Privacy Budgeting for Growing Machine Learning Datasets

Weiting Li*, Liyao Xiang*, Zhou Zhou*, Feng Peng*
* John Hopcroft Center for Computer Science, Shanghai Jiao Tong University, Shanghai, China

IEEE INFOCOM 2021 - IEEE Conference on Computer Communications | 978-1-6654-0325-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/INFOCOM42981.2021.9488920

***Abstract*—The wide deployment of machine learning (ML) models and service APIs exposes the sensitive training data to untrusted and unknown parties, such as end-users and corporations. It is important to preserve data privacy in the released ML models. An essential issue with today's privacy-preserving ML platforms is a lack of concern on the tradeoff between data privacy and model utility: a private datablock can only be accessed a finite number of times as each access is privacy-leaking. However, it has never been interrogated whether such privacy leaked in the training brings good utility. We propose a differentially-private access control mechanism on the ML platform to assign datablocks to queries. Each datablock arrives at the platform with a privacy budget, which would be consumed at each query access. We aim to make the most use of the data under the privacy budget constraints. In practice, both datablocks and queries arrive continuously so that each access decision has to be made without knowledge about the future. Hence we propose online algorithms with a worst-case performance guarantee. Experiments on a variety of settings show our privacy budgeting scheme yields high utility on ML platforms.**

***Index Terms*—Differential privacy, online algorithm, machine learning**

I. Introduction

Fed with abundant data, today's machine learning services have been widely deployed, in areas such as auto-driving, objective detection, machine translation, product recommendation, and others. Recently, the privacy leakage of these services has raised extensive concerns — it has been found the trained machine learning models, being directly or indirectly accessed, reveal an unexpected amount of information of the training data. A disease prediction model may reveal if your medical records have participated its training. Your facial profile may be reconstructed by inverting a face recognition engine.

To protect sensitive training data, differential privacy [1] is adopted as a conventional approach, and has been integrated as a built-in feature on various machine learning platforms such as `TensorFlow Privacy` [2], `PySyft` [3]. By introducing controlled randomness to the training data, or the training process, differential privacy mechanisms [4] guarantee the resulting models or prediction outcomes to be privacy-preserving, *i.e.,* given the output, it is indistinguishable to tell which input distribution that the output comes from.

Liyao Xiang (xiangliyao08@sjtu.edu.cn) is the corresponding author.

This work was partially supported by NSF China (61902245, 62032020, 61960206002, 61822206, 62020106005, 61829201, 62041205, 61532012), National Key R&D Program of China 2018YFB1004700, and the Science and Technology Innovation Program of Shanghai (19YF1424500).

Although privacy mechanisms are implanted to ML platforms [5], they are at the primitive stage: they rarely deal with growing data, and hardly take utility into account by design. Conventional privacy-preserving mechanisms [6], [7] are mainly proposed on static datasets, of which the privacy budget quickly drains with each privacy-leaking access. Sage [8] avoids the problem by accounting the privacy loss on each datablock, rather than on the entire dataset. The privacy budget never runs out on the growing dataset as each new datablock arrives with a fresh privacy budget. However, it remains a question whether the privacy budget is well-spent. Since each private datablock restricts the number of accesses, privacy budgets are valuable resources to be carefully allocated.

We devise a new access control mechanism on ML platforms to inspect the assignment of datablocks to queries. Rather than *accounting* the privacy loss in a posterior way, we *budget* the privacy to generate the most utility. Considering that we have multiple training models on the platform, a datablock may produce a highly accurate result on one but hardly improves the accuracy of another. With datablocks' privacy budget as constraints, we formulate the problem as an optimization one that seeks an assignment to maximize utility. The proposed mechanism prevents a private datablock from serving queries which it brings marginal benefit to. However, the approach faces two difficulties: the value of a datablock to a query cannot be known in prior, and such a value is privacy-leaking. What's worse, with datablocks and queries arriving continuously, an optimal decision can hardly be made without any foresight.

To resolve the first issue, we propose the concept of data-query significance to describe the value of a datablock to a query. We found that it is not the computation of the significance value, but the comparison of such values is privacy-leaking: it would harm the privacy of a datablock if the block is known to be important to a particular query. Hence we introduce a differentially-private filtering scheme to select datablocks with high significance. Essentially, we sacrifice a small portion of the privacy budget to understand the datablocks, and to make wise use of the privacy budget.

For the second issue, we propose online algorithms with a provable performance guarantee for our assignment problem. Note that our problem resembles, but is unique to the multiple knapsack problem. As queries arrive in an online fashion, they first obtain their data-query significance at some mild privacy cost. Our online algorithms admit a query to access a datablock only if its significance surpasses a threshold. If queries require

more than a datablock could afford, we make a compromise to allow a datablock to partially fulfill the query rather than turning it down. We have provided proofs for the competitive ratios that our online algorithms achieve.

Highlights of our contribution are as follows. First, we introduce an access control mechanism for the privacy-preserving ML platforms: it monitors the consumption of the privacy budget of each datablock, and assigns datablocks to queries to yield high utility. Second, we propose a differentially-private filtering scheme to select queries with high data-query significance, providing insight for assignment. Third, in the case where datablocks and queries arrive continuously, we give two online algorithms with the worst-case performance guaranteed. Experimental results on the ML platforms show our access control mechanism brings notable utility improvement under the same privacy budget constraint.

II. Related Work

Our work is mainly related to the following literature.

A. Differential Privacy for Growing Databases

Streaming differential privacy [9], [10] extends the privacy-preserving target from static databases to growing ones. The database is a stream where the datablocks arrive and leave at all times. Each datablock can only be used once. We consider it is wasteful of datablocks in most cases, particularly in machine learning, as a datablock could be repeatedly queried for training. Thus the privacy accounting for streaming data does not fit our scenario. [11] studies a specific class of problem, running predicate sums, based on the notions of decayed data and privacy expiration, whose situation is more similar to ours while still limited by the setting of streaming data.

Differential privacy for growing databases [12] transforms private and accurate algorithms for static databases to private and accurate algorithms for dynamically growing databases. However, their technique suits better to cases where a static database could have been used, but is not designed for an inherently dynamic database. Real-world machine learning algorithms typically run on growing databases and thus may not be a good target for deploying their technique. Moreover, databases facing the issue of running out of the privacy budget renders the technique impractical.

Sage [8] proposes a block composition scheme to resolve the issue of draining the privacy budget. The new privacy loss accounting method leverages the growing database regime to keep training models endlessly on a sensitive data stream. However, its access control module lacks an understanding of the datablocks and merely assigns datablocks to queries as if they are specifically asked for. In practice, a query rarely asks for a specific training datablock, due to a lack of understanding of them. Besides, a random selection of datablocks to answer queries would incur low utilization of data. In our work, we propose a differentially-private mechanism on growing datasets for training ML models, by taking utility into account.

B. Utility-Aware Privacy Budgeting

Our work is not the only one that tries to distribute the limited privacy budget with the consideration of utility. Cuppens *et al.* [13] consider the contextual risk evaluation for queries to distribute the budget. Xiang *et al.* [14] formulate the differentially-private SGD as a privacy-constrained optimization problem to seek an optimized noise distribution w.r.t. the model utility. To reduce the impact of randomized noise, Kellaris *et al.* [15] propose to treat similar data as one object, and group a set of privacy budgets for this object so that less noise is added to the data for guaranteeing the same level of differential privacy. But this similarity measurement only works on sparse datasets where most data is similar. For highly heterogeneous data, their measure may not work well. Our idea shares the same spirit with [15] in comparing the data quality but is widely applicable to data of any distribution.

III. Preliminaries

We introduce some background for a better understanding of the paper.

A. Differential Privacy and Composition

Differential privacy is proposed to constrain an attacker's capability to gain additional knowledge about a particular data record despite that it is in the dataset or not [1]. The privacy guarantee is expressed by the logarithmic distance between the posterior probability distributions of two adjacent inputs given the outputs. Adjacent inputs are two datasets differed by one unit of distance. Different metrics of the distance can be used, which leads to different variants of differential privacy. We use ϵ to define the upper bound of the distribution distance and δ to denote the residual probability. Formally, we have

Definition 1 ((ϵ, δ)-Differential Privacy)**.** *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ with input domian $\mathcal{D}$ and output domain $\mathcal{R}$ satisfies (ϵ, δ)-differential privacy ($\epsilon > 0, \delta \in [0, 1)$) if for any two neighboring inputs $D, D' \in \mathcal{D}$ and any subset of output $S \subset \mathcal{R}$ it holds that:*

$$\Pr[\mathcal{M}(D) \in \mathcal{S}] \leq e^{\epsilon} \Pr[\mathcal{M}(D') \in \mathcal{S}] + \delta. \tag{1}$$

By restricting $\delta = 0$, we have ϵ-differential privacy satisfied. It is obvious that the smaller ϵ, δ are, the higher level of privacy guaranteed.

Generally, differential privacy can be achieved by adding randomized noise. For example, let $f : \mathcal{D} \to \mathcal{R}$ be a deterministic real-valued function. We also define the l_1-sensitivity of f as the maximum l_1-norm distance over all pairs of adjacent datasets: $S_f = \max \|f(D) - f(D')\|_1$ (D, D' is a pair of adjacent datasets). ϵ-differentially private mechanism $\mathcal{M}$ can be achieved by adding randomized Laplace noise scaled by S_f:

$$\mathcal{M}(D) \triangleq f(D) + \mathrm{Lap}(S_f/\epsilon), \tag{2}$$

Apart from Laplace noise, noise drawn from Gaussian or other distributions can also fulfill differential privacy guarantees.

In differential privacy mechanisms, when the same dataset is revisited, its privacy gets leaked more than the case of one-time

visit. Composition theorem of differential privacy depicts how privacy decays with the number of times visiting the dataset. The most fundamental one is the sequential composition theorem [4].

Theorem 1 (Basic Composition)**.** *If mechanism $\mathcal{M}_i$ satisfies (ϵ_i, δ_i)-differential privacy for any $i \in [k]$, then the composition of all these mechanisms satisfies $(\sum_{i=1}^{k} \epsilon_i, \sum_{i=1}^{k} \delta_i)$-differential privacy.*

Parallel composition is adopted when differential privacy is applied to partial instead of the entire dataset. Assuming we partition the dataset into disjoint datablocks, and apply differential privacy mechanisms per block, we have:

Theorem 2 (Parallel Composition)**.** *If the release of datablock i satifies (ϵ_i, δ_i)-differential privacy, the entire dataset is $(\max_i \epsilon_i, \max_i \delta_i)$-differentially private.*

B. Online Algorithms

Without having the entire input available from the start, an online algorithm processes the input piece-by-piece in a serial fashion. In contrast, an offline algorithm is given the whole input from the beginning. Taking the classic knapsack problem as an example, we aim to pick the items with the highest total values given limited capacity of the knapsack. In the offline situation, since we know all the items before making decisions, the solution is deterministic. In the online setting, only the items coming in by the time of decision-making are available, and hence it is more difficult to make a decision without the foresight into the future. Note that all decisions are irrevocable.

From the descriptions above, the offline algorithm yields the optimal solution to a problem while the online algorithm remains suboptimal. It is a convention to use *competitive ratio* to describe the gap between the two algorithms. The closer the gap, the better the online algorithm is. Formally, we define

Definition 2 (Competitive Ratio)**.** *For any input sequence σ, $OPT(\sigma)$ is the optimal solution for σ in the offline situation, and $ALG(\sigma)$ represents the result of the online algorithm ALG for σ. The highest ratio of $OPT(\sigma)$ versus $ALG(\sigma)$ among all possible σs is the competitive ratio of ALG.*

$$CR = \max_{\sigma} \frac{OPT(\sigma)}{ALG(\sigma)}. \tag{3}$$

The goal of designing any online algorithm is to obtain the smallest CR possible, representing the return in expectation is closest to the offine case.

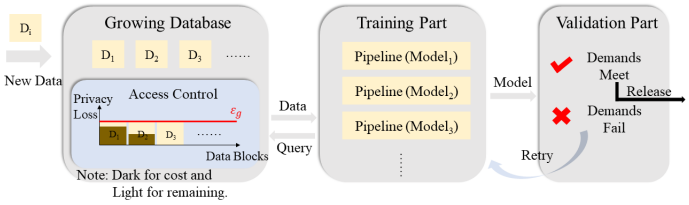


Fig. 1: Privacy-preserving machine learning pipeline.

IV. Architecture and Formulation

In this section, we first introduce the concept of privacy budget and the privacy-preserving machine learning pipeline. Then we formulate the problem of datablock access control w.r.t. the privacy budget. The objective is to make the most use of datablocks under their privacy constraints.

A. Privacy-Preserving ML Pipeline

Real-world machine learning services often deal with growing databases. Different from static databases, the input data are generated in real time and fed to the ML pipeline. Growing databases are not streaming data either, since the streaming data comes and goes in a real-time fashion but growing data would stay until getting retired for use. We consider the private incoming data arrive block-by-block, and each block is associated with a privacy budget. The privacy budget is spent with each access to the block and when the privacy budget drains away, the datablock retires and cannot be used any more. The privacy budget can be gauged by the following privacy loss, and gets deducted each time that a query accesses the datablock.

Definition 3 (Privacy Loss)**.** *For any output o, mechansim $\mathcal{M}$ and a pair of adjacent datasets D, D', the privacy loss of o is*

$$\text{Loss}(o) \triangleq \ln \frac{\Pr[\mathcal{M}(D) = o]}{\Pr[\mathcal{M}(D') = o]}. \tag{4}$$

It is obvious by definition that the privacy loss is upper bounded by ϵ for ϵ-differentially private $\mathcal{M}$. Hence the privacy budget of each datablock gets deducted at each differentially-private access from the query. Correspondingly, each query has a privacy cost indicating how much budget it would spend. A datablock cannot be used any more when its budget drains out, as further access would violate its privacy.

As shown in Fig. 1, the privacy-preserving ML pipeline consists of three components — data access control, training and validation. Private incoming datablock first goes into the access control section, which audits the privacy budget of each datablock and restricts each query's access to the datablock. Only the datablock obtaining the warrant can enter the next section. In training section, ML models are trained on the datablocks and then sent to the validation section. At this stage, if the validation result on the public data meets the demand, the model gets released. Otherwise, the model is sent back to the training section for further training until meeting the performance criteria. We assume the validation data is given in advance rather than chosen from the growing database.

Taking Fig. 1 as an example, datablocks continuously flow into the database, where block D_1, D_2, D_3 are already in the pool. We assume the privacy budget for the entire database is ϵ_g. By parallel composition, we know each datablock's privacy budget is capped by ϵ_g. The shaded area of each datablock represents consumed budget with query accesses. When the entire area of the block gets dark, the block gets retired. Any sequential composition theorem like basic composition

in Thm. 1 can be used for accounting the privacy loss for each single datablock.

B. Problem Formulation

In practice, in the architecture of Fig. 1, there are often multiple training pipelines contending for datablocks at the same time. The privacy of each datablock is a valuable resource that needs to be carefully arranged for better serving queries. For example, repeated queries to a datablock which is not that useful may drain out its privacy for nothing. Unfortunately, a query does not really understand the datablocks since they are mostly private, and thus it leaves to the platform to assign suitable datablocks to queries. We formulate the problem of datablock access control w.r.t. privacy budget as an optimization problem. The objective is to assign suitable datablocks to queries such that the privacy budget is well-spent to generate the highest values.

We introduce the concept of data-query significance to describe the value of a datablock to a query. We use p_{ij} to represent the importance of datablock D_i to query Q_j. Or in other word if Q_j is meant for serving some model, p_{ij} denotes the significance of D_i to the model. This value can be obtained by prior knowledge about the dataset. For instance, for a time-sensitive dataset, if the incoming datablock is outdated or not recent, a lower p_{ij} value can be assigned than those up-to-date data. In the case without any prior knowledge, we can simply adopt the change in the validation accuracy to determine the significance of the training datablock. The larger performance boost it brings to the model, the more significant the datablock is.

Based on the data-query significance, we formulate the datablock assignment problem as an optimization one. We aim to choose the most suitable datablocks to maximize the overall significance for all queries, *i.e.,* the utility of all datablocks upon its release. Assuming that there are m datablocks and n queries, the overall problem is:

$$\begin{aligned} Z \;=\; \max_{x_{ij}} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}x_{ij}, \\ \text{s.t.} \quad & \sum_{j=1}^{n} \epsilon_j x_{ij} \le \epsilon_{gi},\ i \in I = \{1,2,...,m\} \\ & \sum_{i=1}^{m} x_{ij} \le c_j,\ j \in J = \{1,2,...,n\} \\ & x_{ij} \in \{0,1\},\ i \in I,\ j \in J. \end{aligned} \tag{5}$$

x_{ij} is the assignment variable such that $x_{ij} = 1$ if D_i serves Q_j and $x_{ij} = 0$ otherwise. Datablock D_i has a total privacy budget ϵ_{gi}, and query Q_j would consume ϵ_j out of the budget at each access. Moreover, to restrain a query from consuming too much privacy budget, we incorporate a variable c_j to denote the maximum number of datablocks that Q_j can access, which, implicitly imposes fairness across all queries. The second constraint denotes the access limit per query.

V. Privacy Budgeting: An Ideal Case

In this section, we seek the solution to problem (5) in an ideal case where all datablocks and queries are known in advance. Problem (5) is a 0-1 integer programming problem which can be traditionally solved by Branch and Bound algorithm [16]. The idea is to form all candidate solutions as a rooted tree and check the branch against an estimated bound to decide whether to discard it before enumerating the solutions.

In our integer programming problem, we use a binary decision tree to represent possible solutions. Each level represents an assignment variable which appends to the nodes in the level above. The left branch represents a value 1 is chosen, and the right branch denotes a value 0 is chosen for the current assignment variable. Instead of traversing all nodes, we prune the tree based on whether a node violates constraints or whether a node potentially leads to the optimal solution. We track the maximum objective value found so far. If the upper bound of a node does not exceed the recorded maximum objective value, then the branches of this node have no potential and can be eliminated. Thus the search space reduces.

To obtain the upper bound for a specific node, we introduce the Lagrangian relaxation of Z in problem (5) as follows:

$$\begin{aligned} Z(\lambda) \;&=\; \max_{x_{ij}} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}x_{ij} - \sum_{j=1}^{n}\lambda_j\Big(\sum_{i=1}^{m} x_{ij} - c_j\Big) \\ &=\; \max_{x_{ij}} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} (p_{ij}-\lambda_j)x_{ij} + \sum_{j=1}^{n}\lambda_j c_j \\ &\qquad \text{s.t.} \quad \sum_{j=1}^{n} \epsilon_j x_{ij} \le \epsilon_{gi},\ i \in I \\ &\qquad\quad x_{ij} \in 0,1,\ i \in I,\ j \in J, \end{aligned} \tag{6}$$

for any positive multipliers $\lambda = \{\lambda_1, \lambda_2, ..., \lambda_n\}$. Note that $Z(\lambda)$ is an upper bound of Z and can be easily partitioned into m sub-problems, each for $i \in I$:

$$\begin{aligned} \max_{x_{ij}} \quad & \sum_{j=1}^{n} p_{ij}x_{ij} - \lambda_j x_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{n} \epsilon_j x_{ij} \le \epsilon_{gi} \\ & x_{ij} \in 0,1,\ j \in J. \end{aligned} \tag{7}$$

A branch-and-bound algorithm of a much smaller scale can be adopted to solve each sub-problem. And then we perform the dual ascent algorithm [17] to maximize problem (6) by repeatedly adjusting the multipliers. If the upper bound of a node exceeds the maximum value obtained so far, we furtherly check the branches of the node to see if they would replace the current maximum one.

The initial solution has a key role in the efficiency of solving the problem. In practice, we pick blocks with top c_j data-query significance to start with and find the heuristic is efficient in obtaining the optimal solutions.

Advanced composition: In Eqn. (5), advanced composition can replace basic composition to calculate the privacy budget for a tighter bound. In that case, we relax ϵ-differential privacy to (ϵ, δ)-differential privacy and the problem has a similar form to Eqn. (5) but only replaces the sum of $\epsilon_j x_{ij}$ with a tighter one as in Thm. 3.5 of [18], and adds additional constraints on $\tilde{\delta}_i$ such that $\tilde{\delta}_i \leq \delta_{gi},\ i \in I$. We can still apply branch-and-bound algorithm with Lagrangian relaxation to find an offline solution.

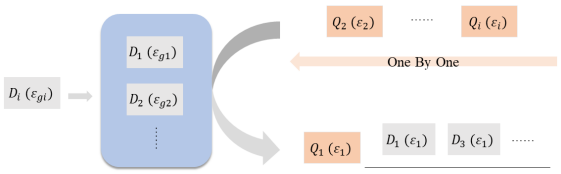


Fig. 2: Privacy accounting for datablocks and queries arriving continuously.

VI. Privacy Budgeting for Growing Databases

In this section, we discuss the real-world cases where both datablocks and queries go to the ML platform in an online fashion. Datablocks are associated with privacy budgets and queries have corresponding privacy costs. We aim to design a privacy budget scheduler to assign datablocks to queries to make the most use out of the datablocks. Typically the decision has to be made irrevocably as soon as the query arrives. A sketch diagram is shown in Fig. 2.

Problem (5) is intrinsically hard to solve when datablocks and queries arrive continuously. From a query perspective, it should take advantage of the datablock in the pool with top data-query significance. However, it may not be the datablock's best interest to serve the query, as it can wait until queries with higher data-query significance arrive.

Relation with multiple knapsack problem: Problem (5) resembles a multiple knapsack problem, if one treats datablocks as knapsacks and queries as goods. With the privacy budget of the datablock as the capacity of the knapsack, the privacy cost of the query as the weight of the goods, the data-query significance as the value, we can think of the problem as one to fill up all knapsacks with the most valuable goods. However, our problem is quite different from the multiple knapsack problem from the following aspects: 1) The number of datablocks is growing so that the number of 'knapsacks' is not fixed but increases with time. 2) A query may pick several datablocks which means 'goods' could be placed into more than one 'knapsack'. 3) The data-query significance concerns both the datablock and the query hence the 'goods' has unique values in different 'knapsacks.' 4) It is unacceptable to abandon any 'goods' as the platform cannot turn down any query. 5) Most importantly, the comparison of value-weight ratios among multiple 'goods' is not free in our setting: a privacy price has to be paid at each comparison since it is privacy-leaking. Hence, our problem cannot be tackled with existing solutions of the multiple knapsack problem.

In a nutshell, our goal is to trade the privacy cost of comparison for better use of datablocks, and design an online algorithm with the smallest CR possible as Section III-B describes. The key idea is to calculate the significance-privacy ratio for each query and compare it with a threshold in a differentially-private way. As each comparison would consume the privacy budget of a datablock, one has to take it into account in calculating the total privacy loss. In the following, we first introduce a differentially-private comparison scheme for our problem, and then describe an online algorithm with the smallest CR under some assumptions. Finally, we extend our algorithm to a more general case.

A. An Online Budgeting Algorithm

Naively, to make the most use of the privacy budget, one should check the ratio between data-query significance and privacy cost of each query, and pick the queries with top ratios to fulfill. However, the naive method consumes too much privacy. Instead, we design a filtering mechanism to select queries with such a ratio surpassing a given threshold. The threshold is defined as a function of the consumed proportion of the privacy budget in the datablock. The intuition is that when more privacy budget is consumed, the datablock becomes stricter at picking which query to answer.

As data-query significance reveals the importance of a datablock to a query, the comparison with a given threshold would leak privacy. Hence we adopt a variant of the Above-Threshold algorithm in [4] to calculate the privacy loss for comparison. The detailed algorithm is given in Alg. 1. For datablock D_i, if its consumed proportion of privacy budget does not exceed a constant z_0, we let it take any query it can take; otherwise, the data-query significance is calculated to compare with $T(z_i)$ which we will elaborate on later. The query is only assigned the datablock if it passes the check.

Algorithm 1: Filtering by Threshold

Input: Data block D_i with privacy budget ϵ_{gi} and consumed budget proportion $z_i \in [0, 1]$; Query Q_j with privacy cost ϵ_j; Data-query significance p_{ij} between D_i and Q_j; Comparison cost ϵ_0.

Output: Assignment x_{ij}, updated z_i.

if $z_i < z_0$ **then**
  $x_{ij} = 1$;
  $z_i = z_i + \frac{\epsilon_j}{\epsilon_{gi}}$;
else
  Calculate p_{ij};
  if $\frac{p_{ij}}{\epsilon_j + \epsilon_0} + Lap(\frac{4}{\epsilon_0}) > T(z_i) + Lap(\frac{2}{\epsilon_0})$ *and* $\epsilon_j + \epsilon_0 < (1 - z_i)\epsilon_{gi}$ **then**
    $x_{ij} = 1$;
    $z_i = z_i + \frac{\epsilon_j + \epsilon_0}{\epsilon_{gi}}$;
  else
    $x_{ij} = 0$;
  end
end

The value of z_0, ϵ_0 and the form of $T(z_i)$ will be given later, as they are concerned with the privacy budget and the value of CR. With Alg. 1, we will introduce our online algorithm for privacy budgeting.

As queries coming continuously, each datablock chooses to serve or decline it. We show that with mild assumptions, we can provide the optimal CR among all online algorithms. The optimality comes from the best CR for the multiple knapsack problem [19] proved by Yao's minimax principle [20].

We make the following two assumptions in our design of online algorithm.

Assumption 1. *The privacy cost of a query is much smaller than the privacy budget of a datablock.*

Assumption 2. *The ratio between the data-query significance and the total privacy cost of a query is bounded by U and L.*

$$L \leq \frac{p_{ij}}{\epsilon} \leq U, \quad \forall\ i \in I,\ j \in J. \tag{8}$$

With Assumption (2), we define $z_0 = \frac{1}{1+\ln(U/L)}$, and the threshold function as

$$T(z) = \begin{cases} L, & 0 \leq z \leq z_0, \\ (\frac{Ue}{L})^z \frac{L}{e}, & z_0 < z \leq 1. \end{cases} \tag{9}$$

If the consumed proportion of the privacy budget does not exceed z_0, our algorithm takes any query for the datablock (as the threshold function value is L), which ensures even a query with a lower significance value would be served. As the consumed budget exceeds z_0, the threshold becomes stricter with the increase of the consumed amount. Whenever a block is chosen, the corresponding model of the query is trained on the datablock and has its performance validated. We use $Flag$ to mark whether the model is accepted: the flag is set to true only if its performance is up to par. The detailed algorithm is stated in Alg. 2. Line 3 ~ 6 add the newly arrived datablock into the pool and Line 7 ~ 11 retire the datablocks which have run out their privacy budgets. Line 15 admits datablocks that pass the filtering-by-threshold algorithm. The model is trained on the chosen blocks and validated until its performance is up to par (with $Flag = True$). Moreover, the query is restricted to visit at most c_j blocks. The model needs to send more queries once such an upper bound is met.

Algorithm 2: Online Budgeting

Input: Data blocks in the pool $D = \{D_1, D_2, ..., D_i, ...\}$. For datablock D_i, its privacy budget is ϵ_{gi}, consumed proportion of budget is z_i and the threshold function is $T(z_i)$. Query Q_j for the corresponding model M_j has privacy cost ϵ_j, limit of accesses c_j and sum of current queried blocks s_j. Model performance validation flag $Flag$. Comparison cost ϵ_0.

Output: Assignment $\{x_{ij}\}$.

1 Initialize $j = 1$, $z_i = 0$ for all $D_i \in D$;
2 **while** $True$ **do**
3 **if** *new data block D_m comes* **then**
4 $D = D \cup \{D_m\}$;
5 $z_m = 0$;
6 **end**
7 **for** D_i *in* D **do**
8 **if** $z_i \geq 1$ **then**
9 $D = D \setminus \{D_i\}$; (D_i is retired.)
10 **end**
11 **end**
12 $s_j = 0$; $Flag = False$; $\hat{D} = D$;
13 **while** $s_j < c_j$ *and* $\hat{D} \neq \emptyset$ *and not* $Flag$ **do**
14 Randomly choose D_i in $\hat{D}$, $\hat{D} = \hat{D} \setminus \{D_i\}$;
15 x_{ij}, z_i = Filtering by Threshold(D_i, ϵ_{gi}, z_i, Q_j, ϵ_j, p_{ij}, ϵ_0);
16 **if** $x_{ij} = 1$ **then**
17 Update M_j and the validation $Flag$;
18 $s_j = s_j + 1$;
19 **end**
20 **end**
21 $j = j + 1$;
22 **end**

B. Proofs

Theorem 3. *Alg. 1 satisfies ϵ_0-differential privacy.*

The proof is ommited here. Interesting readers may refer to the proof of the Above-Threshold algorithm in [4].

Theorem 4. *Alg. 2 has a competitive ratio of $1 + \ln(U/L)$ under Assumption 1, 2.*

Proof. We first discuss the case of a single datablock and then extend the proof to multiple datablocks. Given a fixed input sequence σ, we assume the datablock terminates costing Z fraction of its privacy budget using the online algorithm. We use $ALG_i(\sigma)$, $OPT_i(\sigma)$ to respectively denote the total significance values of queries that D_i satisfies in our online algorithm and in the optimal offline case with S and S' being the corresponding sets of satisfied queries. Let $P_i = \sum_{j \in (S \cap S')} p_{ij}$, $W_i = \sum_{j \in (S \cap S')} \epsilon_{ij}$, $P_i' = \sum_{j \in (S \setminus S')} p_{ij}$.

For queries not picked by ALG_i, their significance-privacy ratios must be smaller than or equal to $T(Z)$ so we can get:

$$OPT_i(\sigma) \leq P_i + T(Z)(\epsilon_{gi} - W_i). \tag{10}$$

Only queries with a ratio surpassing the threshold function would be satisfied. Q_j fulfills the budget of D_i by ϵ_j which raises the consumed proportion from z_i to z_{i+1}. Hence we have:

$$\begin{aligned} P_i \geq \sum_{Q_j \in (S \cap S')} T(z_i)(\epsilon_j + \epsilon_0) &= \sum_{Q_j \in (S \cap S')} T(z_i)(z_{i+1} - z_i)\epsilon_{gi} \\ &= \sum_{Q_j \in (S \cap S')} T(z_i)\Delta z_i \epsilon_{gi} \triangleq P. \end{aligned} \tag{11}$$

As $OPT_i(\sigma) \geq ALG_i(\sigma)$, we have:

$$\frac{OPT_i(\sigma)}{ALG_i(\sigma)} \leq \frac{P_i + T(Z)(\epsilon_{gi} - W_i)}{P_i + P_i'} \leq \frac{P + T(Z)(\epsilon_{gi} - W_i)}{P + P_i'}. \tag{12}$$

By monotonicity, $P \leq T(Z)W_i$. Similar to Eq. (11), $P_i' \geq \sum_{Q_j \in (S \setminus S')} T(z_i)\Delta z_i \epsilon_{gi}$. Eq. (12) turns into:

$$\frac{OPT_i(\sigma)}{ALG_i(\sigma)} \leq \frac{T(Z)\epsilon_{gi}}{\sum_{Q_j \in S} T(z_i)\Delta z_i \epsilon_{gi}} \tag{13}$$

According to Assumption 1, $\Delta z_i = z_{i+1} - z_i$ is a very small amount and therefore the denominator in Eq. (13) can be approximated by:

$$\begin{aligned} \sum_{Q_j \in S} T(z_i)\Delta z_i \epsilon_{gi} &\approx \ \epsilon_{gi} \int_0^Z T(z)dz \\ &= \epsilon_{gi}\left[\int_0^{z_0} L dz + \int_{z_0}^{Z} (\frac{Ue}{L})^z \frac{L}{e} dz\right] \\ &= \frac{\epsilon_{gi} T(Z)}{1 + \ln \frac{U}{L}}. \end{aligned} \tag{14}$$

Then we can figure out the CR for a single datablock. The CR of our online algorithm can be obtained by summing up all blocks:

$$\frac{OPT(\sigma)}{ALG(\sigma)} = \frac{\sum_i OPT_i(\sigma)}{\sum_i ALG_i(\sigma)} \leq 1 + \ln \frac{U}{L}, \tag{15}$$

$$CR = \max_{\sigma} \frac{OPT(\sigma)}{ALG(\sigma)} = 1 + \ln \frac{U}{L}. \tag{16}$$

As $1 + \ln \frac{U}{L}$ is proven to be the best CR for multiple knapsack problems [19], we can similarly prove that Alg. 2 achieves the best CR among all online solutions under Assump. 1, 2. Proof completes. □

Adaptive Composition: Beyond basic composition, we can also apply a tighter privacy composition bound with advanced composition theorem. Different from the offline case, the privacy parameters ϵ_js are assumed to be adaptively chosen rather than fixed in advance, so that Thm. 3.5 in [18] does not apply anymore. We can adopt the adaptive composition theorem [21] by relaxing ϵ-differential privacy to (ϵ, δ)-differential privacy. Similarly, we replace ϵ_j with adaptive privacy budget in our algorithms. A better utility can be achieved with this tighter composition.

C. Online Budgeting with Mixed Queries

The online budgeting algorithm is based on Assump. 1 where the privacy cost of a single query is much smaller than the privacy budget of a datablock. We ask the question that what performance we can guarantee when the privacy cost of some queries is comparable with the budget, or what we refer to as the case of 'mixed queries.'

To the best of our knowledge, there is no constant CR for online algorithms without Assump. 1 so far, as the proof relies on the integration operation like Eqn. (14). We heuristically make a compromise by letting a datablock partially satisfy a query with a higher cost. From the datablock perspective, a high threshold would result in a high total significance with privacy budget constraints, but few queries passing the threshold. We found a better tradeoff can be achieved if we allow datablocks to serve queries with lower significance value by reducing the budgets assigned. From queries' view, it is able to access lower-quality datablocks other than being turned down. We show that by removing Assump. 1, we design a new algorithm having the same CR: $1 + \ln(U/L)$ with the compromise.

We design a constant threshold $T = \frac{U}{1+\ln(U/L)}$, and calculate the corresponding privacy cost $\epsilon^\star$ that query Q_j should have to be able to access datablock D_i:

$$\frac{p_{ij}}{\epsilon^\star + \epsilon_0} = T, \tag{17}$$

where ϵ_0 denotes the privacy loss by the filtering-by-threshold algorithm. We allow D_i to assign some budget to fulfill partial request of Q_j:

$$\epsilon_{ij} = \begin{cases} 0 & \epsilon^\star < \eta \cdot \epsilon_j, \\ \min\{\epsilon^\star, \epsilon_j, \epsilon_{gi}^r - \epsilon_0\} & \epsilon^\star \geq \eta \cdot \epsilon_j. \end{cases} \tag{18}$$

ϵ_{gi}^r is the residual budget of D_i at the time of assignment. $\eta \in [0, 1]$ is a parameter which can be adjusted. When $\epsilon^\star$ is smaller than $\eta\epsilon_j$, it indicates p_{ij} may be too small and D_i is not important to Q_j at all. Hence the query should not be picked by the datablock anyway. Otherwise, D_i assigns an amount depending on the minimum value of $\epsilon^\star$, ϵ_j and the residual budget of D_i. The details are in Alg. 3.

Algorithm 3: Online Budgeting with Large Queries

Input: Datablocks in the pool $D = \{D_1, D_2, ..., D_i, ...\}$. D_i maintains a remaining privacy budget ϵ_{gi}^r. Query Q_j for model M_j has privacy cost ϵ_j, limit of accesses c_j and sum of current queried blocks s_j. Comparison cost ϵ_0. Filtering parameter η. Model performance validation flag $Flag$.

Output: Assignment $\{x_{ij}\}$.

```
Initialize j = 1; ε^r_gi = ε_gi for D_i ∈ D;
while True do
    Update datablocks as Line 3 ~ 11 in Alg. 2;
    s_j = 0; Flag = False; D̂ = D;
    while s_j < c_j and D̂ ≠ ∅ and not Flag do
        Randomly choose D_i in D̂, D̂ = D̂ \ {D_i};
        Calculate p_ij and ε*;
        if ε* + Lap(4/ε_0) ≥ η · ε_j + Lap(2/ε_0) then
            x_ij = 1;
            ε^r_gi = ε^r_gi − min{ε*, ε^r_gi − ε_0, ε_j} − ε_0;
            Update M_j and the validation Flag;
            s_j = s_j + 1;
        else
            x_ij = 0;
        end
    end
    j = j + 1;
end
```

Theorem 5. *Alg. 3 has a competitive ratio of* $1 + \ln(U/L)$ *under Assumption 2.*

Proof. With the new threshold, the privacy budget of each datablock would be spent out as the time passes. Hence the optimal offline result is bounded by Assump. 2:

$$OPT_i(\sigma) \leq U\epsilon_{gi}. \tag{19}$$

The lower bound of our algorithm depends on the threshold T:

$$ALG_i(\sigma) \geq T \cdot \epsilon_{gi} = \frac{U \cdot \epsilon_{gi}}{1 + \ln \frac{U}{L}}. \tag{20}$$

Therefore, the CR satisfies:

$$CR = \max_{\sigma} \frac{OPT(\sigma)}{ALG(\sigma)} = \max_{\sigma} \frac{\sum_i OPT_i(\sigma)}{\sum_i ALG_i(\sigma)} = 1 + \ln \frac{U}{L}. \tag{21}$$

Note that the CR value is equivalent to the previous one, despite that we cannot prove it is the smallest CR in this case. The CR still provides some worst-case performance guarantee. Proof completes. □

VII. Experiments and Evaluation

We mainly consider the following aspects in the experiments: 1) how the introduction of data-query significance affects the model accuracy; 2) the impact of block size on the model accuracy; 3) the gap between the offline and online algorithm; 4) how our algorithms compare with previous works. The experiments are done on three datasets with setup in Table I.

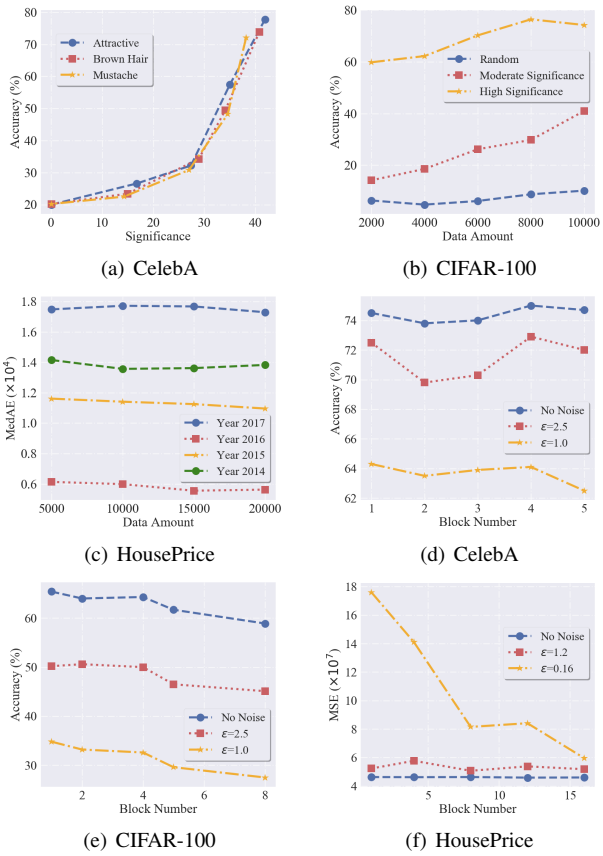
TABLE I: Datasets description

	CelebA [22]	CIFAR-100 [23]	HousePrice [24]
Size	202599 images	60000 images	318851 records
Synopsis	40 attritubes per image	100 classes	22 features per record
Task	Binary classification	Multi classification	Regression
Model	DNN (ResNet-20, etc.)	DNN (ResNet-32, etc.)	ML (GBDT, etc.)
Metric	Testing Accuracy	Testing Accuracy	MSE[1], MedAE[2]

A. Data-Query Significance

For the two classification tasks, the data-query significance is defined by the reduced validation loss over the training datablock. The larger the reduction, the more important the block is. This is a straightforward way to gauge significance as the value depends on the training phase of the model. For the regression task, we adopt the generation time of the datablock as the significance value. The closer the generation time to the prediction task, the more important the datablock.

Fig. 3(a)3(b)3(c) show how different significance values impact model accuracy. On CelebA, we deploy three binary classification tasks respectively for three attributes, and train over the same amount of datablocks of different significant values. As can be seen in Fig. 3(a), the more significant the training datablock is, the higher the testing accuracy is. On CIFAR-100, we train the same model for the same classification task over data chosen by different means: random choices, datablocks with moderate/high significance values. As shown in Fig. 3(b), for the same amount of training data, those with higher significance yields higher accuracies. On the HousePrice dataset, we deploy a task to predict the price of 2016, and respectively train the model over the data from 2014 to 2017. Fig. 3(c) shows that training data closer to 2016 tends to produce a model with a lower error rate.



(a) CelebA (b) CIFAR-100 (c) HousePrice (d) CelebA (e) CIFAR-100 (f) HousePrice

Fig. 3: (a)(b)(c) We verify the effect of data-query significance on various datasets. (d)(e)(f) The effect of block size depends on datasets and the privacy budget.

B. Effect of Block Size

To simulate the growing dataset, we divide the entire dataset into different numbers of disjoint blocks, shuffle, and feed the datablocks one by one to our access control engine. It naturally raises the question that how different block sizes affect the final model accuracy. We split the privacy budget evenly to each data record and compose the budget of a block to insert noise accordingly.

On CelebA and CIFAR-100, as Fig. 3(d) and 3(e) show, the accuracy does not vary significantly with the change in the block size, but so with the privacy budget. On the HousePrice dataset, we observe in Fig. 3(f) that when the privacy budget

[1]Mean Square Error, $\text{MSE}(y, \hat{y}) = \frac{1}{N_{samples}} \sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2$

[2]Median Absolute Error, $\text{MedAE}(y, \hat{y}) = median(|y_1 - \hat{y}_1|, \ldots, |y_n - \hat{y}_n|)$

is relatively small ($\epsilon = 0.16$), a smaller block size (or a larger block number) would result in a lower error rate. It may be because with a smaller privacy budget, a larger amount of noise is inserted, which leads to a higher level of randomness. It is preferred to train the model for more iterations for better accuracy.

C. Offline v.s. Online

Drawing on the experimental results in previous sections, we carefully choose the data-query significance and the block size to boost model performance in the following experiments. We simulate both the offline and online algorithms and show how our online solutions compare with the optimal one. On CelebA and CIFAR-100 each, we have a total of 8 blocks and 12 queries. The queries are all of small costs in the online budgeting case, and of mixed costs in the online budgeting with large queries. We set 10 blocks and 2 queries for the HousePrice. Apart from using metrics in Sec. I, we also use the average Significance value to compare the algorithms' performance, which is the direct optimization objective.

As we observe from Table II, the offline algorithm yields the highest model utility among all cases, while our online algorithms result in models that are reasonably good. We do not observe obvious differences in the offline and online gaps for both our online algorithms. Remember that the CRs are the same for both online algorithms, indicating the worst-case performance is approximately the same. Our experiments may not display the worst case, but show consistent results.

TABLE II: Offline and Online Results

	Small-Cost Queries						
Dataset	CelebA		CIFAR-100		HousePrice		
(Average)	Acc.(%)	Sig.	Acc.(%)	Sig.	MSE	MedAE	Sig.
Offline	75.3	43.4	61.0	23.1	1.5×10^8	7189.2	165.0
Online	65.1	30.7	45.0	13.5	2.0×10^8	8437.3	145.0
	Mixed-Cost Queries						
Dataset	CelebA		CIFAR-100		HousePrice		
(Average)	Acc.(%)	Sig.	Acc.(%)	Sig.	MSE	MedAE	Sig.
Offline	75.3	43.9	62.0	23.7	1.1×10^8	6014.7	166.5
Online	64.5	32.2	53.3	17.0	1.6×10^8	6909.2	146.5

D. Comparison with Other Works

We compare our design with two private datablock distribution methods in previous works:

- Streaming [9]: Data comes and goes in a stream. Each datablock can only be used once. Queries choose datablocks in a random way.
- Sage [8]: Datablocks can be repeatedly accessed until their privacy budget runs out. Queries choose datablocks in a random way.

Again, we simulate the situations where queries are of small costs, and queries of mixed costs, for all the methods in comparison. We conduct three sets of experiments.

First, on the same amount of training data, we compare the model performance for different methods. Since we cannot ensure an equivalent amount of privacy costs on streaming data, we only compare our algorithms with Sage here. On three datasets, we respectively prepare 6, 6, and 4 queries in total. The averaged query (model) performance is reported in Table III, where we found our algorithm outperforms Sage by a large margin in all cases.

TABLE III: Model Performance Using Same Amount of Data

	Small-Cost Queries			
Dataset	CelebA	CIFAR-100	HousePrice	
(Average)	Acc.(%)	Acc.(%)	MSE	MedAE
Our Alg.	72.8	61.8	1.8×10^8	7657.3
Sage	47.5	39.0	2.5×10^8	9684.6
	Mixed-Cost Queries			
Dataset	CelebA	CIFAR-100	HousePrice	
(Average)	Acc.(%)	Acc.(%)	MSE	MedAE
Our Alg.	76.5	61.3	2.4×10^8	9477.0
Sage	31.5	37.5	3.6×10^8	11689.9

Next, we investigate the amount of data that queries would require to pass the validation phase. A query passes the validation only if the model accuracy/error surpasses/drops below a presetted value. Five queries with distinctive models and tasks are adopted for each of the three datasets. On CelebA and CIFAR-100, we choose datablocks from a pool of 12 blocks with each consisting of 5000 images. On HousePrice, we choose datablocks from a pool of 20 blocks with each block containing 2000 records. We count the total number of blocks used until passing the validation phase and summarize the accumulated results in Table IV. Our algorithms require the least amount of datablocks, followed by Sage, and then the Streaming method. This shows the strength of our method in saving the privacy budget for better utility. Streaming method consumes the most datablocks since its data cannot be repeatedly used and hence the privacy budget quickly drains out. The consumption of Sage is in between, which shows the advantage of reusing datablocks and a lack of awareness on where the privacy budget is spent.

TABLE IV: Accumulated Number of Blocks Used

	Small-Cost Queries														
Dataset	CelebA					CIFAR-100					HousePrice				
Model	①	②	③	④	⑤	❶	❷	❸	❹	❺	①	②	③	④	⑤
Our Alg.	1	3	3	4	5	1	2	3	4	5	3	4	4	6	10
Sage	3	4	5	6	7	2	4	5	5	6	5	8	10	14	15
Streaming	3	5	7	10	12	2	4	7	9	11	5	9	13	15	19
	Mixed-Cost Queries														
Dataset	CelebA					CIFAR-100					HousePrice				
Model	①	②	③	④	⑤	❶	❷	❸	❹	❺	①	②	③	④	⑤
Our Alg.	2	3	4	5	5	1	3	4	4	6	3	4	4	8	10
Sage	3	4	5	7	8	3	5	5	6	7	5	7	7	12	14
Streaming	3	5	7	10	12	2	4	7	9	11	5	8	10	13	19

Finally, we test the three methods for the number of queries served on the same amount of data, or equivalently, the same privacy budget. We design a fixed sequence of incoming queries to see how many queries are served until the privacy budget is spent out. The results in Table V show that our method is most efficient by serving the most number of queries with the same privacy budget, followed by Sage. Streaming yields the lowest data utility among all.

TABLE V: Queries Served with Same Amount of Data

	Small-Cost Queries								
Dataset	CelebA			CIFAR-100			HousePrice		
Data Amount($\times 10^4$)	2	4	6	1.5	3	4.5	2	3	4
Our Alg.	19	33	41	20	31	39	8	14	19
Sage	11	20	27	13	20	29	7	12	16
Streaming	1	2	4	1	2	3	1	3	4
	Mixed-Cost Queries								
Dataset	CelebA			CIFAR-100			HousePrice		
Data Amount($\times 10^4$)	2	4	6	1.5	3	4.5	2	3	4
Our Alg.	13	20	33	11	15	22	9	15	20
Sage	4	7	11	4	7	10	7	12	15
Streaming	1	2	4	1	2	3	2	3	5

From the above experiments, we verify that our algorithm yields the highest model utility by assigning datablocks to queries under the constraint of the privacy budget.

VIII. Conclusion

In this paper, we propose a differentially-private access control mechanism on the ML platform. Instead of accounting privacy loss in a posterior way, we carefully do privacy budgeting by assigning datablocks to queries that yield the highest utility. An offline, two online solutions are given under different assumptions. Experimental results support that our algorithms are most effective in serving queries under the same privacy budget.

References

[1] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography conference (TCC)*. Springer, 2006, pp. 265–284.

[2] "Tensorflow privacy," https://github.com/tensorflow/privacy.

[3] "Pysyft," https://github.com/OpenMined/PySyft.

[4] 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.

[5] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[6] I. Roy, S. T. V. Setty, A. Kilzer, V. Shmatikov, and E. Witchel, "Airavat: Security and privacy for mapreduce," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'10. USA: USENIX Association, 2010, p. 20.

[7] P. Mohan, A. Thakurta, E. Shi, D. Song, and D. Culler, "Gupt: Privacy preserving data analysis made easy," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '12. New York, NY, USA: Association for Computing Machinery, 2012, pp. 349–360.

[8] M. Lecuyer, R. Spahn, K. Vodrahalli, R. Geambasu, and D. Hsu, "Privacy accounting and quality control in the sage differentially private ml platform," *Proceedings of the 27th ACM Symposium on Operating Systems Principles (SOSP)*, 2019.

[9] C. Dwork, "Differential privacy in new settings," in *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '10. USA: Society for Industrial and Applied Mathematics, 2010, pp. 174–183.

[10] C. Dwork, M. Naor, T. Pitassi, G. N. Rothblum, and S. Yekhanin, "Pan-private streaming algorithms." in *ICS*, 2010, pp. 66–80.

[11] J. Bolot, N. Fawaz, S. Muthukrishnan, A. Nikolov, and N. Taft, "Private decayed predicate sums on streams," in *Proceedings of the 16th International Conference on Database Theory (ICDT)*, 2013, pp. 284–295.

[12] R. Cummings, S. Krehbiel, K. A. Lai, and U. Tantipongpipat, "Differential privacy for growing databases," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 8864–8873.

[13] F. Cuppens, F. Bouattour, and F. B. Fredj, "Optimal distribution of privacy budget in differential privacy," in *Risks and Security of Internet and Systems: 13th International Conference, CRiSIS 2018, Arcachon, France, October 16–18, 2018, Revised Selected Papers*, vol. 11391. Springer, 2019, p. 222.

[14] L. Xiang, J. Yang, and B. Li, "Differentially-private deep learning from an optimization perspective," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 559–567.

[15] G. Kellaris, S. Papadopoulos, X. Xiao, and D. Papadias, "Differentially private event sequences over infinite streams," *Proc. VLDB Endow.*, vol. 7, no. 12, pp. 1155–1166, Aug. 2014.

[16] "Knapsack problem with branch and bound pruning," Website, http://faculty.chas.uni.edu/~east/teaching/153/branch_bound/knapsack/overview_algorithm.html.

[17] J. S. Park, B. H. Lim, and Y. Lee, "A lagrangian dual-based branch-and-bound algorithm for the generalized multi-assignment problem," *Management Science*, vol. 44, no. 12-part-2, pp. S271–S282, 1998. [Online]. Available: https://pubsonline.informs.org/doi/abs/10.1287/mnsc.44.12.S271

[18] P. Kairouz, S. Oh, and P. Viswanath, "The composition theorem for differential privacy," ser. 32nd International Conference on Machine Learning, ICML 2015, D. Blei and F. Bach, Eds. International Machine Learning Society (IMLS), Jan. 2015, pp. 1376–1385.

[19] Y. Zhou, D. Chakrabarty, and R. Lukose, "Budget constrained bidding in keyword auctions and online knapsack problems," in *Internet and Network Economics*, C. Papadimitriou and S. Zhang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 566–576.

[20] A. C. Yao, "Probabilistic computations: Toward a unified measure of complexity," in *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, 1977, pp. 222–227.

[21] R. M. Rogers, A. Roth, J. Ullman, and S. P. Vadhan, "Privacy odometers and filters: Pay-as-you-go composition," *CoRR*, vol. abs/1605.08294, 2016. [Online]. Available: http://arxiv.org/abs/1605.08294

[22] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[23] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 05 2012.

[24] "Second-hand house trade records in beijing," https://bj.lianjia.com/fangjia/.