A Profit-Maximizing Model Marketplace with Differentially Private Federated Learning

Peng Sun[†,‡], Xu Chen[≀], Guocheng Liao[§], and Jianwei Huang[†,‡,*]
[†]School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen, China
[‡]Shenzhen Institute of Artificial Intelligence and Robotics for Society (AIRS), Shenzhen, China
[≀]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[§]School of Software Engineering, Sun Yat-sen University, Zhuhai, China
Email: sunpeng@cuhk.edu.cn, {chenxu35, liaogch6}@mail.sysu.edu.cn, jianweihuang@cuhk.edu.cn

IEEE INFOCOM 2022 - IEEE Conference on Computer Communications | 978-1-6654-5822-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/INFOCOM48880.2022.9796833

***Abstract*—Existing machine learning (ML) model marketplaces generally require data owners to share their raw data, leading to serious privacy concerns. Federated learning (FL) can partially alleviate this issue by enabling model training without raw data exchange. However, data owners are still susceptible to privacy leakage from gradient exposure in FL, which discourages their participation. In this work, we advocate a novel differentially private FL (DPFL)-based ML model marketplace. We focus on the broker-centric design. Specifically, the broker first incentivizes data owners to participate in model training via DPFL by offering privacy protection as per their privacy budgets and explicitly accounting for their privacy costs. Then, it conducts optimal model versioning and pricing to sell the obtained model versions to model buyers. In particular, we focus on the broker's profit maximization, which is challenging due to the significant difficulties in the revenue characterization of model trading and the cost estimation of DPFL model training. We propose a two-layer optimization framework to address it, i.e., revenue maximization and cost minimization under model quality constraints. The latter is still challenging due to its non-convexity and integer constraints. We hence propose efficient algorithms, and their performances are both theoretically guaranteed and empirically validated.**

***Index Terms*—ML model marketplace, federated learning, differential privacy, incentive mechanism**

I. Introduction

A. Background and Motivations

A marketplace for machine learning (ML) models, which connects data owners and model buyers, can significantly facilitate various ML applications [1]–[4]. Recent years have witnessed the emergence of several online ML model marketplaces, such as Modzy [5].

The conventional centralized ML paradigm adopted in existing model marketplaces requires data owners to share their raw data, leading to serious privacy concerns. Instead, the newly-emerging federated learning (FL) [6]–[12] enables multiple data owners to collaboratively train ML models without raw data exchange, which is a promising approach to enable next-generation model marketplace design to alleviate data owners' privacy concerns.

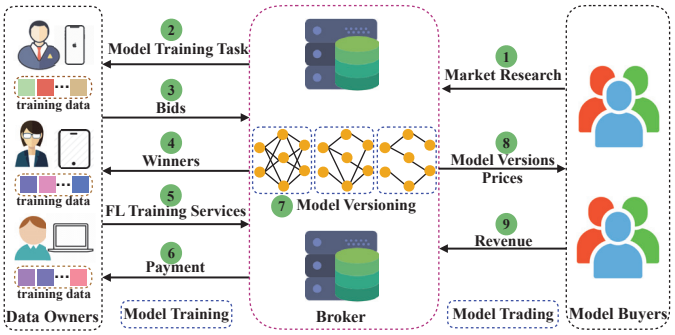

Fig. 1. Framework of *DEVELOP* (where circled numbers represent the sequence of the events).

However, FL cannot provide an absolute privacy guarantee. Recent works (e.g., [13]–[18]) have demonstrated that data owners involved in FL are still susceptible to privacy attacks. This is because the gradient exposed during FL also contains data owners' sensitive information. Typical privacy attacks include membership inference attacks (e.g., [14], [15]) that infer whether a particular data sample (e.g., a patient profile) belongs to a specific data owner, and reconstruction attacks (e.g., [16]–[18]) that recover data owners' private data (e.g., facial images).

To safeguard data owners against these privacy threats from gradient exposure, researchers have proposed differentially private federated learning (DPFL) (e.g., [19]–[23]). In the DPFL framework, the involved data owners obfuscate their local model updates or intermediate gradients (according to their privacy budgets) before sharing in each round of FL. Nonetheless, data owners under differential privacy (DP) protection (with moderate privacy budgets) still sustain a certain degree of potential privacy disclosure, incurring some privacy cost [24], [25]. Without an effective incentive mechanism incorporating such privacy cost, self-interested data owners will be reluctant to participate in model training, leading to an inefficient model marketplace. However, to the best of our knowledge, none of the existing model marketplace design has jointly considered the privacy protection and incentive design for data owners.

This work is supported by the Shenzhen Science and Technology Program (JCYJ20210324120011032), Shenzhen Institute of Artificial Intelligence and Robotics for Society, the Presidential Fund from the Chinese University of Hong Kong, Shenzhen, the National Natural Science Foundation of China (Grant No. 62102337, No. U20A20159, No. 61972432), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (Grant No. 2017ZT07X355), and China Postdoctoral Science Foundation (Grant No. 2021M703079).

*Jianwei Huang is the corresponding author.

Motivated by this, in this paper, we promote a novel DiffErentially priVate fEderated Learning-based ML mOdel marketPlace (*DEVELOP*). As shown in Fig. 1, *DEVELOP* consists of two major interaction processes, i.e., the DPFL-based model training process between data owners and the broker, and the model trading process between model buyers and the broker. Specifically, the broker incentivizes data owners (by explicitly accounting for their privacy costs) to train an ML model via DPFL and then sells the model to model buyers. The broker wishes to maximize her profit, which is the revenue extracted from selling an ML model minus the cost of incentivizing data owners to train this model.

There are several challenges when designing the profit-maximizing *DEVELOP*. First, it is challenging to characterize the revenue function (and then the profit function) due to model buyers' heterogeneous model quality preferences and valuations. In addition, the complex distributed on-device model training process of FL makes it challenging to figure out how many and which subset of data owners can train an ML model with desired quality. Thus, it is difficult to estimate the model training cost. These issues motivate us to tackle the first key question of this paper:

Question 1: *How to characterize model trading revenue and estimate model training cost to determine the profit-maximizing model to train?*

Second, it is challenging for the broker to incentivize only the desirable data owners to participate in DPFL model training. Specifically, recruiting data owners with larger privacy budgets helps enhance model quality, and recruiting those with less privacy costs helps reduce payment expenditure. However, data owners' privacy costs (dependent on privacy budgets) are their private information, which may not be easily accessed by the broker. Moreover, unlike vanilla FL where more involved data owners always lead to better models, the relationship between the model quality and the number of involved data owners (with heterogeneous privacy budgets) is much more complicated in DPFL. This drastically increases the difficulty in determining the payment for data owners while preventing untruthful behaviors. These issues lead to the second key question of this paper:

Question 2: *How to incentivize data owners with heterogeneous privacy budgets and (private) privacy costs to participate in DPFL model training under the quality constraints?*

B. Contributions

We summarize our key contributions as below.

- *DPFL-based ML Model Marketplace Design:* To the best of our knowledge, the proposed *DEVELOP* is the first ML model marketplace with differentially private federated learning. *DEVELOP* jointly accounts for the privacy preservation and incentive design, and hence can encourage more data owners to contribute in FL model training. This is crucial for operating an efficient ML model marketplace.
- *Profit Maximization:* The broker's profit maximization problem is challenging due to the significant difficulties in characterizing the revenue function and estimating the cost of FL model training. We propose a two-layer optimization framework to address it, i.e., revenue maximization and cost minimization given a model quality constraint.
- *Algorithm Design for Cost Minimization:* The cost minimization problem is still challenging as it is an integer non-convex optimization problem. We design efficient algorithms under both complete and incomplete information scenarios, and their outstanding performances are theoretically guaranteed and empirically validated.
- *Numerical Results:* Numerical results show that the proposed algorithms respectively achieve up to 97.6% and 82.5% under the complete and incomplete information scenarios, of the maximum profit obtained by the optimal solution. Surprisingly, in some market settings, blindly training and then selling the best model needed in the market could lead to a maximum of 97% profit loss and even cause a deficit for the broker under the complete and incomplete information scenarios, respectively.

II. Related Work

In this section, we review and discuss the related work from two aspects, i.e., ML model marketplace design and incentive mechanisms for FL.

A. ML Model Marketplace Design

The increasing pervasiveness of ML applications has led to an emerging interest in ML model marketplace design (e.g., [1]–[3], [26]). Chen *et al.* in [2] proposed the first formal framework of ML model marketplace which directly prices models instead of data. Liu *et al.* in [3] built an end-to-end ML model marketplace which considers both compensation for data owners and model pricing for buyers.

Our work differs from [2], [3] in two aspects. First, instead of the conventional centralized ML paradigm, we adopt FL for model building, which could significantly mitigate data owners' privacy concerns. Second, rather than revenue maximization, we focus on profit (i.e., revenue minus cost) maximization for the broker, which is more complicated as the broker needs to interact with both model buyers and data owners. Moreover, the complex distributed on-device model training process in FL makes it more challenging to evaluate the model quality and then estimate the model training cost.

B. Incentive Mechanisms for FL

Despite a wide spectrum of incentive mechanisms developed for FL (e.g., [24], [25], [27]–[35]), only a few of them [24], [25], [35] incorporated data owners' privacy concerns. Specifically, Hu *et al.* in [24] proposed two-stage Stackelberg game-based incentive mechanism that compensates users' costs of privacy leakage in FL. Wu *et al.* in [25] employed a multi-dimensional contract approach to jointly account for data owners' computation, communication, and privacy costs. Zheng *et al.* in [35] combined a reverse auction-based incentive scheme and a novel gradient aggregation method for privacy-preserving FL.

Our work differs from [24], [25], [35] in two aspects. First, instead of optimizing model performance or minimizing gradient aggregation error, we focus on training cost minimization under a complicated model quality constraint. Second, we consider an end-to-end ML model marketplace, and our incentive design is dependent on model buyers' characteristics. The two aspects substantially increase the challenge of designing a truthful incentive mechanism.

III. Preliminaries

In this section, we briefly introduce some background knowledge of FL and DP.

A. Federated Learning

FL is a distributed ML paradigm which enables multiple data owners to collaboratively train a shared ML model without disclosing their raw data, under the coordination of a central server. Consider an FL system involving a set $\mathcal{N} = \{1, \ldots, N\}$ data owners. Each data owner $n \in \mathcal{N}$ has a private training dataset $\mathcal{D}_n = \left\{\left(\mathbf{x}_1^n, y_1^n\right), \ldots, \left(\mathbf{x}_{D_n}^n, y_{D_n}^n\right)\right\}$ including D_n data samples (i.e., $|\mathcal{D}_n| = D_n$), with $\mathbf{x}_j^n$ and y_j^n respectively denoting the feature vector and the corresponding label of the j-th training sample at data owner n.

We use $\ell\left(\mathbf{x}, y; \boldsymbol{\theta}\right)$ to represent the loss function associated with the data sample $(\mathbf{x}, y)$ and the parameter vector $\boldsymbol{\theta}$. Then the local loss function at data owner n is represented as $f_n(\boldsymbol{\theta}) = \frac{1}{D_n}\sum_{j=1}^{D_n} \ell\left(\mathbf{x}_j^n, y_j^n; \boldsymbol{\theta}\right)$, which is usually assumed to be L-Lipschitz smooth [20]. Then, the global loss function is $f(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n\in\mathcal{N}} f_n(\boldsymbol{\theta})$. The goal of FL is to learn a shared ML model with parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$ by solving the empirical risk minimization (ERM) problem $\min_{\boldsymbol{\theta}\in\mathbb{R}^d} f(\boldsymbol{\theta})$ [6].

A classic and widely-adopted method to solve the above ERM problem is distributed stochastic gradient descent [36]. This method involves a number of communication rounds T between data owners and the server and has a convergence rate of $O(1/\sqrt{NT})$ [36]. Specifically, in each communication round $t \in \{0, 1, \ldots, T-1\}$, an engaged data owner n first downloads the latest global model $\boldsymbol{\theta}_t$ ($\boldsymbol{\theta}_0$ at the very beginning is a random initialization) from the server. Then, using a mini-batch $\mathcal{B}_n \subseteq \mathcal{D}_n$ of size B_n, data owner n computes a stochastic gradient $g\left(\mathcal{B}_n; \boldsymbol{\theta}^t\right) = 1/B_n \sum_{\left(\mathbf{x}_j^n, y_j^n\right)\in\mathcal{B}_n} \nabla\ell\left(\mathbf{x}_j^n, y_j^n; \boldsymbol{\theta}^t\right)$ and uploads it to the server. Upon receiving stochastic gradients from all engaged data owners at round t, the server updates the global model with a learning rate η as

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \frac{\eta}{N}\sum_{n\in\mathcal{N}} g\left(\mathcal{B}_n; \boldsymbol{\theta}^t\right). \tag{1}$$

B. Differential Privacy

A randomized mechanism $\mathcal{A}$ is differentially private if and only if the inclusion of a single instance in the input dataset of $\mathcal{A}$ causes statistically indistinguishable changes to the output [37]. In this work, we leverage ρ-zero-concentrated differential privacy (henceforth ρ-zCDP) proposed in [38] for privacy analysis. Its privacy parameter composes nicely in sequential algorithms. Thus, it is suitable to analyze the overall privacy loss of data owners participating in DPFL, which involves sequential stochastic gradient computations.

Before presenting the formal definition of ρ-zCDP, we first define a privacy loss random variable Y. Specifically, we define the privacy loss Y of a randomized mechanism $\mathcal{A} : \mathcal{D} \to \mathcal{R}$ for any two adjacent datasets $D, D' \in \mathcal{D}$ (that differ on one data sample) and an output $o \in \mathcal{R}$ as

$$Y = \log \frac{\Pr[\mathcal{A}(D) = o]}{\Pr\left[\mathcal{A}\left(D'\right) = o\right]}. \tag{2}$$

The formal definition of ρ-zCDP is then as below.

Definition 1. *(ρ-zCDP [38]): A randomized mechanism $\mathcal{A} : \mathcal{D} \to \mathcal{R}$ satisfies ρ-zCDP if for any $\alpha \in (1, \infty)$, we have*

$$\mathbb{E}\left[e^{(\alpha-1)Y}\right] \leq e^{(\alpha-1)(\rho\alpha)}, \tag{3}$$

where Y is the privacy loss random variable defined in Eq. (2), and $\mathbb{E}$ denotes the expectation operator over Y.

The bound in Eq. (3) implies that Y resembles a Gaussian distribution with mean ρ and variance 2ρ [38]. A closer-to-zero ρ implies a smaller privacy loss Y with a higher probability. Later in Section IV-C, we will leverage the parameter ρ to quantify a data owner's privacy budget. Next, we present how to achieve ρ-zCDP with Gaussian mechanism in Lemma 1.

Lemma 1 (Gaussian Mechanism [38])**.** *Given a query function $h : \mathcal{D} \to \mathcal{R}$ with a query sensitivity $\Delta_h = \max_{D,D'} |h(D) - h(D')|$ for any two adjacent datasets $D, D' \in \mathcal{D}$, the Gaussian mechanism $\mathcal{A} : \mathcal{D} \to \mathcal{R}$, which takes as input a dataset $D \in \mathcal{D}$ and outputs $\mathcal{A}(D) = h(D) + \mathcal{N}(0, \sigma^2)$, satisfies $(\Delta_h^2/2\sigma^2)$-zCDP, where σ^2 is the variance of the Gaussian random noise.*

IV. System Model

In this section, we first provide an overview of *DEVELOP* including the three entities and their operations in the market. Then, we describe the two major interaction processes, i.e., the model trading between model buyers and the broker, as well as the model training between data owners and the broker.

A. Overview of DEVELOP

As shown in Fig. 1, *DEVELOP* consists of the following three entities.

- **Model Buyers:** Model buyers have diverse demands for ML models with various quality levels.
- **Data Owners:** Data owners can participate in model training via DPFL, which requires sufficient incentives.
- **Broker:** The broker acts as a middleman and operates the marketplace. It recruits data owners to train an ML model and then sells the model to model buyers. In addition, the broker also coordinates the DPFL process.

In this work, we focus on the broker-centric design and the objective of *DEVELOP* is to maximize the broker's profit, which is the revenue extracted from model trading to buyers minus the cost of incentivizing data owners for model training. Next, we introduce these two interaction processes.

B. Model Trading

In this subsection, we introduce the model trading interaction between the broker and model buyers.

1) Versioning: To satisfy model buyers' diverse demands for ML models with various quality levels, the broker needs to offer different versions of an ML model, which is called versioning [39]. There are various ways to achieve ML model versioning. In this work, we use a simple model pruning method [40]. Specifically, given an ML model, the broker could derive other versions of the model with lower quality and more lightweight structure by pruning synapses or neurons. Thus, different versions of a model present different quality levels and requirements on execution capabilities, and running a better-quality version requests stronger capabilities.

2) Pricing: The broker needs to know how many versions are desired by model buyers and set a price for each version. Through standard market research, the broker could determine the $\mathcal{K} = \{1, \ldots, K\}$ candidate model versions and the quality vector $\boldsymbol{Q} = \{Q_k, k \in \mathcal{K}\}$, where $Q_1 < Q_2 < \ldots < Q_K$. It is natural that the broker sells different versions of an ML model at different prices. Specifically, the broker specifies a price menu $\boldsymbol{p} = \{p_k, k \in \mathcal{K}\}$ for all K model versions.

3) Revenue: After pricing each model version, we could characterize the revenue function. Assume there are C model buyers in the market. Among the model versions that satisfy the buyer's quality requirement, we assume that each model buyer only considers buying the most lightweight one. This is due to her limited execution capability constraint that potentially hinders the deployment of more powerful model versions (requesting stronger capabilities).

Version Preference Distribution: We assume that the version preference of model buyers follows a distribution with a probability mass function $g_k(\cdot)$, meaning that there are $C_k = C \times g_k(\cdot)$ potential buyers who are interested in the kth model version, in expectation.

Valuation Distribution: A model buyer will finally purchase her preferred model version if she has a valuation larger than the price. For the kth version, we assume that the valuations of potential buyers are drawn from a distribution with cumulative distribution function $V_k(\cdot)$. With a slight abuse of notation, we group all K valuation distributions into a vector $\boldsymbol{V} = \{V_k(\cdot), k \in \mathcal{K}\}$.

Similar as [41], we make a Bayesian assumption that the broker could obtain the knowledge of the distributions $g_k(\cdot)$ for $k \in \mathcal{K}$ and the vector $\boldsymbol{V}$ by learning historical transactions. Note that this is a common assumption in economic literature [42]. Then, we can characterize the revenue function for each model version.

Revenue Characterization: If a model buyer interested in model version k has a valuation larger than the price p_k, she would purchase this version. In this case, the broker could obtain a revenue of p_k. The expected revenue of selling the kth version to buyers interested in it is calculated as:

$$r_k = C \times g_k(\cdot) \times (1 - V_k(p_k)) \times p_k. \tag{4}$$

Moreover, an ML model, as one kind of digital goods, is generally considered to have a fixed cost of production (the model training cost here) but a negligible cost of reproduction. Specifically, the broker could degrade a high-quality model to lower-quality versions without incurring additional costs using techniques like model pruning. Thus, when the broker creates the kth model version, despite obtaining the revenue r_k as in Eq. (4), she could also extract revenue r_i from each of the lower model versions $1 \leq i \leq k-1$. Therefore, the expected accumulative revenue for the broker to conduct FL to obtain the kth model version is $R_k = \sum_{i=1}^{k} r_i$.

C. Model Training

In this subsection, we describe the model training process via DPFL, where data owners perform gradient perturbation according to their privacy budgets.

Data owners perform gradient perturbation when participating in FL to preserve privacy. To thwart privacy attacks that infer private information from their shared gradients, data owners will adopt the Gaussian mechanism to obfuscate their calculated gradients.

Given the stochastic gradient $g\left(\mathcal{B}_n; \boldsymbol{\theta}^t\right)$ of data owner $n \in \mathcal{N}$ in round t, she would inject Gaussian noises $\mathbf{b}_n \sim \mathbb{N}\left(0, \sigma_n^2 \mathbf{I}_d\right)$ (with $\mathbf{I}_d$ denoting the d-dimensional identity matrix) into $g\left(\mathcal{B}_n; \boldsymbol{\theta}^t\right)$, leading to the perturbed stochastic gradient $\tilde{g}\left(\mathcal{B}_n; \boldsymbol{\theta}^t\right)$. Formally,

$$\tilde{g}\left(\mathcal{B}_n; \boldsymbol{\theta}^t\right) = g\left(\mathcal{B}_n; \boldsymbol{\theta}^t\right) + \mathbf{b}_n. \tag{5}$$

Generally, data owners will adopt different noise variances σ_n^2 for gradient perturbation, which are determined by their heterogeneous privacy budgets (i.e., maximum tolerable privacy loss) ρ_n. Intuitively, a data owner will perform a weaker gradient perturbation (smaller σ_n^2) if she has a larger privacy budget (i.e., she can tolerate more privacy loss). Leveraging Lemma 1, we describe the explicit relationship between σ_n^2 and ρ_n in Lemma 2.

Lemma 2. *In DPFL, if data owner n with a privacy budget of ρ_n adopts the Gaussian mechanism for gradient perturbation, the noise variance is $\sigma_n^2 = \frac{2L^2}{\rho_n B_n^2}$, where L is the Lipschitz constant and B_n is the mini-batch size.*

Even with gradient perturbation, data owners still sustain a certain degree of privacy leakage, leading to privacy cost. According to [43], a data owner's privacy cost is determined by two factors: her privacy budget ρ_n, and her privacy preference α_n. The privacy preference measures how sensitive towards privacy leakage she behaves. Specifically, the privacy cost c_n of data owner n is computed as $c_n = \alpha_n c^p(\rho_n)$, where $c^p(\rho_n)$ is the adopted privacy cost function. There are different forms of $c^p(\rho_n)$. For example, a linear function yields $c_n = \alpha_n \rho_n$.

Given data owners' heterogeneous privacy budgets and privacy costs, the broker would like to select a set $\mathcal{W}$ of W data owners from $\mathcal{N}$ ($\mathcal{W} \subseteq \mathcal{N}$) to participate in model training via DPFL, and in each round updates the global model $\boldsymbol{\theta}^t$ as

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \frac{\eta}{W} \sum_{w \in \mathcal{W}} \left(g\left(\mathcal{B}_w; \boldsymbol{\theta}^t\right) + \mathbf{b}_w\right). \tag{6}$$

Here, a key issue is how to select $\mathcal{W}$ to maximize the broker's profit. We will address it in the next section.

V. Broker's Profit Maximization Problem

In this section, from a broker-centered perspective, we first formulate its profit maximization problem, and then decompose it into a two-layer optimization problem, i.e., revenue maximization from model trading and cost minimization of DPFL model training under quality constraints.

A. Profit Maximization

In this subsection, we formulate the broker's profit maximization problem and present the challenges of solving it.

1) Tradeoff between Revenue and Cost: Revenue from model trading and cost of model training jointly determine the profit. When the broker creates model version k, the revenue is $R_k = \sum_{i=1}^{k} r_i$. Thus, a higher model version always brings the broker a larger revenue. Meanwhile, the broker needs to recruit more data owners with larger privacy budgets (inducing larger privacy costs) to train a higher model version, leading to a larger model training cost.

2) Profit Maximization Formulation: Denote the cost of recruiting data owners in $\mathcal{W}$ for model training and the revenue created by the trained model by $S(\mathcal{W})$ and $R(\mathcal{W})$, respectively. Then, the profit is $\Phi(\mathcal{W}) = R(\mathcal{W}) - S(\mathcal{W})$. The broker aims to find the optimal data owner set $\mathcal{W}^*$ that creates the largest profit:

Problem 1. *Profit Maximization for the Broker.*

$$\mathcal{W}^* = \arg\max_{\mathcal{W}\subseteq\mathcal{N}} R(\mathcal{W}) - S(\mathcal{W}). \tag{7}$$

3) Key Challenges of Profit Maximization: Solving Problem 1 via classic optimization methods is very challenging. This is because the distributions of model buyers' version preferences and valuations are rather complicated in practical model marketplaces, and it is difficult to obtain an explicit expression of $R(\mathcal{W})$ in terms of $\mathcal{W}$ (and then $\Phi(\mathcal{W})$).

B. Two-Layer Optimization Framework

In this subsection, we first elaborate on how we decompose the profit maximization problem into revenue maximization and cost minimization. Then, we sequentially address them.

1) Rationale of Problem Decomposition: In practical ML model marketplaces, the broker normally offers only a small number of model versions [3], [41]. Thus, our basic idea to solve the profit maximization problem is to enumerate the profit of each model version and select the maximum one as the final result. To calculate the profit of each model version k, by the definition of profit, we need to figure out the highest possible revenue R_k and its corresponding lowest model training cost.

2) Revenue Maximization for Each Model Version: As described in Section IV-B (in particular Eq. (4)), the optimal price p_k^* for model version k is determined as

$$p_k^* = \arg\max_{p_k} \left[(1 - V_k(p_k)) \times p_k\right]. \tag{8}$$

Using the above optimal pricing strategy, the maximum expected (accumulative) revenue extracted from obtaining the kth model version via DPFL is then calculated as

$$R_k = \sum_{i=1}^{k} r_i = \sum_{i=1}^{k} \left[C \times g_i(\cdot) \times (1 - V_i(p_i^*)) \times p_i^*\right]. \tag{9}$$

3) Cost Minimization for Each Model Version: After determining the revenue R_k for model version k, we need to know the lowest cost of incentivizing a proper $\mathcal{W}$ of data owners to train a model no worse than version k. To this end, we first present the model quality estimation through convergence analysis of model training. Then, we formulate the cost minimization problem under model quality constraints.

Model Quality Estimation: We conduct convergence analysis of model training via DPFL to characterize how privacy budgets of data owners in $\mathcal{W}$ affect the quality of the trained model. We start with some typical assumptions [20], [21].

Assumption 1 (Lipschitz-smooth loss function [44])**.** *The loss function $f(\boldsymbol{\theta})$ is L-Lipschitz smooth with constant L, i,e.,*

$$\|\nabla f(\boldsymbol{\theta}_1) - \nabla f(\boldsymbol{\theta}_2)\|_2 \leq L\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2 \ \forall \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d. \tag{10}$$

Assumption 2 (First and second moment limits [44])**.** *For all $t \in \mathbb{N}$, the loss function satisfies $f(\boldsymbol{\theta}^t) \geq f^*$ and the stochastic gradients at each data owner $w \in \mathcal{W}$ satisfy:*

$$\nabla f(\boldsymbol{\theta}^t)^T \mathbb{E}[g(\mathcal{B}_w; \boldsymbol{\theta}^t)] \geq \mu \|\nabla f(\boldsymbol{\theta}^t)\|_2^2, \tag{11}$$

$$\|\mathbb{E}[g(\mathcal{B}_w; \boldsymbol{\theta}^t)]\|_2 \ \leq \mu_G \|\nabla f(\boldsymbol{\theta}^t)\|_2, \tag{12}$$

$$\mathbb{V}[g(\mathcal{B}_w; \boldsymbol{\theta}^t)] \leq M + M_V \|\nabla f(\boldsymbol{\theta}^t)\|_2^2, \tag{13}$$

where $\mu_G \geq \mu > 0$, $M \geq 0$, and $M_V \geq 0$ are constants.

We present the convergence result of non-convex loss functions in Theorem 1.[1] The proof leverages Assumptions 1 and 2, Lemma 2 and follows the rationale of Section 4 [44] (replacing the raw stochastic gradient there with the *aggregated perturbed* stochastic gradients here and adding more involved analysis).

Theorem 1. *Under Assumptions 1 and 2, suppose that the global model $\boldsymbol{\theta}$ is updated according to Eq. (6) with a fixed learning rate η satisfying $0 < \eta \leq \frac{\mu}{L(\mu_G^2 + M_V)}$. Then, the expected average-squared ℓ_2-norm of gradients of $f(\cdot)$ over T rounds of updates satisfies*

$$\mathbb{E}[\frac{1}{T}\sum_{t=0}^{T-1} \|\nabla f(\boldsymbol{\theta}^t)\|_2^2] \leq \frac{2(f(\boldsymbol{\theta}^0) - f^*)}{T\eta\mu} + \Phi, \tag{14}$$

where $\Phi = \frac{L\eta(W^2 M + 2dL^2/B_w^2 \sum_{w\in\mathcal{W}} 1/\rho_w)}{\mu W^2}$.

[1] We would like to stress that the insights behind the result are also applicable to convex loss functions.

Theorem 1 shows that the quality of a trained model is determined by the number of involved data owners W and their corresponding privacy budgets ρ_w.[2] Specifically, the model quality is indicated by $\psi = (\sum_{w\in\mathcal{W}} \frac{1}{\rho_w})/W^2$. A smaller value of ψ indicates a better model quality with a high probability.

To meet the quality requirement Q_k of each model version k, we could resort to enforcing a constraint on the value of ψ, say ψ_k. Specifically, we need to recruit a set $\mathcal{W}$ of data owners such that $\psi = (\sum_{w\in\mathcal{W}} \frac{1}{\rho_w})/W^2 \leq \psi_k$. Obviously, a higher model version k corresponds to a lower value of ψ_k (i.e., $\psi_k > \psi_{k+1}$ for $k = 1, \ldots, K-1$).

Cost Minimization Formulation: Now, we formally formulate the DPFL model training cost minimization problem for each model version k. The problem is to select a set $\mathcal{W}$ of W data owners that could train an ML model with a quality no worse than Q_k (i.e., $(\sum_{w\in\mathcal{W}} \frac{1}{\rho_w})/W^2 \leq \psi_k$) with the minimum model training cost. Here, the model training cost refers to the total payment for data owners in $\mathcal{W}$, i.e., $\sum_{w\in\mathcal{W}} q_w$ (with q_w denoting the payment for data owner w trying to cover her privacy cost c_w). Using x_n to denote the selection result of data owner $n \in \mathcal{N}$, where $x_n = 1$ represents she is selected, and $x_n = 0$ otherwise, we can formulate the training cost minimization problem for version k as follows.

Problem 2. *Cost Minimization for Model Version k.*

$$\min \quad \sum_{n\in\mathcal{N}} q_n x_n \tag{15a}$$

$$s.t. \quad \frac{\sum_{n\in\mathcal{N}} \frac{x_n}{\rho_n}}{\left(\sum_{n\in\mathcal{N}} x_n\right)^2} \leq \psi_k, \tag{15b}$$

$$\sum_{n\in\mathcal{N}} x_n \geq G, \tag{15c}$$

$$var: \quad x_n \in \{0,1\}, \forall n \in \mathcal{N}.$$

In Problem 2, the objective function (15a) is to minimize the model training cost (note that the parameters q_n have different realizations in different scenarios, and here we just consider them as parameters). Constraint (15b) enforces the model quality requirement. Besides, the broker sets a minimum required number G of selected data owners (as in Constraint (15c)) to avoid a few malicious data owners manipulating the FL model training process (e.g., conducting Byzantine attacks [45]). We can see that Problem 2 is a $0-1$ integer non-convex optimization problem, which is NP-complete and computationally intractable. We thus focus on finding the near-optimal solution.

VI. Algorithm Design under Complete Information

In this section, we design an efficient algorithm (named *DEVELOP-C*) to solve Problem 2 in a benchmark case of complete information, where the broker knows each data owner's privacy cost beforehand.[3]

[2]We set the mini-batch size B_w as a constant across communication rounds and data owners. Besides, other system parameters, which are not related to data owners, are constants as well.

Algorithm 1: Greedy Data Owner Selection (*GDOS*)

Input: Privacy budget $\boldsymbol{\rho} = (\rho_n, n \in \mathcal{N})$, privacy cost $\boldsymbol{c} = (c_n, n \in \mathcal{N})$, the model quality requirement ψ_k, the minimum required number of selected data owners G.

Output: The selected data owner set $\mathcal{W}$.

1 $\mathcal{W} = \emptyset$, the model quality indicator $\psi = 0$, the number of selected data owners $t = 0$;
2 Calculate the privacy cost per unit privacy budget for each data owner, i.e., $v_n = c_n/\rho_n, \forall n \in \mathcal{N}$;
3 Sort data owners in $\mathcal{N}$ in ascending order of v_n, i.e., $c_1/\rho_1 \leq c_2/\rho_2 \leq \ldots \leq c_N/\rho_N$;
4 **for** $t < G$ **do**
5 $\quad t = t+1$; $\mathcal{W} = \mathcal{W} \cup t$; $\psi = \psi + \frac{1}{\rho_t}$;
6 **end**
7 **while** $\psi/(t^2) > \psi_k$ *and* $t < N$ **do**
8 $\quad t = t+1$; $\mathcal{W} = \mathcal{W} \cup t$; $\psi = \psi + \frac{1}{\rho_t}$;
9 **end**
10 Return $\mathcal{W}$;

A. Algorithm Design

We first elaborate on our proposed algorithms in this subsection. With the information of each data owner's privacy cost c_n, the broker can pay her c_n. That is, the parameter q_n in Problem 2 is exactly c_n. Since we have demonstrated that a larger privacy budget helps improve the model quality, one heuristic solution to Problem 2 is to greedily select data owners with larger privacy budgets and less privacy costs. Following this rule, we design a greedy data owner selection algorithm (named *GDOS*), as described in Algorithm 1.

Workflow of Algorithm 1: After calculating the privacy cost per unit privacy budget v_n for each data owner $n \in \mathcal{N}$ (Line 2) and sorting data owners in ascending order of v_n (Line 3), Algorithm 1 selects G data owners with the smallest v_n (Lines $4-6$). Then, it selects data owners one by one along the sequence until the desired model quality is reached or all N data owners are already selected (Lines $7-9$).

Combining revenue maximization and approximate cost minimization via *GDOS*, we now describe how to determine the maximum approximate profit in Algorithm 2.

Workflow of Algorithm 2: Algorithm 2 first determines the optimal price p_k^* for each model version k (Lines $2-4$). Then, it repeats the following procedures for each model version $k = 1, \ldots, K$:

- Calculate the maximum expected revenue R_k using the derived optimal prices (Line 6);

[3]Note that the broker could detect the noise level of gradient perturbation (and then the privacy budget) of each data owner by installing a specialized app on their mobile devices, which automatically injects noises and inputs training data [35].

Algorithm 2: Profit Maximization for the Broker

Input: A set $\mathcal{C} = \{1, \ldots, C\}$ of model buyers with valuations $\boldsymbol{V}$ and version preferences $g_k(\cdot)$ ($k \in \mathcal{K}$), privacy budget $\boldsymbol{\rho} = (\rho_n, n \in \mathcal{N})$, privacy cost $\boldsymbol{c} = (c_n, n \in \mathcal{N})$, the model quality requirement vector $\boldsymbol{\Psi} = (\psi_1, \ldots, \psi_K)$, the minimum required number of selected data owners G.

Output: The maximum approximate profit Φ^* and the corresponding optimal model version k^*.

1 $k^* = 0$, $\Phi^* = 0$;
2 **for** $k = 1$ *to* K **do**
3 $\quad p_k^* \leftarrow \arg\max_{p_k} [(1 - V_k(p_k)) \times p_k]$
4 **end**
5 **for** $k = 1$ *to* K **do**
6 $\quad R_k = \sum_{i=1}^{k} [C \times g_i(\cdot) \times (1 - V_i(p_i^*)) \times p_i^*]$;
7 $\quad \mathcal{W} \leftarrow$ *GDOS*$(\boldsymbol{\rho}, \boldsymbol{c}, \psi_k, G)$;
8 $\quad S(\mathcal{W}) = \sum_{w \in \mathcal{W}} c_w$;
9 $\quad \Phi_k \leftarrow R_k - S(\mathcal{W})$;
10 $\quad$ **if** $\Phi_k > \Phi^*$ **then**
11 $\quad\quad \Phi^* \leftarrow \Phi_k$; $k^* \leftarrow k$;
12 $\quad$ **end**
13 **end**
14 Return Φ^* and k^*;

- Calculate the minimum approximate cost via *GDOS* (Lines $7-8$);
- Calculate the approximate profit Φ_k (Line 9).

Finally, among K model versions, Algorithm 2 chooses the maximum approximate profit and the corresponding version as the final output (Lines $10-12$).

B. Performance Analysis

In this subsection, we analyze the approximation ratio of Algorithm 2. Define $c_{max} = \max_{n \in \mathcal{N}} c_n$ and $c_{min} = \min_{n \in \mathcal{N}} c_n$. For each model version k, denote the optimal solution to Problem 2 and its corresponding profit by $\mathcal{W}_k^*$ and Φ_k^*, respectively; define $\zeta_k = R_k / S(\mathcal{W}_k^*)$; denote the number of selected data owners via Algorithm 1 by W_k. Define the maximum profit as $OPT = \max_{k=1,\ldots,K} \Phi_k^*$, and we can derive the approximation ratio of Algorithm 2 in Theorem 2. We omit the proof due to the page limit.

Theorem 2. *The maximum approximate profit Φ^* of Algorithm 2 satisfies*

$$\frac{\Phi^*}{OPT} \geq \min_{k=1,\ldots,K} \left\{ \frac{\zeta_k - \frac{W_k c_{max}}{G c_{min}}}{\zeta_k - 1} \right\}. \tag{16}$$

Theorem 2 shows that the profit approximation ratio is related to the distribution of data owners' privacy costs and the system parameter G. We demonstrate in our numerical experiments that our derived maximum approximate profit is close to that obtained by the optimal solution.

VII. Mechanism Design under Incomplete Information

In this section, we consider a more realistic scenario of incomplete information, where the broker does not know each data owner's privacy cost. We design a reverse auction-based mechanism (named *DEVELOP-IC*) to incentivize data owners' truthful reporting of privacy costs.

A. Mechanism Design

We model the interaction between data owners and the broker as a multi-item single-unit reverse auction, where the broker acts as an auctioneer or a buyer who wants to buy the model training services from competitive data owners. The workflow of the reverse auction is described as follows:

1) **Task Announcement:** The broker announces a model training task to data owners.
2) **Bidding:** Each data owner $n \in \mathcal{N}$ reports a bid b_n of privacy cost c_n based on her consumed privacy budget ρ_n and privacy preference α_n (for example, $c_n = \alpha_n \rho_n$ if she adopts the linear cost function). Note that rational and selfish data owners would misreport their privacy cost (i.e., $b_n \neq c_n$) if doing so can bring them more utility.
3) **Winner Determination and Payment Calculation:** Given data owners' bid profile $\boldsymbol{b} = (b_n, n \in \mathcal{N})$ and privacy budget $\boldsymbol{\rho} = (\rho_n, n \in \mathcal{N})$, the broker runs the auction to derive an allocation result $\boldsymbol{x} = (x_n, n \in \mathcal{N})$, where $x_n = 1$ indicates data owner n wins the auction, and $x_n = 0$ otherwise. Notice $\boldsymbol{x} = (x_n, n \in \mathcal{N})$ also implies winner set $\mathcal{W}$. Besides, the auction determines each data owner's payment $\boldsymbol{q} = (q_n, n \in \mathcal{N})$. The utility of data owner n is then computed as $u_n = q_n - x_n c_n$.

Generally, the reverse auction needs to satisfy a set of desiderata listed as below.

- **Incentive Compatibility (IC):** to prevent strategic data owners from untruthfully reporting their privacy costs to strive for overvalued compensation.
- **Individual Rationality (IR):** to ensure all data owners obtain non-negative utility and thus be willing to participate in model training via DPFL.
- **Computational Efficiency (CE):** to ensure the broker runs the auction mechanism efficiently.

Different from other reverse auction design which generally operates under a simple financial budget constraint (e.g., [33], [35], [46]), *the complicated model quality constraint related to both the number of winners and their heterogeneous privacy budgets makes our auction design quite challenging*. Specifically, in our reverse auction, it is indeterminate how the model quality evolves with the inclusion or exclusion of data owners. Thus, it is challenging to determine the critical bid and then the payment to data owners.

Our proposed *DEVELOP-IC* first employs *GDOS* (replacing data owners' privacy costs $\boldsymbol{c}$ with their bids $\boldsymbol{b}$) to determine the winner set $\mathcal{W}$ (Line 2). Then, to determine the critical bid in order to calculate the payment to data owners, *DEVELOP-IC* identifies the maximum number Z of data owners needed

Algorithm 3: *DEVELOP-IC*

Input: Bid $\boldsymbol{b} = (b_n, n \in \mathcal{N})$, privacy budget $\boldsymbol{\rho} = (\rho_n, n \in \mathcal{N})$, the model quality requirement ψ_k, the minimum required number of winners G.

Output: Winner set $\mathcal{W}$, allocation result $\boldsymbol{x} = (x_n, n \in \mathcal{N})$, payment $\boldsymbol{q} = (q_n, n \in \mathcal{N})$.

1 **Winner Determination:**
2 $\mathcal{W} = GDOS(\boldsymbol{\rho}, \boldsymbol{b}, \psi_k, G)$;
3 **Payment Calculation:**
4 The model quality indicator $\psi = 0$, the number of selected data owners $Z = 0$;
5 Sort data owners in $\mathcal{N}$ in ascending order of ρ_n, i.e., $\rho_1 \leq \rho_2 \leq \ldots \leq \rho_N$;
6 **do**
7 $\quad Z = Z + 1$; $\psi = \psi + \frac{1}{\rho_Z}$;
8 **while** $\psi/(Z^2) > \psi_k$ *and* $Z < N$;
9 Sort data owners in $\mathcal{N}$ in ascending order of b_n/ρ_n, i.e., $b_1/\rho_1 \leq b_2/\rho_2 \leq \ldots \leq b_N/\rho_N$;
10 **for** *each data owner* $n \in \mathcal{N}$ **do**
11 $\quad$ **if** $n \in \mathcal{W}$ **then**
12 $\quad\quad q_n = \frac{\rho_n}{\rho_{Z+1}} b_{Z+1}$; $x_n = 1$;
13 $\quad$ **else**
14 $\quad\quad q_n = 0$; $x_n = 0$;
15 $\quad$ **end**
16 **end**
17 Return $\mathcal{W}$, $\boldsymbol{q}$, and $\boldsymbol{x}$;

to train an ML model with desired quality (Lines $4-8$). The critical bid is determined as the $(Z+1)$-th value in the sequence of sorted data owners' bid per unit privacy budget b_n/ρ_n. Finally, *DEVELOP-IC* calculates the payment q_n to each data owner $n \in \mathcal{N}$ using the critical bid as the benchmark (Lines $10-16$). Intuitively, a data owner n with a larger privacy budget ρ_n will receive higher payment as compensation.

B. Theoretical Analysis

In this subsection, we prove that *DEVELOP-IC* satisfies CE, IC, and IR. First, by inspecting Algorithm 3, we can see

Theorem 3. *DEVELOP-IC has a complexity of* $O(N \log N)$.

Theorem 3 shows that *DEVELOP-IC* runs in polynomial time, and thus is computationally efficient.

According to Myerson's Lemma [47], a mechanism with a monotone allocation rule and critical payment is incentive compatible. In *DEVELOP-IC*, we use greedy data owner selection procedures to ensure monotonicity. Then, we leverage the bid per unit privacy budget associated with index $(Z+1)$ to characterize the critical payment. Thus, we can show

Theorem 4. *DEVELOP-IC is incentive compatible.*

Our payment determination ensures non-negative utility of the selected data owners. Thus,

Theorem 5. *DEVELOP-IC is individually rational.*

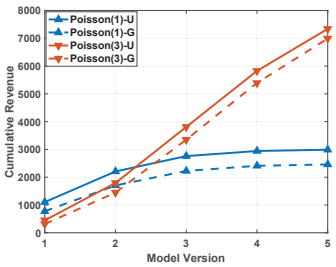

Fig. 2. Revenue versus model version.

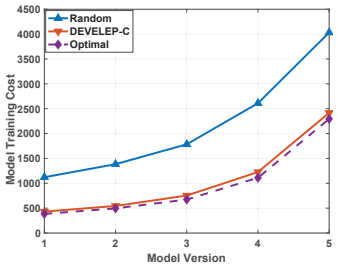

Fig. 3. Cost versus model version.

VIII. Numerical Results

In this section, we provide numerical results to validate the effectiveness of the proposed algorithms.

A. Experimental Setup

1) Setup for Model Marketplace: Without loss of generality, we set the number of model versions as $K = 5$, and the corresponding quality vector $\boldsymbol{Q} = [\psi_1, \psi_2, \psi_3, \psi_4, \psi_5] = [0.5, 0.4, 0.3, 0.2, 0.1]$. Note that a lower value of ψ_k indicates a higher model version (i.e., a better model quality).

2) Setup for Model Buyers: We assume that there are $C = 1000$ model buyers in the market, and consider different distributions of model version preferences and valuations to characterize their different purchase behaviors. Specifically, following [41], we employ the *Poisson* distribution with two choices of parameter λ (i.e., $\lambda = 1, 3$) as the version preference distribution, and utilize the following two distributions to simulate buyers' valuations on model version k.

- *Uniform Distribution:* $\mathbb{U}(3k, 5k)$,
- *Truncated Gaussian Distribution:* $\mathbb{N}(4k, 4)$ with the negative tail truncated.

3) Setup for Data Owners: We assume there are $N = 50$ data owners in the market. The privacy budget ρ_n and privacy preference α_n of data owner $n \in \mathcal{N}$ respectively follow the uniform distributions of $\mathbb{U}(0.5, 1.5)$ and $\mathbb{U}(200, 300)$. Each data owner n randomly selects one from the following four cost functions to calculate her privacy cost c_n.

- linear function ($c_n^L = \alpha_n \rho_n$);
- quadratic function ($c_n^Q = \alpha_n \rho_n^2$);
- square root function ($c_n^R = \alpha_n \sqrt{\rho_n}$);
- exponential function ($c_n^E = \alpha_n (e^{\rho_n} - 1)$).

B. Performance under Complete Information

1) Revenue from Model Trading: Fig. 2 shows the maximum expected accumulative revenue of each model version, under different distributions of buyers' version preferences and valuations. Note that hereafter we use "Poisson(λ)-U(G)" to represent that buyers' version preferences follow a Poisson distribution with parameter λ and the valuations follow uniform (Gaussian) distribution. We can see that the revenue increases with a higher version regardless of buyers' purchase behaviors.

2) Cost of Model Training: We now evaluate the cost of training different versions of ML models. We compare the proposed algorithm with the optimal solution (denoted by "Optimal") and the random solution (denoted by "Random"). Specifically, "Optimal" finds the global minima using

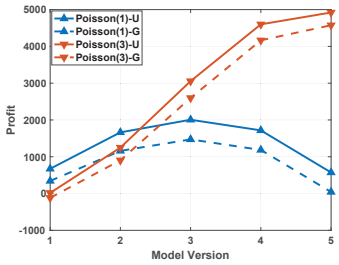

Fig. 4. Profit versus model version under complete information.

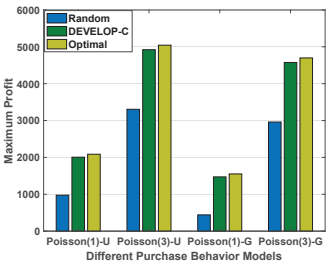

Fig. 5. Profit comparison.

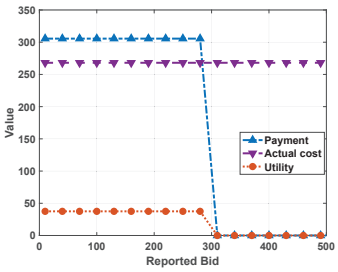

Fig. 6. Incentive compatibility.

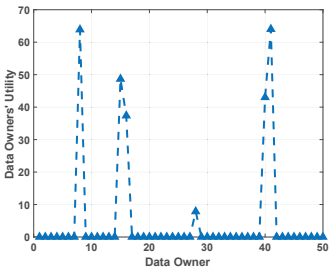

Fig. 7. Individual rationality.

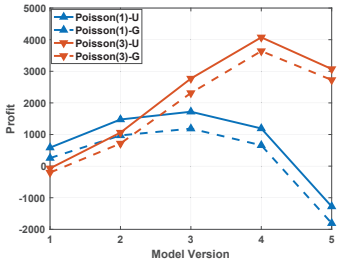

Fig. 8. Profit versus model version under incomplete information.

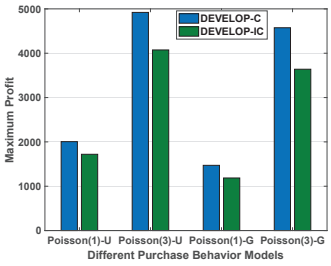

Fig. 9. Profit comparison between complete/incomplete information.

a mathematical optimization solver called Gurobi [48], while "Random" seeks the solution via randomly selecting data owners (without replacement) successively until reaching a certain termination condition (i.e., meeting the quality requirement). Fig. 3 shows that *our proposed algorithm is near-optimal and significantly outperforms "Random"*. Specifically, our proposed algorithm achieves less than 13% higher cost compared to "Optimal". Meanwhile, compared to "Random", our proposed algorithm reduces the cost by up to 61%.

3) Profit Evaluation: We show the profit versus model version in Fig. 4. We observe that the distributions of model buyers' version preferences significantly impact the profit. Specifically, when more model buyers tend to purchase a higher version (i.e., $\lambda = 3$ here), a higher model version always creates a larger profit. In contrast, when $\lambda = 1$, meaning that more model buyers are interested in lower versions, such monotonicity vanishes. Specifically, under "Poisson(1)-(G)", training and selling model version 5 leads to 97% profit loss compared to the largest profit achieved by version 3. This highlights the importance of identifying the optimal model version to train and trade.

We compare the broker's profit among *DEVELOP-C*, "Random", and "Optimal" under four different buyers' purchase behaviors in Fig. 5. The results demonstrate that *DEVELOP-C* significantly outperforms "Random" and achieves comparable profit (up to 97.6%) to "Optimal". From Fig. 5, we further observe that the broker earns more profit under "Poisson(1)-U" ("Poisson(3)-U") than "Poisson(1)-G" ("Poisson(3)-G"). The insight is that *given identical mean value of model buyers' valuations on each model version k ($\mathbb{U}(3k, 5k)$ and $\mathbb{N}(4k, 4)$), a dispersed distribution of valuations is more profitable for the broker in comparison to a concentrated one.*

C. Performance under Incomplete Information

1) Economic Properties: We first show two economic properties of *DEVELOP-IC*, i.e., incentive compatibility and individual rationality. First, we randomly select a data owner whose actual privacy cost is 268.1 and present her payment and utility versus the bid value in Fig. 6. From Fig. 6, we see the critical payment is around 305.6, and no matter how the data owner changes her bid, she either maintains the fixed utility of around 37.5 as a winner or gets zero utility. Thus, she has no incentive to misreport her privacy cost, which guarantees the incentive compatibility. We further present the utility of data owners in Fig. 7, where all of them obtain non-negative utility, validating the individual rationality.

2) Profit Evaluation: Fig. 8 characterizes the relationship between the profit and model version under *DEVELOP-IC*. We find that *the highest model version could be less profitable or even cause a deficit (i.e., a negative profit) for the broker*. Specifically, blindly training and selling the highest model version rather than the optimal version (i.e., version 3) causes a profit variation from $+1186$ to -1807 for the broker under "Poisson(1)-G". Again, this result underlines the significance of conscientious market research before transactions and judiciously determining the optimal model version to create.

3) Profit Comparison Between Complete and Incomplete Information: Finally, we compare the brokers' profit between *DEVELOP-IC* and *DEVELOP-C* under different model buyers' purchase behaviors in Fig. 9. Obviously, in call cases, *DEVELOP-IC* achieves less profit than *DEVELOP-C* (14.2%, 17.2%, 19.4%, and 20.5% in order), as under incomplete information, the broker needs to address data owners' strategic behaviors and disburses more for model training.

IX. Conclusion

This paper built the first DPFL-based ML model marketplace. We focus on profit maximization for the broker. This problem is challenging due to the significant difficulties in characterizing the revenue function and estimating the DPFL model training cost. We propose a two-layer optimization framework to address it, i.e., revenue maximization and cost minimization under a model quality constraint. We design efficient algorithms to solve the cost minimization problem under both complete and incomplete information scenarios, and their performances are theoretically guaranteed and empirically validated. In our future work, we will further study how model buyers' strategic behaviors (e.g., arbitrage) affect the ML model marketplace design.

REFERENCES

[1] J. Pei, "A survey on data pricing: from economics to data science," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[2] L. Chen, P. Koutris, and A. Kumar, "Towards model-based pricing for machine learning in a data marketplace," in *International Conference on Management of Data*, 2019, pp. 1535–1552.

[3] J. Liu, J. Lou, J. Liu, L. Xiong, J. Pei, and J. Sun, "Dealer: an end-to-end model marketplace with differential privacy," *Proceedings of the VLDB Endowment*, vol. 14, no. 6, pp. 957–969, 2021.

[4] A. Agarwal, M. Dahleh, and T. Sarkar, "A marketplace for data: An algorithmic solution," in *ACM Conference on Economics and Computation*, 2019, pp. 701–726.

[5] [Online]. Available: https://www.modzy.com/marketplace/

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[7] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2020.

[8] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[9] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, 2019.

[10] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning design," in *IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[11] P. Sun, H. Che, Z. Wang, Y. Wang, T. Wang, L. Wu, and H. Shao, "PAIN-FL: Personalized privacy-preserving incentive for federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3805–3820, 2021.

[12] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning in mobile edge networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3606–3621, 2021.

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

[14] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *IEEE Symposium on Security and Privacy*, 2019, pp. 691–706.

[15] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE Symposium on Security and Privacy*, 2017, pp. 3–18.

[16] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *IEEE Conference on Computer Communications*, 2019, pp. 2512–2520.

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

[18] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 774–14 784.

[19] A. Triastcyn and B. Faltings, "Federated learning with bayesian differential privacy," in *IEEE International Conference on Big Data*, 2019, pp. 2587–2596.

[20] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, 2020.

[21] K. Wei, J. Li, M. Ding, C. Ma, H. Su, B. Zhang, and H. V. Poor, "User-level privacy-preserving federated learning: Analysis and performance optimization," *IEEE Transactions on Mobile Computing*, 2021.

[22] R. Hu, Y. Guo, H. Li, Q. Pei, and Y. Gong, "Personalized federated learning with differential privacy," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9530–9539, 2020.

[23] M. Kim, O. Günlü, and R. F. Schaefer, "Federated learning with local differential privacy: Trade-offs between privacy, utility, and communication," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2021, pp. 2650–2654.

[24] R. Hu and Y. Gong, "Trading data for learning: Incentive mechanism for on-device federated learning," in *IEEE Global Communications Conference*, 2020, pp. 1–6.

[25] M. Wu, D. Ye, J. Ding, Y. Guo, R. Yu, and M. Pan, "Incentivizing differentially private federated learning: A multi-dimensional contract approach," *IEEE Internet of Things Journal*, 2021.

[26] A. Kumar, B. Finley, T. Braud, S. Tarkoma, and P. Hui, "Marketplace for AI models," *arXiv preprint arXiv:2003.01593*, 2020.

[27] S. R. Pandey, N. H. Tran, M. Bennis, Y. K. Tun, A. Manzoor, and C. S. Hong, "A crowdsourcing framework for on-device federated learning," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3241–3256, 2020.

[28] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet of Things Journal*, 2020.

[29] N. Ding, Z. Fang, and J. Huang, "Optimal contract design for efficient federated learning with multi-dimensional private information," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 186–200, 2020.

[30] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 700–10 714, 2019.

[31] T. H. T. Le, N. H. Tran, Y. K. Tun, Z. Han, and C. S. Hong, "Auction based incentive design for efficient federated learning in cellular wireless networks," in *IEEE Wireless Communications and Networking Conference*, 2020, pp. 1–6.

[32] Y. Jiao, P. Wang, D. Niyato, B. Lin, and D. I. Kim, "Toward an automated auction framework for wireless federated learning services market," *IEEE Transactions on Mobile Computing*, 2020.

[33] J. Zhang, Y. Wu, and R. Pan, "Incentive mechanism for horizontal federated learning based on reputation and reverse auction," in *The Web Conference*, 2021, pp. 947–956.

[34] S. Wei, Y. Tong, Z. Zhou, and T. Song, "Efficient and fair data valuation for horizontal federated learning," in *Federated Learning*. Springer, 2020, pp. 139–152.

[35] S. Zheng, Y. Cao, and M. Yoshikawa, "Incentive mechanism for privacy-preserving federated learning," *arXiv preprint arXiv:2106.04384*, 2021.

[36] H. Gao, A. Xu, and H. Huang, "On the convergence of communication-efficient local SGD for federated learning," in *AAAI Conference on Artificial Intelligence*, vol. 35, no. 9, 2021, pp. 7510–7518.

[37] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy." *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.

[38] M. Bun and T. Steinke, "Concentrated differential privacy: Simplifications, extensions, and lower bounds," in *Theory of Cryptography Conference*. Springer, 2016, pp. 635–658.

[39] C. Shapiro and H. R. Varian, "Versioning: the smart way to sell information," *Harvard Business Review*, vol. 107, no. 6, p. 107, 1998.

[40] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *IEEE International Conference on Computer Vision*, 2017, pp. 1389–1397.

[41] Z. Zheng, Y. Peng, F. Wu, S. Tang, and G. Chen, "Trading data in the crowd: Profit-driven data acquisition for mobile crowdsensing," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 2, pp. 486–501, 2017.

[42] G. Koop, D. J. Poirier, and J. L. Tobias, *Bayesian econometric methods*. Cambridge University Press, 2007.

[43] A. Ghosh and A. Roth, "Selling privacy at auction," *Games and Economic Behavior*, vol. 91, pp. 334–346, 2015.

[44] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.

[45] M. Fang, X. Cao, J. Jia, and N. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in {*USENIX*} *Security Symposium*, 2020, pp. 1605–1622.

[46] W. Jin, M. Xiao, M. Li, and L. Guo, "If you do not care about it, sell it: Trading location privacy in mobile crowd sensing," in *IEEE Conference on Computer Communications*, 2019, pp. 1045–1053.

[47] R. B. Myerson, "Optimal auction design," *Mathematics of Operations Research*, vol. 6, no. 1, pp. 58–73, 1981.

[48] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2021. [Online]. Available: https://www.gurobi.com