5$. Clearly, the encryption cost is independent of the number of workers, while the decryption cost grows mildly with n. In both cases, the cryptographic operations incur fairly limited system overhead, compared with training the local models.

b) ρ-Visibility and Local Differential Privacy: Recall that $\propto$MDL integrates ρ-visibility with the local differential privacy (LDP) mechanism. In this set of experiments, we examine their joint impact on the utility of the global model. Specifically, we measure the classification accuracy of the trained models as the settings of LDP and ρ-visibility vary. For LDP, we consider the per-parameter privacy budget $\epsilon = 0.01, 0.1, 1$, with smaller budget implying stronger protection; for ρ-visibility, we consider the setting of $\rho = 0.125, 0.5$ with larger ρ meaning synchronization among more workers.

[3]http://deeplearning.net/software/theano/

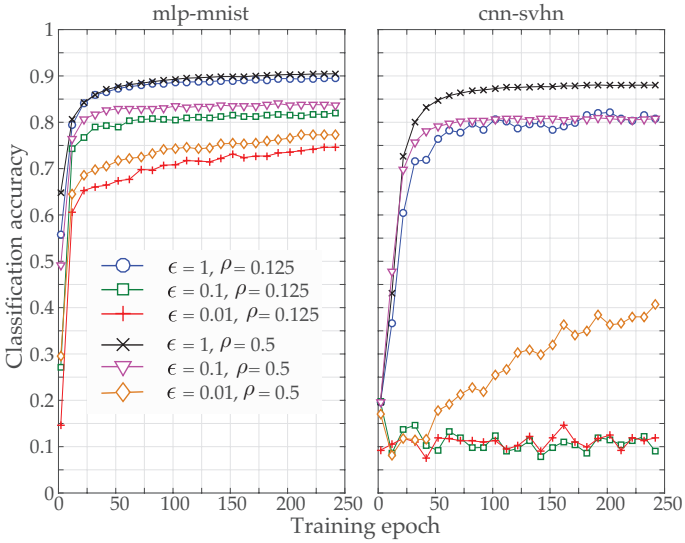

Figure 6: Synergy between differential privacy and ρ-visibility.

As shown in Figure 6, the large accuracy gap between $\epsilon = 0.01$ and 1 ($\rho = 0.125$) indicates the detrimental effect of the LDP mechanism on the model utility. For instance, in the case of CNN-SVHN, the training under $\epsilon = 0.1, 0.01$ even fails to converge. Yet, this negative impact is greatly mitigated by increasing ρ. For example, in the case of MLP-MNIST, the accuracy improves by 19.5% as we increase $\rho = 0.125$ to 0.5 under $\epsilon = 1$. Interestingly, this improvement is even more evident in the case of CNN-SVHN (333.0% accuracy boost). This phenomenon is explained by that as more workers aggregate their local gradients, due to the symmetric distribution used in sampling random noise, a larger part of injected noise cancels out, leading to more reliable global gradient estimates. More complicated DNN models (e.g., CNN) tend to be more sensitive to the reliability of such estimates.

Thus, we empirically show that ρ-visibility and LDP can be integrated in synergy and existing DP-based PEMs can be enhanced by ρ-visibility to achieve better utility.

c) Model Utility and Training Efficiency: Next we study the intricate tradeoff between model utility and training efficiency. We fix the LDP per-parameter budget as $\epsilon = 0.5$.

Figure 7 (a) depicts the classification accuracy of $\propto$MDL^c converges under varying settings of ρ (from 0.125 to 1). It is noticed that the model utility (accuracy) of $\propto$MDL^c is, to a large extent, determined by the combination of DNN models and DL tasks. For instance, in the case of MLP-SVHN (i.e., weak model versus hard task), none of the settings of ρ leads to any satisfying accuracy; while in the case of CNN-MLP (i.e., strong model versus easy task), $\propto$MDL^c converges to fairly similar accuracy under varied settings of ρ. Meanwhile, observe that the setting of ρ also significantly impacts the model utility under given task-model settings. For example, in the case of CNN-SVHN, the model trained under $\rho = 1$ achieves 91.4% accuracy, in comparison of 81.5% accuracy obtained by that trained under $\rho = 0.125$, which

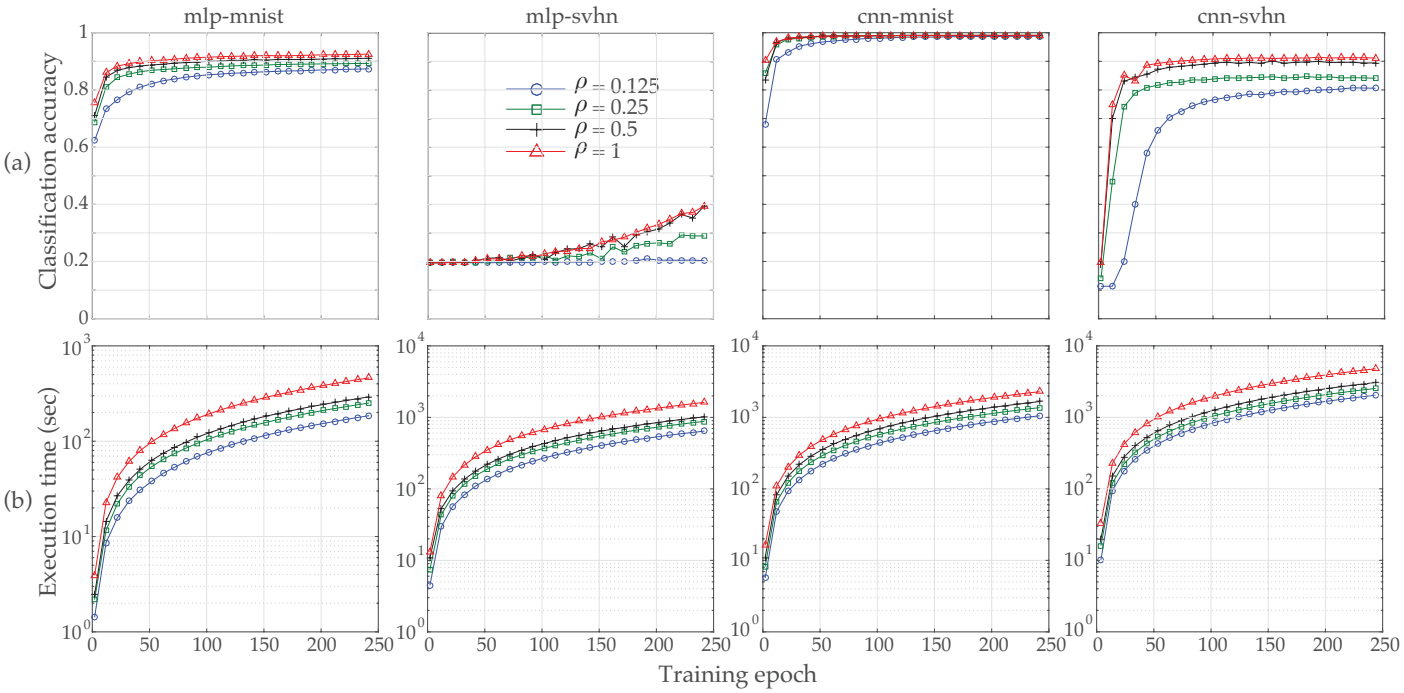

Figure 7: Tradeoff between model utility and training efficiency.

empirically validates our analysis in Section III. Also note that regardless of their eventual accuracy, most models converge approximately within the same number of epochs.

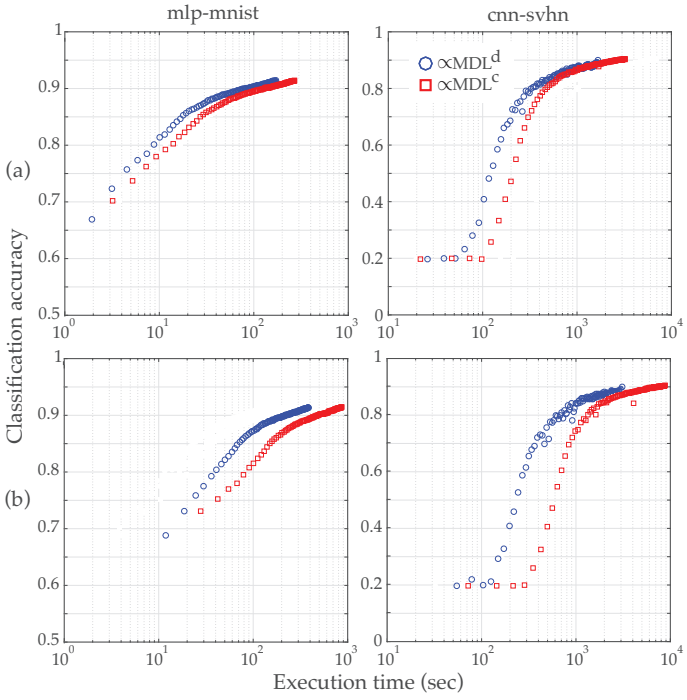

Figure 8: Training efficiency of $\propto$MDL^c and $\propto$MDL^d.

We then measure the training efficiency of $\propto$MDL^c under varying ρ. As suggested by our analysis, larger ρ implies higher training cost per epoch, as the manager needs to coordinate more workers to commit local gradients. Nevertheless, it is observed in Figure 7 (b) that across all DNN model

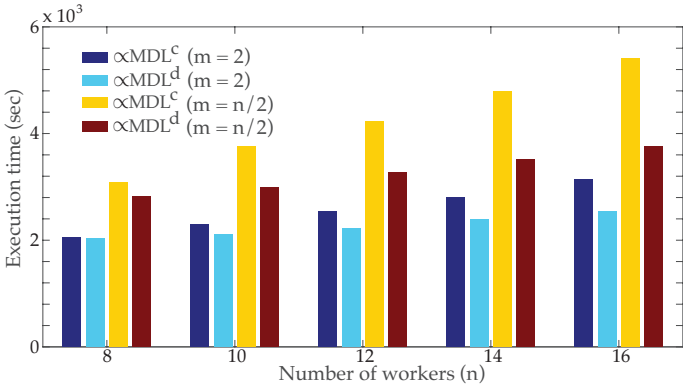

Figure 9: Execution time (250 epochs) of $\propto$MDL with respect to the number of participating workers.

and DL task combinations, the increased cost is not prohibitive. For example, in the case of CNN-SVHN, as ρ increases from 0.125 to 1, the execution time grows about 3 times. However, in large-scale training tasks or low-end system configurations (e.g., limited network bandwidth), a decision has to be made to balance training efficiency and model utility.

d) $\propto$MDL^c versus $\propto$MDL^d: We compare the training efficacy of the two variants of $\propto$MDL. Recall our analysis in Section V that at the expense of slightly higher communication cost, $\propto$MDL^d achieves better training efficiency, especially when different workers possess heterogenous computation capacities.

Figure 8 (a) shows the execution time of $\propto$MDL^c and $\propto$MDL^d with $\rho = 0.5$. In both cases of MLP-MNIST and CNN-SVHN, $\propto$MDL^d converges much faster than $\propto$MDL^c. For example, in the case of CNN-SVHN, $\propto$MDL^c requires more than 27.0% of training time than $\propto$MDL^d to reach 85%

accuracy, which empirically validates our analysis.

We further consider the setting wherein the workers possess imbalanced computation power. To construct this scenario, we introduce delay in each epoch. The delay time of each worker is drawn randomly from a worker-specific normal distribution. In particular, we set the normal distribution for the i-th worker to be $\mathcal{N}(5i, 1)$. As shown in Figure 8 (b), $\propto$MDL$^{\text{d}}$ demonstrates even more evident advantage than that in Figure 8 (a). This implies that $\propto$MDL$^{\text{d}}$ is more suitable for scenarios of heterogenous computation capacities. Astute readers may point out that this may lead to the fairness issue (i.e., workers with weaker computation power may be unable to contribute to MDL at their will). We consider addressing this issue one future research direction.

Also note that although it is often the case that the cost of local training dominates the communication cost, in settings where this premise does not hold (e.g., extremely limited network bandwidth), the superiority of $\propto$MDL$^{\text{d}}$ may not hold either. It is up to the MDL operators to choose the right variant of $\propto$MDL for given settings.

e) Scalability: In the last set of experiments, we study the scalability of $\propto$MDL with respect to the number of workers. Figure 9 shows the training time (250 epochs) of $\propto$MDL as the number of workers (n) varies from 8 to 16. We consider two settings, $m = \rho \cdot n = 2$ and $m = n/2$.

Observe that with fixed m, n has fairly limited impact on the efficiency of $\propto$MDL. For example, the execution time of $\propto$MDL$^{\text{c}}$ increases by 1,076 seconds as n grows from 8 to 16. In comparison, with ρ fixed, n influences the execution efficiency of $\propto$MDL more significantly. For example, the running time of $\propto$MDL$^{\text{c}}$ increases by 2,318 seconds as m grows from 4 to 8. This indicates that compared with n, m has a larger impact on the training efficiency of $\propto$MDL$^{\text{c}}$. Also note that compared with $\propto$MDL$^{\text{c}}$, $\propto$MDL$^{\text{d}}$ is much less sensitive to the setting of n or m, thanks to its more dynamic and adaptive nature.

VII. Related Work

DL proves extremely effective at learning nonlinear features and functions from complex data. Besides beating records in image recognition, text classification, and natural language understanding [1], DL outperforms traditional machine learning in healthcare domains, e.g., predicting the effect of mutations in non-coding DNA [25]. The training data in such applications is often highly sensitive, thus requiring incorporating effective privacy enhancing mechanisms (PEMs). Our goal is to protect the input privacy of participating parties in MDL; thus PEMs designed for protecting model privacy [26], [27] and output privacy [28] well complement this work.

Techniques using secure multiparty computation (SMC) help protect input privacy of multiple parties when they collaboratively train machine learning models on their proprietary data. SMC has been applied to design algorithm-specific PEMs, e.g., linear regression functions [12], association rule [11] and Naïve Bayes classifiers [29]. However, most of carefully engineered SMC protocols incur non-trivial performance overhead, not to mention general-purpose SMC [16], [30]. Thus, the application of SMC to DL remains an open question.

As the de facto privacy definition, differential privacy (DP) [6] has also been applied to construct PEMs, e.g., principal component analysis [31], regression [32], risk minimization [18] and neural network [33], [34]. However, most of these PEMs are designed for the setting of centralized data repositories, thus inapplicable for MDL.

The most relevant work is perhaps to adapt stochastic gradient descent to the multiparty setting [5]. Iteratively, each party partially shares its local model with others and applies the updated global model to its private data. However, it is unclear how much sensitive information is leaked in the revealed local models. One solution to this indirect leakage is to apply DP during training local models; yet, this leads to significant utility loss in the global model. $\propto$MDL addresses this issue by seamlessly integrating with DP in synergy, yielding PEMs with strong privacy assurance and desirable model utility simultaneously.

VIII. Conclusion & Future Work

This work presents the design, implementation and evaluation of a new MDL paradigm which departs from prior work with a series of distinct features: (i) it provides strong privacy assurance and desirable model utility simultaneously; (ii) it offers an intuitive handle for practitioners to balance model utility and training efficiency; (iii) it also enables delicate control over communication and computational costs, thus optimizable for given system settings.

This work also opens several directions for further investigations. For example, in current implementation, we use a simple contribution-based download policy. It is worth to study more complicated policies and their impact on the system performance. It is also interesting to explore cryptosystems other than ElGamal (e.g., Paillier [35] natively supports additive homomorphism), which may further improve training efficiency. Further, while $\propto$MDL$^{\text{d}}$ adapts to the setting of participating parties with imbalanced computation capacities, we need to address the potential fairness issue of unequal contributions by different parties. Finally, each session of worker-manager interactions is implemented as a meta transaction. It is possible to design protocols that interleave multiple sessions, with the hope of achieving even higher concurrency.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. 1566526, 1350324, and 1464800. Shouling Ji was partly supported by the Provincial Key Research and Development Program of Zhejiang, China under No. 2016C01G2010916, the Fundamental Research Funds for the Central Universities, the Alibaba-Zhejiang University Joint Research Center for Frontier Technologies under Program No. XT622017000118, and the CCF-Tencent Open Research Fund under No. AGR20160109.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] M. Pathak, S. Rane, and B. Raj, "Multiparty differential privacy via aggregation of locally trained classifiers," in *NIPS*, 2010.

[4] A. Rajkumar and S. Agarwal, "A differentially private stochastic gradient descent algorithm for multiparty classification," in *AISTATS*, 2012.

[5] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *CCS*, 2015.

[6] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3-4, pp. 211–407, 2014.

[7] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 492–542, 2016.

[8] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola, "Parallelized stochastic gradient descent," in *NIPS*, 2010.

[9] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. aurelio Ranzato, A. Senior, P. Tucker, K. Yang, Q. V. Le, and A. Y. Ng, "Large Scale Distributed Deep Networks," in *NIPS*, 2012.

[10] Y. Lindell and B. Pinkas, "Privacy Preserving Data Mining," in *CRYPTO*, 2000.

[11] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *KDD*, 2002.

[12] W. Du, S. Chen, and Y. S. Han, "Privacy-preserving multivariate statistical analysis: Linear regression and classification," in *SDM*, 2004.

[13] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[14] T. El Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *CRYPTO*, 1985.

[15] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *STOC*, 2009.

[16] a. Shelat and C.-h. Shen, "Fast two-party secure computation with minimal assumptions," in *CCS*, 2013.

[17] P. Mohassel, M. Rosulek, and Y. Zhang, "Fast and secure three-party computation: The garbled circuit approach," in *CCS*, 2015.

[18] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *J. Mach. Learn. Res.*, vol. 12, pp. 1069–1109, 2011.

[19] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *SEC*, 2014.

[20] B. Schneier, *Applied Cryptography (2Nd Ed.): Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., 1995.

[21] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," in *CRYPTO*, 1990.

[22] T. Hwang, "Cryptosystem for group oriented cryptography," in *EUROCRYPT*, 1991.

[23] M. Kim, A. Mohaisen, J. H. Cheon, and Y. Kim, "Private over-threshold aggregation protocols," in *ICISC*, 2013.

[24] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *STOC*, 1985.

[25] H. Y. Xiong, B. Alipanahi, L. J. Lee, H. Bretschneider, D. Merico, R. K. C. Yuen, Y. Hua, S. Gueroussov, H. S. Najafabadi, T. R. Hughes, Q. Morris, Y. Barash, A. R. Krainer, N. Jojic, S. W. Scherer, B. J. Blencowe, and B. J. Frey, "The human splicing code reveals new insights into the genetic determinants of disease," *Science*, vol. 347, no. 6218, 2015.

[26] M. Pathak, S. Rane, W. Sun, and B. Raj, "Privacy preserving probabilistic inference with hidden markov models," in *ICASSP*, 2011.

[27] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *NDSS*, 2015.

[28] T. Wang and L. Liu, "Output privacy in data mining," *ACM Trans. Database Syst.*, vol. 36, no. 1, pp. 1–34, 2011.

[29] J. Vaidya, M. Kantarcioglu, and C. Clifton, "Privacy-preserving naïve bayes classification," *The VLDB Journal*, vol. 17, no. 4, pp. 879–898, 2008.

[30] K. Nayak, X. S. Wang, S. Ioannidis, U. Weinsberg, N. Taft, and E. Shi, "Graphsc: Parallel secure computation made easy," in *S&P*, 2015.

[31] K. Chaudhuri, A. D. Sarwate, and K. Sinha, "A near-optimal algorithm for differentially-private principal components," *J. Mach. Learn. Res.*, vol. 14, no. 1, pp. 2905–2943, 2013.

[32] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, "Functional mechanism: Regression analysis under differential privacy," *Proc. VLDB Endow.*, vol. 5, no. 11, pp. 1364–1375, 2012.

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[34] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *CCS*, 2016.

[35] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, 1999.