Federated Data Distribution Shift Estimation

Graham Cormode
Meta and University of Warwick
gcormode@meta.com

Daniel Ting
Meta
dting@meta.com

ABSTRACT

As data is increasingly held at the edge of the network, new methods are needed to perform analysis over distributed inputs. This has led to the emergence of the federated model of distributed computation, which places emphasis on privacy and scalability. A central problem is to analyze data distributions where the data is spread across a large number of distributed clients. This supports a number of tasks within federated learning and federated analytics. We present techniques to measure the similarity of distributions of data in the federated model. We define sketches for this task that allow efficient estimation of the difference between two distributions based on the total variation distance (L_1) metric. These have accuracy and privacy guarantees, and can be computed incrementally over dynamic data. Our experimental study shows that these are practical to implement and provide accurate estimates.

PVLDB Reference Format:
Graham Cormode and Daniel Ting. Federated Data Distribution Shift Estimation. PVLDB, 18(8): 2399 - 2412, 2025.
doi:10.14778/3742728.3742736

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://figshare.com/s/7a1c725a293d1c5b88a8.

1 INTRODUCTION

The *federated model* of computation captures the case where many clients cooperate with a central aggregator to compute over their distributed data, while preserving the privacy of their information. Federated computation has been most heavily studied in the context of distributed machine learning (ML) [38], but applies more generally to any application where many clients hold data for analysis and modelling [9]. Within this setting, modern data management systems must address many important tasks: building models for prediction and inference; tracking statistics and analytics; and performing general computations over the distributed data. These are all used to inform decision making using past data. It is thus of particular importance to know when data distributions, and hence the relevant decisions, may have changed.

Several key distinctions separate the federated setting from traditional notions of computation. Privacy is paramount, and so the participating clients only have their own data to operate on. The scale can vary widely: from a small number of separate institutions forming a federation to perform a collaborative computation over the union of their (sensitive) data, to a vast number of mobile clients, such as all users of a popular app. Within this scale, heterogeneity is common: some clients have many examples while others have only a few. Moreover, the data distributions among clients can vary drastically. Understanding properties of the (global) data distribution by gathering statistics of the clients' (local, heterogenous) distributions is at the heart of many federated tasks in data management.

A key concern in (federated) data analytics and machine learning is detecting when the data distribution has shifted. For instance, when a pre-trained ML model is applied to new data, the current data distribution should be similar to the training distribution. When this is not the case, the old model's results can be incorrect, and we should trigger some remediation or retraining instead. This paper addresses this question of detecting when the data distribution has measurably changed or shifted, formalized as the problem of quantifying *data distribution shift*. This could be tackled in a number of ways. A parametric model could flag when an estimated parameter changes by more than a threshold. However, such approaches depend on whether the model still represents the data well after distribution shift. Here, we adopt a non-parametric approach, and focus on *total variation distance* (D_{TV}) between pairs of data distributions, a common statistical formulation. Our work allows data heterogeneity to be quantified, by measuring the extent to which the distributions held by different clients vary among each other, or from a reference distribution. We consider discrete distributions defined by frequencies of different *items* i. Each client holds a subset of items, which gives a local empirical distribution. The normalized item frequency gives its empirical probability, p_i.

We will use a scenario based on tracking popular music as a running example, since music can be a very personal matter, and reflective of an individual's political, sexual, and religious views. Consider a collection of distributed users who each express their personal music preferences: some listen exclusively to a few artists, while others have very wide-ranging tastes. Over this population, the preferred artists of each user correspond to the items, and so define a current distribution of popular artists. The task is to quantify how the support changes between snapshots.

If we could directly inspect the distributions, the task is easy: we can compare their (empirical) probabilities, and derive a measure of distance. The problem of measuring distances is more challenging when the distributions are: Large (i.e., high dimensional, and supported over a large number of indices); Distributed (i.e., spread out over a large collection of users); and Private (i.e., the values held by the users are private, and cannot be shared easily). To address this (Large, Distributed, Private) challenge, we describe techniques which provide compact summaries of distributions that can be merged across distributed observers, and are amenable to privacy protections. This captures the federated model of computation [9].

Our main tool is to build a "sketch" of each input distribution, which forms a small representation of it that allows the distance

Proceedings of the VLDB Endowment, Vol. 18, No. 8 ISSN 2150-8097.
doi:10.14778/3742728.3742736

to be estimated accurately. By ensuring the sketch can be merged and is amenable to combine with privacy-preserving distributed noise addition and secure aggregation, it satisfies the requirements for federated computation. Our focus is on finding the L_1 distance between distributions, such as the distance between the current distribution and a historical distribution from a previous month; or the distance between values observed on the East coast versus on the West coast of a large country. The L_1 distance corresponds to the total variation distance in statistics. Federated computation of other statistical distances, such as information divergences and Fréchet distance [32] are also of interest, but are outside the scope of this paper, and are subject to strong lower bounds for sketching [2, 29]. The sketch we build can be viewed as a method for sampling elements from the input distribution where items with larger contributions to the distance are more likely to be sampled and where sampling is coordinated across individuals by using a random hash function. The sketches can be combined with an addition-like operation, allowing us to build summaries of (the average of) multiple distributions, or other weighted combinations.

The resulting summaries tackle the challenge of the federated model: (a) using summaries reduces the large size of the inputs and communication costs; (b) the summaries can be combined with arithmetic-like operations to address the distributed nature of the problem; (c) in addition, these summaries can be constructed quickly with low overhead on participants, who may use weak devices (mobile devices or wearables). Privacy concerns can be addressed by (d) performing a secure aggregation of summaries [8, 12, 49], and (e) adding suitably calibrated noise to mask the true content [24]. The summaries can be understood in the context of federated computation: instead of collecting all of each user's information, the summary captures only a small amount of data, which can be further masked. They also apply data compression: reducing the communication cost for mobile clients with low bandwidth availability.

1.1 Technical overview and contributions

We now describe the objective and our results in more mathematical terms. We consider inputs that can be thought of as vectors, although they may not be presented in this format. That is, given a set of observations of items from a finite universe $\mathcal{U}$, we represent them as a vector x that records the number of occurrences of each item. Then, given any input vector x, our aim is to estimate $\|x\|_p$ (the L_p norm of x), in a way that can be distributed over multiple participants, and is amenable to privacy. We are most interested in the case when x is formed as the difference of two probability distributions, P and Q, and $p = 1$. The *total variation distance* of P and Q is given by the L_1 norm of the difference of the two probability distributions, as $D_{\mathrm{TV}}(P, Q) = \frac{1}{2}\|P - Q\|_1$. Estimating $\|x\|_1$ for $x = \frac{1}{2}(P - Q)$ gives us the total variation distance between P and Q. Since $\|P\|_1, \|Q\|_1 \leq 1$, we have $\|x\|_1 = \sum_i \frac{1}{2}|r_i - s_i| \leq 1$. In our music example, P and Q can represent the popularity of artists at different times, or in different regions. For instance, we could write p_{xcx} for the probability mass associated with the singer Charli XCX in distribution P, so $|x_{\mathrm{xcx}}|$ is her contribution to D_{TV}. If $p_{\mathrm{xcx}} = 0.03$ and $q_{\mathrm{xcx}} = 0.05$, then $|x_{\mathrm{xcx}}| = 0.01$. A key technical notion we use is L_p *sampling*, which samples elements from x according to their contribution to the L_p norm [19].

Our contributions. We define summaries that build on the core notion of L_p sampling. L_p sampling lets us preferentially sample coordinates that are more informative for the norm. We derive two estimators that accurately estimate the target norm from the sample. In detail, our contributions are:

- We present new sketch-based techniques for estimating D_{TV} in the federated setting. These build on prior L_p sampling works [5, 13] but simplify and optimize them for the L_1 case.
- We show how our sketches can be built quickly based on a statistical technique to only sample a single value instead of k, resulting in orders of magnitude speed-ups.
- We show for the first time how these sketches can be combined with privacy models, demonstrating their flexibility.
- We evaluate the utility of the sketches on real and synthetic data. We see that they obtain very strong accuracy, while being considerably faster than alternatives from prior work.

2 RELATED WORK

Federated Computation. Recent years have seen an explosion of interest in systems implementing federated learning, where an ML model is trained on labeled examples held by a number of clients and are not allowed to be directly read by a central server. Progress typically takes place over many rounds. In each round, the server broadcasts a current model, and each client proposes a model update based on gradients of model parameters. The client updates are usually combined by averaging [44]. By sharing updates rather than raw data points, clients enjoy a baseline level of privacy; ongoing research in federated learning seeks to offer more formal privacy guarantees, optimize the communication costs, better handle heterogeneous data, and so forth. [38]. Techniques like *Secure Aggregation* [12] can ensure the coordinating server only sees the final aggregation over clients and not individual updates. Outside of machine learning, there is growing interest in applying the federated model to other data analysis tasks, building on earlier efforts in private and distributed systems. The umbrella term 'federated analytics' (FA) covers the private collection of statistics on distributed data, such as building histograms and identifying the most popular items [16, 54]. Other works in FA have looked at tracking frequency distributions [20] and range queries [52], among others [9, 25].

Sketches. The notion of *data sketches* – compact summaries of data that can be manipulated with arithmetic-like operations – has been well-studied in the database community for many years [22]. Sketches can estimate the cardinality of sets [26], track the membership of sets [11], or summarize frequency distributions [17]. Sketches that capture the norm of vectors have been widely used in nearest neighbor search for over two decades [35], and form the basis of deployed similarity search systems [36]. Within federated computation, sketches have been used to reduce the communication cost and speed up convergence of federated model training [30, 47].

Several sketching techniques have been developed to estimate the L_1 or L_p norm of vectors. In particular, Indyk [34] introduced sketches based on sampling from stable distributions. Brinkman and Charikar showed that it was not possible to achieve dimensionality reduction for L_1 via embedding-based approaches, meaning that non-embedding based sketches are needed [14]. Li introduced

improved estimators based on the stable distributions technique with sparsity [40, 41]. For sketches with near-optimal space cost and fast update time, Nelson and Woodruff proposed a different hashing-based approach that uses stable distributions [45]. However, we find that these above-mentioned approaches are not very well-suited to the federated setting. Specifically, they can be complex to implement and incur large hidden constants as shown by our evaluation of stable distribution-based sketches in our empirical study (Section 5). They are also not amenable to providing a privacy guarantee: the noise added can considerably distort the results.

Our focus is on *hashing-based techniques* that can be made to run fast. Ioffe [37] introduced a modified hashing technique that can be applied to L_1 sketching that ensures that processing each item takes constant time. Andoni, Krauthgamer and Onak introduced the idea of precision sampling, which uses a reweighting based on hash values to determine a subset of items with which to build an estimator [5]. More recently, Braverman, Krauthgamer and Yang [13] propose a different "alpha sampling"-based technique for L_p norms. These two last techniques form the basis of our estimators discussed in subsequent sections. Collectively, the prior work studying sketching techniques for L_1 norms have focused on theoretical aspects, with little empirical evaluation; we seek to make them practical to implement at scale and privacy-friendly.

Privacy. Given the sensitivity of data held by clients (e.g., on mobile devices), there are strong incentives to provide formal privacy guarantees for data analysis procedures. The dominant model is the statistical notion of Differential Privacy [24], which places stipulations on the output distribution of a randomized procedure. Differential privacy guarantees have been provided for a large number of tasks, such as model training via stochastic gradient descent [1], data mining [27], census table release [3], and gathering statistics on device usage [23]. Most relevant to this work is interest in privacy and sketching [53] and counting distinct items [31, 46]. Various results are known that show that sketches for distinct items [48] and Euclidean norms [10], and more generally frequency moments (a closely related objective to L_p norms) [51] achieve a level of privacy, under the assumption that the hash functions used are secret. In our setting, the hash functions are shared among many participants, and so we seek to achieve privacy by noise addition instead. Part of our contribution in this paper is in studying the interaction between privacy and sketches for distance estimation.

3 PRELIMINARIES

3.1 Norms and Sketching

Given a vector x of dimension d, denote its L_p norm by $\|x\|_p = (\sum_{i=1}^{d} |x_i|^p)^{1/p}$. Throughout, we will be interested in frequency vectors x, where $x_i \geq 0$ denotes the frequency of an input item i, and probability vectors y such that $\forall 1 \leq i \leq d, 0 \leq y_i \leq 1$ and $\|y\|_1 = 1$. We can easily build a probability vector from a frequency vector as $y = x/\|x\|_1$, and so will not distinguish between these two notions in what follows. We write $x^{\text{tail}(t)}$ to denote the vector x with the t largest entries (in absolute magnitude) removed. We will often suppress the parameter t, and just write x^{tail}.

Table 1: Table of notation

Symbol	Meaning
$\|x\|_1$	L_1 norm of vector x
$D_{\text{TV}}(P, Q)$	Total variation distance between P and Q
f_i	frequency of item i in the data
$x^{\text{tail}}(t)$	Vector x with t largest entries removed
S	Sketch transform (represented as a matrix)
ψ	Sketch error on v is bounded by $\psi\|v^{\text{tail}}\|_1$
(ϵ, δ)-approx.	Relative error $1 \pm \epsilon$ with prob. $1 - \delta$
τ	Threshold used by heavy hitters estimator
κ	Number of samples used by top-k estimator
(ε, Δ)-privacy	Differential privacy with parameters ε, Δ
$\mathbb{1}(a)$	Indicator function: 1 if a is true, 0 otherwise
$w \odot x$	Hadamard product of x and y
$\tilde{O}(x)$	Big-O of x times a polylogarithmic factor

Definition 3.1 (Linear sketch). We say that an algorithm computes a (linear) *sketch* of a vector x if it can be written as Sx, where S is a linear mapping sampled from an appropriate random distribution.

We are concerned with finding sketches that let us approximate the L_p norm of an input vector. Specifically, we seek (ϵ, δ) approximations that provide an estimate $\hat{x}$ so that

$$(1 - \epsilon)\|x\|_1 \leq \hat{x} \leq (1 + \epsilon)\|x\|_1 \tag{1}$$

with probability at least $1 - \delta$, where the probability is taken over the randomness used to draw the sketch mapping S. We use the notation $X = (1 \pm \epsilon)Y$ as shorthand for $(1-\epsilon)Y \leq X \leq (1+\epsilon)Y$. We say estimators satisfying (1) with constant δ are "$(1 \pm \epsilon)$-estimators".

In building our sketches, we make use of *frequency sketches*, such as Count-Min sketch [21] and CountSketch [17], which estimate any v_i from vector v with additive error bounded by $\epsilon\|v^{\text{tail}}\|_1$ [22].

3.2 Federated computation model

The sketching-based methods that we present are flexible, and can be applied in a variety of distributed computational models. We next specify a concrete model, with the understanding that our results also apply to other settings.

Definition 3.2 (Federated distribution). Given a set of clients who each hold a collection of items, let f_i denote the frequency (count) of item i across the set of clients, and let $F = \sum_i f_i$. The (aggregate) probability distribution P is defined as $p_i = f_i/F$.

Definition 3.3 (Data distribution shift problem.). Given probability distributions P and Q defined in the federated setting, the data distribution shift problem is to compute an estimate of $D_{\text{TV}}(P, Q) = \frac{1}{2}\|P - Q\|_1$ by building a *sketch* of each distribution, sketch(P) and sketch(Q), where the sketches can be combined to give the estimate.

A trivial solution to this problem would be to compute the sketch as a complete description of the corresponding probability distribution. In what follows, we will evaluate the suitability of a sketch based on (i) the size of the sketch data structure (ii) the speed with which the sketch can be built from updates (iii) the accuracy guarantees of the resulting estimator. The data shift sketches that we build can be combined algebraically, so that

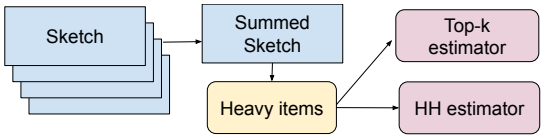


Figure 1: Querying the combined sketch.

$\alpha\,\text{sketch}(P) + \beta\,\text{sketch}(Q) = \text{sketch}(\alpha P + \beta Q)$ (see Section 4.6) Hence, in what follows, we focus on the task of using sketches to estimate $\|P\|_1$: this then solves the data distribution shift problem.

3.3 Security and Privacy

We seek both security (protection of the data while in transit) and privacy (ensuring the output does not leak sensitive information).

Definition 3.4 (Secure Aggregation (SecAgg)). Given a collection of n vectors $x^{(j)}$, their aggregation (summation) is the sum $X = \sum_{j=1}^{n} x^{(j)}$. A *secure aggregation* protocol allows the computation of X where each $x^{(j)}$ is held by a distinct client j, without revealing any intermediate values to any participant.

There are various alternatives to implement secure aggregation, such as via multi-party computation [49], communication between clients and a server [8, 12] or with secure hardware [33].

Definition 3.5 (Differential Privacy, DP). An algorithm M is said to be (ε, Δ)-differentially private if $\Pr[M(x) \in O] \leq \exp(\varepsilon)\Pr[M(x') \in O] + \Delta$ for any possible set of outputs O for inputs x, x' that are *neighboring*. In this paper, we define neighboring to be inputs that differ in the occurrence of one element, i.e., $\|x - x'\|_1 \leq 1$.

There are many standard ways to achieve differential privacy [24]. We adopt methods which proceed by noise addition: adding random noise sampled from an appropriate distribution (Gaussian or Laplace) to statistics computed by an algorithm.

The DP definition can be applied at different places in a distributed system according to the requirements and trust model adopted. If there is a fully trusted central entity, it can receive all the data from clients and perform DP noise addition itself. However, in our federated case we reduce the degree of trust, and focus on noise added by clients. In the most extreme case when clients have zero trust in other entities, each client can add noise to its own message and ensure its privacy before releasing it for (secure) aggregation: this is called the local model of DP. Since the accuracy of the result degrades due to many independent noise additions, we advocate distributed noise generation. Here, each client adds only a small level of noise, so the accumulation of all (independent) client noise samples under secure aggregation yields the desired level of privacy protection without needing to trust the central entity. This is achieved by Pólya noise [6], Discrete Gaussian noise [4], or Poisson-Binomial noise [18]. Section 4.7 explains how we can combine DP noise with our sketching approach.

4 DATA SHIFT SKETCHES

In what follows, we build up the technical basis for our approach, starting with an overview in Section 4.1. In Section 4.2, we give the basic notion of L_p-sampling by reweighting the input vector. We use this to build two different estimators for estimating data shift, a heavy hitters (Section 4.3) and a top-k estimator (Section 4.4). These make use of the same sketch, but have different properties, yielding two different algorithms presented in Section 4.5. We discuss how to implement these methods in the federated setting in Section 4.6, and how to achieve privacy guarantees in Section 4.7. The technical details in Sections 4.2, 4.3 and 4.4 are required to build up the solutions, but can be skipped on an initial read if so desired.

4.1 High-level overview

The core of our methods builds on notions of L_p sampling [19], which can be represented as a linear sketch. The starting point is to build a sketch data structure to represent a vector x and which allows us to sample an index i (and the corresponding x_i) with probability proportional to $|x_i|$, i.e., index i's contribution to the L_1 norm of x. From this, we can build estimators for the L_1 norm based on counting how many samples exceed a threshold value (the HH estimator), or looking at the weight associated with the heaviest samples (top-k estimator). The approach is suitable for the federated setting, since each client can build a sketch on local data, and these can be combined to produce a sketch for the summation of their inputs. Importantly, the resulting sketch immediately applies to measuring the data shift via total variation distance. The total variation distance between two distributions is the L_1 norm of a vector that is the difference of the two distributions. Given sketches of the two distributions, the distance is measured by negating the entries in one sketch and adding it (entrywise) to the other, then applying the estimation methods described below (see Fact 4.11).

To obtain an accurate result, we need to extract multiple samples from the data structure, and repeat the procedure hundreds or thousands of times. Doing this explicitly would be burdensome, due to the time cost of the independent repetitions. Instead, we will present a "faster" version of the sampling procedure that allows the same accuracy level to be met with the cost of only a single modified execution of the algorithm. In what follows, we develop this by first describing the 'basic' (slow) estimator, then show the final 'faster' version with equivalent or better accuracy. We follow this path for the two estimators (top-k and HH), which both operate on the same data structure summarizing the input x.

4.2 L_p sampling

The L_p sampling approach draws a random weight vector w, and computes the reweighted vector $v = w \odot x$ where $\odot$ is the Hadamard product, so that $v_i = w_i x_i$ [19]. This allows us to approximate $\|x\|_p$ from properties of v. To be of practical use, we use a hash function to define the weight vector w to obtain a compact representation that allows any entry to be accessed quickly without requiring us to store w explicitly. In what follows we also discuss how to store v in a summary data structure that is much smaller than d, so that the entire representation is the composition of a number of linear transformations, and so forms a linear sketch (Definition 3.1).

To instantiate the L_p sampling method for parameter p, we *randomly* pick the vector w so that $w_i \sim 1/(U[0,1])^{1/p}$. That is, entries are drawn as the reciprocal of uniform random values in the range $[0, 1]$, raised to the power $1/p$. We can then examine the sampling probability for an item when we take the items with highest weight.

From this, we have for a suitably large threshold τ, that the probability the absolute value of v_i exceeds τ is proportional to the p'th power of the original weight x_i, as follows:

Fact 4.1. $\Pr[|v_i| \geq \tau] = \left(\frac{|x_i|}{\tau}\right)^p$

Proof.

$$\Pr[|v_i| \geq \tau] = \Pr[|x_i|w_i \geq \tau] = \Pr\left[\frac{|x_i|}{(U[0,1])^{1/p}} \geq \tau\right] \quad (2)$$

$$= \Pr\left[\left(\frac{|x_i|}{\tau}\right)^p \geq U[0,1]\right] = \left(\frac{|x_i|}{\tau}\right)^p \qquad \square$$

This property holds provided $\tau > |x_i|$. The probability is over the randomness associated with the choice of w. In our music example, suppose we sample $w_{\text{XCX}} = 3.2$, and have $x_{\text{XCX}} = -0.01$. For $\tau = 10$, we find $|x_{\text{XCX}}|w_{\text{XCX}} = 0.032$, so XCX does not pass the threshold.

Fact 4.1 states that picking large entries of v_i is the same as sampling an index i with probability proportional to its contribution to the L_p norm of the vector x. This is a key step for the subsequent estimators. Small sketches of v enable us to do this efficiently. Importantly, because v is a linear transformation of x, v can be computed *incrementally* and *additively*. Updates Δ_i to the input vector x generate a corresponding update $v_i' = v_i + \Delta_i w_i$ which is processed by the frequency sketch. To ensure the update is efficient and every update to index i will retrieve the same value w_i, w_i is defined using a random hash function and not stored explicitly. The size of the frequency sketch is a function of an accuracy parameter, ϵ, and independent of the dimension of the input, d. Although we will treat the entries of w as independent draws for the purposes of analysis, the proof will still hold when the hash is drawn from a pairwise-independent family. From here on, we focus on the case $p = 1$, which captures the L_1 distance (total variation distance).

4.3 Heavy hitters-based estimator

Our first estimator is based closely on adapting precision sampling [5], but is simplified and streamlined due to our focus on estimating distances between distributions. We build a frequency sketch of size $O(1/\psi)$ of the vector v, so that we can estimate any v_i by $\hat{v}_i$ with error $\pm\psi\|v^{\text{tail}}\|_1$. Recall that v^{tail} is the vector v with the t largest entries removed, for a parameter t we will define later. This is equivalent (after rescaling) to finding an estimate $\hat{x}_i$ of x_i with additive error $\psi\|v^{\text{tail}}\|_1/w_i$. We will later bound $\|v^{\text{tail}}\|_1$ in terms of $\|x\|_1$ and other parameters, and choose ψ so that $\psi\|v^{\text{tail}}\|_1 \leq 1$. Hence, we can obtain $|\hat{x}_i - x_i| \leq 1/w_i$.

4.3.1 Basic HH Estimator. The basic estimator counts the number of v_i values greater than a threshold τ that we will determine below. That is, we define the estimator $X = \tau \cdot |\{i : |\hat{v}_i| > \tau\}|$. For our running example, if we set $\tau = 10$ and there are 6 artists whose (estimated) $\hat{v}_i$ value is above τ, then the estimator is $6\tau = 60$.

Lemma 4.2. *The expectation of the estimator X is $(1 \pm \epsilon)\|x\|_1$, and the variance is at most $2\tau\|x\|_1$.*

Proof. To analyze X, we apply the sampling approach of (2), and consider the probability that $|\hat{v}_i| \geq \tau$. That is, we sample i if $\hat{x}_i w_i \geq \tau$. Provided the sketch parameter ψ is set so that $(\hat{x}_i - x_i) \leq 1/w_i$, this can be written as $w_i|x_i| \pm 1 \geq \tau$, i.e., $w_i|x_i| \geq \tau \pm 1$. In other words, using $\hat{x}_i$ gives a very similar result as if we used x_i.

Algorithm 1 Update the data structure for item i with weight w

Input Item i with associated weight w

$u_i \leftarrow \text{Sample}(U[0,1])$ by hashing i

$W_i \leftarrow 1/(1-u_i^{1/k})$ ▷ For fast estimators (Sections 4.3.3, 4.4.3)

$s \leftarrow w * W_i$ ▷ Scaled weight

Sketch.Update(i, s)

$v_i \leftarrow$ Sketch.Query(i) ▷ v_i is (estimated) weight of i

FixedSizeHeap.Update$(i, |v_i|)$ ▷ Track the k largest weights

To re-express this proximity, we choose a value of τ large enough so that $\tau \pm 1 \subseteq \tau/(1 \pm \epsilon)$,[1] for some accuracy parameter $0 < \epsilon \leq 1$. This is satisfied, for example, by choosing $\tau = \frac{1+\epsilon}{\epsilon} \leq 2/\epsilon$. Then we can write the sampling condition as $w_i|x_i|(1 \pm \epsilon) \geq \tau$. Our estimator X counts τ towards the estimate for every index i where $\hat{v}_i$ is such that $|\hat{v}_i| \geq \tau$. Based on (2), we have that

$$\mathsf{E}[X] = \sum_{i=1}^{d} \tau \Pr[|\hat{v}_i| \geq \tau] = \tau \sum_{i=1}^{d} \frac{(1 \pm \epsilon)|x_i|}{\tau}$$

$$= (1 \pm \epsilon) \sum_{i=1}^{d} |x_i| = (1 \pm \epsilon)\|x\|_1 \quad (3)$$

$$\mathsf{Var}[X] \leq \sum_i \tau^2 \Pr[|\hat{v}_i| \geq \tau] = \tau^2 \sum_i \frac{(1 \pm \epsilon)|x_i|}{\tau} \leq 2\tau\|x\|_1 \quad (4)$$

$\square$

4.3.2 Accurate HH Estimator. The variance of a single estimate is quite large, so the standard way to reduce error is to take the mean of k independent repetitions of X. That is, we use k randomly drawn weight vectors $w^{(j)}$. This gives a new estimator $Y = \frac{1}{k}\sum_{j=1}^{k} X^{(j)}$.

Corollary 4.3. *Averaging $k = O(\epsilon^{-3})$ independent copies of X gives a $(1 \pm \epsilon)$ estimate of $\|x\|_1$ with probability at least $\frac{3}{4}$.*

Proof. It follows immediately that X and Y have the same expectation ($\mathsf{E}[Y] = \mathsf{E}[X] \subseteq (1 \pm \epsilon)\|x\|_1$), and Y's variance is $\mathsf{Var}[Y] = \frac{1}{k}\mathsf{Var}[X] \leq 2\tau\frac{\|x\|_1}{k}$. Applying Chebyshev's inequality,

$$\Pr[|Y - \mathsf{E}[Y]| \geq \delta] \leq \frac{\mathsf{Var}[Y]}{\delta^2} \leq \frac{1}{k\delta^2} 2\tau\|x\|_1 \leq \frac{4}{k\epsilon\delta^2}$$

where we use $\|x\|_1 \leq 1$ and $\tau \leq 2/\epsilon$. We can make this probability of a poor estimate at most a constant, say $\frac{1}{4}$, by setting $k \geq \frac{16}{\epsilon\delta^2}$. For our guarantee, we choose $\epsilon = \delta$, which sets $k = O(\epsilon^{-3})$. $\square$

The success probability can be further amplified using standard techniques (taking the median of $O(\log 1/\delta')$ repetitions of the overall procedure) to $1 - \delta'$. However, this construction may be slow in practice, so we next present a faster accurate estimator.

4.3.3 Fast HH Estimator. Naively, taking the mean of k repetitions would incur a slowdown by a factor k, as we would need to compute vectors $v^{(1)} \ldots v^{(k)}$, based on weights $w^{(1)} \ldots w^{(k)}$. The key idea is to use a single sample to *simulate* the effect of all these k repetitions at once. This concept is used in prior work on sampling [5, 50] – here, we apply it to L_1 estimation.

[1] Recall that $x \pm y$ is shorthand for the range $[x - y, x + y]$ so $x \pm y \subseteq w \pm z$ means that $w - z \leq x - y$ and $x + y \leq w + z$.

Algorithm 2 Fast heavy hitters estimation procedure

Input Parameter ϵ, Heap and Sketch
Output HH estimate for the L_1 of the processed input

1: $\tau \leftarrow 2/\epsilon$; $L \leftarrow 0$ ▷ Initialize values
2: **for all** items i in Heap **do**
3: $\hat{v}_i \leftarrow$ Sketch.Query(i)
4: **if** $\hat{v}_i > \tau$ **then** $L \leftarrow L + \tau(1 + \frac{k-1}{k} \cdot \frac{\hat{v}_i/\tau - 1}{W_i - 1})$ ▷ Via eqn. (6)
5: **return** (L)

Lemma 4.4. *The information needed to build the estimator Y can be gathered using $O(1)$ time to process each update.*

Proof. We analyze what information is needed to build the estimator Y, and argue that we can represent this with a single sample, deferring other sampling decisions until estimation time. We note that for each index i, it suffices to track only the contribution from the *largest* weight $W_i = \max_{j=1}^{k} w_i^{(j)}$. That is, we represent the whole set $\{w_i^{(1)}, w_i^{(2)} \dots w_i^{(k)}\}$ by just storing information on W_i. The max of k uniform random variables is distributed as $U[0,1]^{1/k}$, and so the min of k is (symmetrically) distributed as $(1 - U[0,1]^{1/k})$. Hence, we directly sample $W_i \sim 1/(1 - U[0,1]^{1/k})$, and use this to update the sketch in time $O(1)$, as shown in Algorithm 1.

To complete the proof, we use the information stored about W_i to reconstruct the information needed at query time to build the estimator Y. Recall that our estimator Y is built by computing

$$Y = \frac{1}{k} \sum_j X^{(j)} = \frac{1}{k} \sum_j \tau |\{(i,j) : \hat{v}_i^{(j)} \geq \tau\}|. \tag{5}$$

We can write the contribution to the sum from item i as

$$Y_i = \frac{1}{k} \sum_j X_i^{(j)} = \frac{1}{k} \sum_j \mathbb{1}(\hat{v}_i^{(j)} \geq \tau).$$

Observe that for an index i, if we know W_i, there is enough information to sample the other $k-1$ weights for i, by conditioning on the value of W_i: sample $k-1$ other values, conditioned on the fact that they are smaller than W_i, by picking uniform random values for $u_i^{(j)}$ that are in the range $(1/W_i, 1]$. More directly, since we need the count of the number of cases that cross the threshold τ, we can work with the expectation instead. See [5] for an argument that using the expectation of such a random variable only serves to decrease the variance. Conditioned on W_i, we count the number of the other $k-1$ samples that exceed τ. Working in the space of the uniform random values $u_i^{(j)}$, the smallest of these corresponds to W_i, and the remaining values are distributed uniformly over $(1/W_i, 1]$. The condition $\hat{x}_i w_i^{(j)} \geq \tau$ maps onto $u_i^{(j)} \leq \hat{x}_i/\tau$. Hence, the probability for each sample of being picked is $\frac{\hat{x}_i/\tau - 1/W_i}{1 - 1/W_i} = \frac{\hat{x}_i W_i/\tau - 1}{W_i - 1}$. The contribution to the mean from each of the $k-1$ samples that meet this criterion is $\frac{1}{k}$, so the expected contribution is $\frac{k-1}{k} \frac{\hat{x}_i W_i/\tau - 1}{W_i - 1}$. This lets us compute the estimator (also shown in Algorithm 2) as

$$Y_i = \begin{cases} \frac{\tau}{k}\left(1 + (k-1)\frac{\hat{x}_i W_i/\tau - 1}{W_i - 1}\right) & \text{if } W_i |\hat{x}_i| \geq \tau \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

□

4.3.4 Bound on $\|v^{\text{tail}}\|_1$. Next, in order to set the sketch parameter ψ, we analyze and bound the magnitude of v^{tail}.

Fact 4.5. *With constant probability, $\|v^{\text{tail}}\|_1 = O(k)$.*

Proof. The value of $\|v^{\text{tail}}\|_1$ is determined by the n values of W_i that are sampled to make v, where as before $W_i = \max_j 1/u_i^{(j)}$.

First, observe that there cannot be too many very large values in $v^{(j)}$. Set a threshold $\theta = 10\|x\|_1$. Then, $\Pr[v_i^{(j)} > \theta] = |x_i|/\theta$ (using (2)), so the expected number of entries in $v^{(j)}$ that exceed θ is bounded by $\sum_{i,j} |x_i|/\theta = k\|x\|_1/\theta = k/10$. By Markov's inequality, there are at most k such large values with probability at least 0.9.

Now consider the norm of the 'tail' of the vector after removing entries with weight more than θ. Let $V_\theta = \sum_{i,j: v_i^{(j)} \leq \theta} (v_i^{(j)})^2$. Then

$$\begin{aligned} \mathsf{E}[V_\theta] &= \mathsf{E}\left[\sum_{i,j: v_i^{(j)} \leq \theta} x_i^2 (w_i^{(j)})^2\right] \\ &= k \sum_i x_i^2 \int_1^{\theta/|x_i|} y^2 \Pr[w_i^{(j)} = y]\, dy \\ &= k \sum_i x_i^2 \int_1^{\theta/|x_i|} 1\, dy \\ &\leq k \sum_i x_i^2 (\theta/|x_i|) = k\theta\|x\|_1 = 10k\|x\|_1^2 \end{aligned}$$

By Markov's inequality, we have that $\Pr[V_\theta > 100k\|x\|_1^2] \leq \frac{1}{10}$.

Define the vector v^{tail} as the vector v with the k largest elements removed. By the above argument, with constant probability,

$$\begin{aligned} \|v^{\text{tail}}\|_1 \leq \|v^{\text{tail}}\|_2^2 &= \sum_{i: \max_j v_i^{(j)} \leq \theta} \max_j (v_i^{(j)})^2 \\ &\leq V_\theta = O(k\|x\|_1^2) = O(k). \end{aligned}$$

□

4.3.5 Space and time cost. We bound the size of the sketches needed. Frequency sketches of size $\tilde{O}(1/\psi)$ offer an error guarantee of at most $\psi\|v^{\text{tail}}\|_1$. Our error bound sets this equal to 1, so rearranging, the space bound is $\tilde{O}(\|v^{\text{tail}}\|_1) = \tilde{O}(k)$ for the sketches. When $k = O(\epsilon^{-3})$, the total space is $\tilde{O}(\epsilon^{-3})$. Thus,

Theorem 4.6. *The Fast HH-based estimator builds sketches of size $\tilde{O}(\epsilon^{-3})$, and takes time $O(1)$ to process each update. These sketches allow us to form a $(1 \pm \epsilon)$ estimate for the L_1 norm.*

4.4 Top-k-based estimator

Our second estimator uses the same basic data structure as that for the fast HH estimator, but instead of counting the number of estimates above a threshold, it finds the k'th largest weight. The basic version of this estimator corresponds to the approach of [13] for the L_1 norm, which we then modify and further adapt to be faster. We first analyze the behavior based on the exact vectors $v^{(j)}$.

4.4.1 Basic top-k Estimator. As above, consider an estimator based on vectors $v^{(j)}$, formed as the product of inputs x with k weight vectors $w^{(j)}$, each entry of which is sampled as $1/(U[0,1])$. Now we define our estimate Z to be the $k/2$'th largest entry across all the $v^{(j)}$ vectors, scaled by $1/2$. In our running music example, if $w_{\text{XCX}} = 3.2$ and $x_{\text{XCX}} = -0.01$ (as before) then we have $|x_{\text{XCX}}| w_{\text{XCX}} = 0.032$. If

it happens that this is the $k/2$th largest entry, then we estimate $\|x\|_1 = 0.016$.

Lemma 4.7. *With probability at least $\frac{4}{5}$, we have that $Z \in (1 \pm \epsilon)\|x\|_1$ for k at least $30\epsilon^2$.*

Proof. To analyse Z, observe that it will be a good estimate if there are not more than $k/2$ entries across the $v^{(j)}$ vectors that are higher than $2(1+\epsilon)\|x\|_1$, and not fewer than $k/2$ entries that are larger than $2(1-\epsilon)\|x\|_1$. Consider entries that are higher than $2c\|x\|_1$ for c that is close to 1, and let Z_c denote the number of such entries. By (2), we have that $\Pr[|v_i| \geq 2c\|x\|_1] = \frac{|x_i|}{2c\|x\|_1}$. Then,

$$\mathsf{E}[Z_c] = \sum_{i=1}^{n}\sum_{j=1}^{k}\frac{1}{2c}\frac{|x_i|}{\|x\|_1} = \frac{k}{2c} \text{ and } \mathsf{Var}[Z_c] \leq \mathsf{E}[Z_c] = \frac{k}{2c}.$$

$$\text{By Chebyshev's ineq., } \Pr[|Z_c - \mathsf{E}[Z_c]| \geq t] \leq \frac{\mathsf{Var}[Z]}{t^2} \leq \frac{k}{2ct^2} \tag{7}$$

We use this to analyze $\Pr[Z_{1+\epsilon} > k/2]$, since for $c > 1$,

$$\Pr\left[Z_c - \tfrac{k}{2c} > \tfrac{k}{2} - \tfrac{k}{2c}\right] \leq \Pr\left[Z_c - \mathsf{E}[Z_c] > \tfrac{k}{2}\tfrac{c-1}{c}\right] \leq \frac{4kc^2}{2ck^2(c-1)^2} = \frac{2c}{k(c-1)^2} \tag{8}$$

For $c = (1+\epsilon)$, this yields a probability of $\frac{2(1+\epsilon)}{k\epsilon^2} \leq 4/k\epsilon^2$. Similarly, for $\Pr[Z_{1-\epsilon} < k/2]$, we have for $c < 1$,

$$\Pr\left[\tfrac{k}{2c} - Z_c > \tfrac{k}{2c} - \tfrac{k}{2}\right] \leq \Pr\left[\mathsf{E}[Z_c] - Z_c > \tfrac{k}{2}\tfrac{1-c}{c}\right] \leq \frac{4kc^2}{2ck^2(c-1)^2} = \frac{2c}{k(c-1)^2} \tag{9}$$

Substituting $c = (1-\epsilon)$, this probability is $\frac{2(1-\epsilon)}{k\epsilon^2} \leq 2/k\epsilon^2$.

Summing both these probabilities yields $6/k\epsilon^2$. Hence, if we set $k \geq 30/\epsilon^2$, this bound on the estimate being good is at least $\frac{4}{5}$. □

4.4.2 Smoothed top-k Estimator. We propose a smoothed version of the estimator, using a range of κ entries around the $k/2$'th largest.

Lemma 4.8. *Let Z' be the estimate formed by picking any item whose rank is between $k/2$ and $(1+\epsilon/2)k/2$ in the sorted order, for $k = O(1/\epsilon^2)$. Then, with constant probability, Z' is a $(1 \pm \epsilon)$-approximation of $\|x\|_1$.*

Proof. We consider the entries between $k/2$ and $(1+\epsilon/2)k/2$ in the sorted order, and modify the above argument to consider the probability that there are not fewer than $(1+\epsilon/2)k/2$ entries that are larger than $2(1-\epsilon)\|x\|_1$. Adapting (9), we obtain for $c = (1-\epsilon)$

$$\Pr\left[\tfrac{k}{2c} - Z_c > \tfrac{k}{2c} - \tfrac{(1+\frac{\epsilon}{2})k}{2}\right] \leq \Pr\left[\mathsf{E}[Z_c] - Z_c > \tfrac{k(1-(1+\frac{\epsilon}{2})c)}{2c}\right]$$
$$\leq \frac{2c}{k(1-c(1+\epsilon/2))^2} = \frac{2(1-\epsilon)}{k(\epsilon/2(1+\epsilon))^2} = \frac{8(1-\epsilon)}{k\epsilon^2(1+\epsilon)^2} \leq \frac{8}{k\epsilon^2}$$

This ensures there are $\Omega(\epsilon k) = \Omega(\sqrt{k})$ items in the target range. □

This result gives some additional flexibility in picking an estimator: we can pick any of the items in this window, or take the mean of multiple items in the window, and still obtain an accurate estimate. In our experiments, we take the mean of the items in the range $[k/2, k/2+\kappa]$, where κ denotes the size of this window.

Algorithm 3 Fast top-k estimation procedure

Input Parameter k, Heap and Sketch
Output Top-k estimate for the L_1 of the processed input

```
M ← []                                          ▷ Initialize empty list
for all items i in Heap do
    v̂_i ← Sketch.Query(i)
    x̂_i ← v̂_i / W_i                  ▷ Extract estimated x_i, via Section 4.2
    M.Append(x̂_i)
    for j ∈ [k − 1] do
        M.Append(x̂_i((W_i − 1)U[0, 1] + 1)/W_i)  ▷ Via eqn. (10)
return (M.SortDecreasing())[k/2]
```

4.4.3 Fast top-k Estimator. Naively implementing the estimator Z requires us to repeat the process k times in parallel for each update. As for the heavy-hitters based estimator, we reduce update cost by computing the largest weight W_i only, using Algorithm 1 again.

Lemma 4.9. *The information needed to build our estimator Z' can be gathered using $O(1)$ time to process each update.*

Proof. As in Lemma 4.4, the max weight $W_i \sim 1/(1-U[0,1]^{1/k})$, so we can track x_iW_i via sketches. At query time, identify the $k/2$ largest values of x_iW_i. For each of these i, compute $u_i^{(j)}$ for $j = 2 \ldots k$, conditioned on $u_i^{(j)} > u_i^{(1)}$, where $u_i^{(1)} = 1/W_i$. That is,

$$u_i^{(j)} \sim U[1/W_i, 1] \sim \tfrac{1}{W_i} + U[0,1](1-\tfrac{1}{W_i}) \sim \tfrac{(1+(W_i-1)U[0,1])}{W_i} \tag{10}$$

The estimator is built by "backfilling" the values that would have been inserted in the naive estimator, by finding $\hat{x}_i = (x_iW_i)/W_i = v_i/W_i$, and making a list with all $\hat{x}_i u_i^{(j)}$ values inserted (Algorithm 3). The top-k estimate Z' is applied to the reconstituted list of values. The correctness follows by invoking the principle of deferred decisions to sampling of $u_i^{(j)}$ at query time. □

4.4.4 Top-k estimate using approximate weights and sketches. The discussion so far assumes that the top-k estimator uses rescaled values of the exact weights. To implement the estimator in the federated setting, it must instead use frequency sketches to summarize the weight values. This means the estimator takes an approximate value for the kth largest entry of v, requiring an updated proof.

Theorem 4.10. *The top-k estimator builds sketches of size $\tilde{O}(\epsilon^{-3})$, and takes time $O(1)$ to process each update. These sketches allow us to form a $(1 \pm \epsilon)$ estimate for the L_1 norm.*

Proof. First, assume that for every entry in vector v, the frequency sketch estimates $\hat{v}$ such that $|v - \hat{v}| \leq \epsilon'$, for some ϵ'. Observe that the ℓ'th largest value from $\hat{v}$ is close to the ℓ'th largest value from v, for any ℓ. Assume for convenience that the values are re-indexed so that $v_1 > v_2 > v_3 \ldots$. Now consider $\hat{v}_\ell$. There are at most ℓ entries in v such that $v_i \geq v_\ell$, so $\hat{v}_\ell$ cannot be more than $v_\ell + \epsilon'$ (since $\hat{v}_\ell$ being the ℓ'th largest implies that there are ℓ entries in $\hat{v}$ that are as big as $\hat{v}_\ell$). Similarly, $\hat{v}_\ell$ cannot be less than $v_\ell - \epsilon'$, since there are no more than ℓ entries that are as big as v_ℓ in v. Hence, $|\hat{v}_\ell - v_\ell| \leq \epsilon'$.

Combining this with our estimator, the frequency sketch parameters must be set so that the first assumption is met, i.e., all entries of v have error proportional to ϵ. By a standard analysis, sketches

of size s obtain an error bound of $\|v^{\text{tail}}\|_1/s$ [22]. Using Lemma 4.5 above, $\|v^{\text{tail}}\|_1 = O(k)$ (with constant probability) means that the sketch size is set to $\tilde{O}(k/\epsilon)$. Since we set $k = O(1/\epsilon^2)$, the total space is $\tilde{O}(1/\epsilon^3)$ to obtain additive error of ϵ for $\|x\|_1$. □

Comparing Theorem 4.10 with Theorem 4.6, we see that the space and update costs for these two estimators are the same. Importantly, both estimators use the same sketch: while the estimation algorithms differ, the sketching procedure is the same for both. This is shown schematically in Figure 1. Hence, we compare the accuracy of estimates built using these two methods in our experiments.

4.5 Pseudocode and Algorithmic Summary

Algorithm 1 lists out the pseudocode for updating the information needed for the two (fast) estimators. It uses a frequency Sketch data structure to summarize weights, and a (fixed-size) Heap to track the items with the largest weights. For each update i that arrives, Lines 1-2 compute W_i (based on hash functions on i, to ensure that W_i is the same every time an update to x_i arrives), and Lines 4-6 update the sketch and heap based on the updated contribution to v. The heap is used to make queries faster: as items are updated, it keeps track of the k heaviest (estimated) weights seen so far. This avoids probing the estimated weights of all possible items at query time. The size of this heap is bounded by k in both cases. The HH estimator needs to recall only those items larger than the absolute value $\tau = 2/\epsilon$. Recall that we showed the bound that $\|v^{\text{tail}}\|_1 = O(k\|x\|_1^2)$ with constant probability. Consequently, the number of items above this threshold is at most $k + O(k\epsilon\|x\|_1^2) = O(k)$, using that $\|x\|_1 \leq 1$ in our setting. Hence, keeping a heap of fixed size $H = O(k)$ will allow us to track the heavy items. Meanwhile, the top-k-based estimator just needs to track the $H = O(k)$ largest estimated v_i values in order to have the information necessary to build the final estimator. In both cases, we can maintain these v_i values in a maxheap, truncated to hold H values. Every time we perform an update to item x, we can test its current estimated value in the current sketch, and add it to the heap (or update its weight in the heap) if necessary (Algorithm 1, line 6).

The estimators are computed directly from the information stored in Sketch and Heap, following the mathematical definitions in Section 4.3 and 4.4. These are spelled out in Algorithms 2 and 3, respectively. These assume access to W_i on demand, by applying the same hashing procedure as in Algorithm 1 lines 1-2.

4.6 Distributed Computation

When the methods are applied to distributed data, as is the case in the federated setting, it is necessary to combine information from multiple, distributed computational entities. If each entity has kept a sketch of their (scaled) inputs, along with a heap of the current heavy items, they are merged in a straightforward way. The sketches are added, entrywise, and scaled appropriately (e.g., if uniformly combining m sketches, we weight each sketch by $\frac{1}{m}$). For D_{TV}, we subtract the sketches, entrywise.

Fact 4.11. *The sketched information can be combined linearly.*

Proof. Recall that we can write the derived vector $v = w \odot x$, where w is the weight vector, and x is the input vector (Section 4.2). The frequency sketch is a linear map S applied to v (Definition 3.1). Hence, the overall sketch is $Sw \odot x$. Given vectors x and y, and scalars α and β, it follows by linearity that

$$\begin{aligned}\alpha(Sw \odot x) + \beta(Sw \odot y) &= (Sw \odot \alpha x) + (Sw \odot \beta y)\\ &= S(w \odot \alpha x + w \odot \beta y) = Sw \odot (\alpha x + \beta y).\end{aligned}$$

In other words, we can algebraically combine the sketches to obtain the exact sketch of the correspondingly manipulated inputs. □

To combine the heaps, the union of all items from each individual heap are extracted. The combined sketch is used to estimate each of their combined weights, which are used to insert into a fresh fixed-size heap. A challenge arises when some items may have negative weight (corresponding to finding the difference between two distributions): this could mean that the true aggregate weight of the item becomes much smaller. The consequence is that items with smaller weight might have been overlooked for inclusion in the new heap, but should still count towards the estimate. For the heavy hitters approach, this is not a concern. Items contribute towards the estimator only if their weight is above a fixed (data independent) threshold, so the addition or removal of other items does not affect whether they cross this threshold. For the top-k estimator, we can observe that, if we are satisfied with an additive approximation of the L_1 distance, then we can safely overlook all items whose weight is less than some low threshold. Similar to the above argument for the heap size, dropping items whose weight is less than γ in absolute magnitude will only cause the estimate to be off by at most γ. Hence, we just need ensure all items with magnitude above γ are retained, of which there are at most $O(k/\gamma)$. Thus, in summary,

Corollary 4.12. *The HH and top-k estimators can be built over federated distributions (Definition 3.2) with the same guarantees as Theorems 4.6 and 4.10, respectively.*

4.7 Privacy and Security

We show how to achieve privacy and security guarantees. First, we show how to achieve our main result, of differential privacy protection based on client noise addition. Then, to show the flexibility of our approach, we show how our approach performs when security is needed but differential privacy is relaxed.

4.7.1 Main result: Differential Privacy via local or distributed noise. Our method enables a differential privacy guarantee in the federated setting by having the clients add noise to their data before it is processed into a sketch. We first describe the process for local differential privacy (LDP) at the item level (Section 3.3). That is, we assume that the client's input is a histogram of items observed by that client, and the noise is added to introduce sufficient uncertainty about the presence or absence of a particular item in the client's input such as a particular artist in our running example. In other words, if X denotes the client's histogram of item counts, input histograms X and X' are neighboring if $\|X - X'\|_1 \leq 1$.

To achieve (ε, Δ)-differential privacy each client first adds independent noise to each count in their histogram using the "stability-based histograms" technique [7]. Importantly, this uses a truncated noise distribution which has finite support – for example, a symmetric geometric distribution with parameter $\exp(\varepsilon)$, truncated so that the residual probability mass is at most Δ. This ensures that

the expected noise is a constant, and is of bounded magnitude T. A threshold is applied to the noisy histogram, so that counts with noisy weight T or less are removed. This means that any input items with true count 0 cannot be included in the thresholded output. This lets us work with arbitrarily large domains of input items without introducing arbitrarily large amounts of noise. As a result,

Theorem 4.13. *The fast sketch estimator with client noise addition provides (ε, Δ)-item level local differential privacy in the federated model for the data distribution shift problem (Defnition 3.3).*

For constant values of ε, the magnitude of the noise is constant, i.e., $\Theta(1)$. If a client holds m distinct values, then the noise on each is $\Theta(1)$, totalling up to $\Theta(m)$. After normalization, the contribution to the error in the L_1 distance estimation is $\Theta(1)$, i.e., potentially large. However, when the client has a less uniform input value distribution, we expect the error to reduce, to $O(d/m)$, where d is the number of distinct values held by the client, while m is the (larger) number of values. A first crude initial test on synthetic data (comparing two Normal distributions) bears this out: using $\varepsilon = 3$, and setting the threshold to 4, we observe a change to the L_1 norm of less than 1%, for $\Delta \sim 10^{-6}$. More in-depth results are reported in the experimental study in Section 5.

The exact same outline is used to instantiate the distributed differential privacy version, where each client again adds some noise to their input. However, this time, the magnitude of the noise is much less (Section 4.7). The noise volume is now calibrated so that the sum of noise from all clients is equivalent to the noise that would have been added in the central setting. To ensure that the server only sees the aggregated output of the computation with the full noise, Secure Aggregation (Definition 3.4) should be used to combine the sketches from the clients via addition [12].

4.7.2 Security without noise addition. Our sketches can also be applied in other privacy models. Next, we describe how to use the sketches in a distributed setting when DP is not required.

Here, there are m clients who each hold a distribution P_ℓ, and who wish to cooperate to compute the difference between a given proxy distribution Q and the (weighted) sum of the client distributions, as $D_{\mathrm{TV}}(P, Q) = \| \sum_{\ell=1}^{m} P_\ell - Q \|_1$. In this situation, the clients send their sketches to the server, who will sum them together as described in Section 4.6, and probe them to build the final estimate. This instantiates the model of "Federated Analytics" [9, 25], where each client reveals only a partial view of their data in order to compute the desired function, rather than sharing their data in its entirety. However, if the messages from client to server were accessible to a curious third party, this would reveal some information about the inputs. The observer would not learn every item and weight in the client's distribution, but they may see some sensitive item identifiers from the client's distribution. If the hash function used to define the weights w is public, an observer could also infer approximate original weights of items from the client's distribution.

To address this, we ensure that messages from clients to server are encrypted (to prevent observation by external parties), and that the server is trusted to receive the sketches and perform the aggregations without "snooping" on the intermediate results. This trusted server can compute the required estimate, and release it (possibly with noise). To reduce the level of trust in the server, the clients cooperate to mask the identity of the items that they are reporting. Specifically, the clients agree on a secret (salted) one-way hash function that they apply to the identities of their items, but which is kept secret from the server. Because the labels attached to the items are unimportant for the L_1 distance, relabelling them does not change it. Then, the server is still able to compute the necessary estimate, but is oblivious as to which items are present in the inputs. This model is suitable when no clients will collude with the server to give up the details of the hash function.

Finally, Secure Aggregation (Definition 3.4) is used to produce a combined sketch, without any information about which client contributed what. However, it is still necessary to use (masked) item identifiers to extract the estimate, so an additional secure primitive is needed to gather the union of the item identifiers from the clients. The options are to use anonymous messages (using anonymous credentials [39]), or by secure aggregation over one-hot vector encodings. Together, this information is sufficient to probe the aggregated sketches, and build either of the estimators.

5 EXPERIMENTAL STUDY

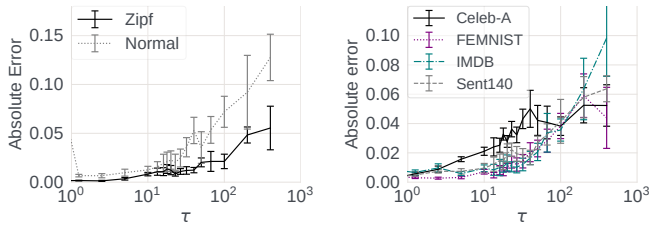
We implemented the different sketching techniques in Python. The methods we implemented were:

- Our "fast estimators" using the HH and top-k approaches described in Sections 4.3 and 4.4.
- The α-sketches of Braverman *et al.* [13] which correspond to the basic version of the top-k approach (Section 4.4).
- The same α-sketch but using the HH estimation procedure, i.e., the accurate but slow HH estimator of Section 4.3.
- The Very Sparse Stable Random Projections ("Sparse") sketch due to Li [40]. These use hashing to sample a β fraction of entries, then compute the projection of these by symmetric Pareto distributions, repeated k times. The estimate is formed by taking the scaled geometric mean of the k sampled scaled sums. We default to $\beta = 0.05$ and $k = 100$ for this sketch, the same values as in [40].

Our experiments are arranged in three parts. First, we see how best to set the parameters of our fast sketches. Second, we compare against the alternatives (α-sketches and sparse sketches) in terms of both speed and accuracy. Third, we evaluate across a range of real and synthetic data both with and without privacy imposed, to test the consistency of our findings. For the initial tests, we evaluate accuracy when there is no privacy noise addition, i.e., cases where Secure Aggregation is sufficient to protect the data; in our later tests we add privacy noise in the local model (so all client messages have privacy protection before aggregation).

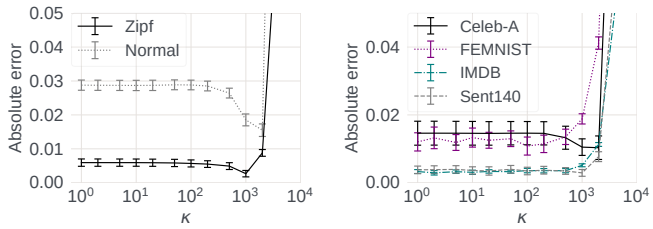
We evaluate on both real and synthetic data. The synthetic data is sampled from two Zipfian distributions with different skewness parameters (default: skewness 1.2 and 1.4), resulting in a measurable D_{TV}. The default domain size for the synthetic data is 350,000. The real data are standard benchmark datasets from the ML and Federated Learning communities [15]. In each case, we map the raw examples into frequency vectors to act as distributions.

Sent140 [28]. The Sent140 dataset is a collection of 1.6M tweets, each tagged as positive or negative in sentiment. Each word is hashed to a 20 bit range, and the dataset was divided up variously



(a) Varying τ, synthetic data **(b) Varying τ, Sent140 data**

Figure 2: Calibration experiments for the HH estimator



(a) Varying κ, synthetic data **(b) Varying κ, Sent140 data**

Figure 3: Calibration experiments for the top-k estimator

as: (i) distribution of word use in positive reviews vs. in negative reviews (ii) word use between two clusters of users.

FEMNIST [15]. The FEMNIST dataset is a collection of 803,000 handwritten characters in greyscale. We map the digits into intensity vectors, encoding the brightness in different regions of the image. These are formed into pairs of distributions in three ways: (i) intensity distribution of odd digits vs. even digits; (ii) intensity distribution of digits with more curves (0, 3, 6, 8, 9) vs. digits with more straight lines (1, 2, 4, 5, 7); (iii) splitting the digits into two disjoint halves by author.

Celeb-A [42] The Celeb-A dataset is a collection of 200,000 color images of celebrities with a variety of tags. We map these into vectors of luminance and chroma values, and diivide them based on the labels: (i) males vs. females; (ii) old vs. young; (iii) smiling vs. not smiling.

IMDB [43]. The IMDB review sentiment database is a collection of 50,000 movie reviews, divided into positive and negative classes. Each word is hashed to a 20 bit range, and the dataset was divided up variously as: (i) distribution of word use in positive reviews vs. in negative reviews (ii) word use in two disjoint halves.

In each experiment, we use the various sketch techniques to estimate the true total variation distance (D_{TV}) between each pair of input distributions (building a sketch for each distribution then combining them to estimate the difference), and compare this to the ground truth by exhaustively building the exact distribution. We measure and report the absolute difference between the estimated and exact distances; ideally, we want this to be as close to zero as possible for an accurate estimate. We perform 5 repetitions of each experiment, and show the standard error via error bars.

Our experiments are performed on a single machine with an Intel Xeon CPU with 3.2GHz cores and 48GB memory. They fully simulate the work of the clients and the server in the protocol, and account for the communication, storage, and computational costs incurred by all parties. The plots focus on the accuracy and the client side costs (space and time), since the server-side costs are negligible ($\ll$ 1MB storage and $\ll$ 1s time).

5.1 Initial parameter setting

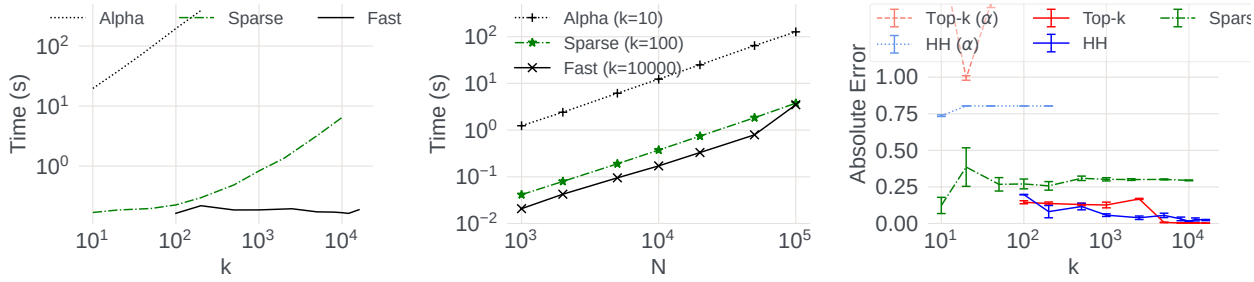
Our initial set of experiments determine suitable values for the key parameters: τ for the heavy-hitters estimator, and κ for the (smoothed) top-k estimator (Section 4.4.2). Importantly, the goal here is to find a parameter choice that is robust to use over the range of different datasets, **not** to tune τ and κ for each dataset. We ran experiments to estimate the TV-distance between two Zipfian distributions ($D_{\mathrm{TV}} \approx 0.16$), two Normal distributions ($D_{\mathrm{TV}} = 0.31$) and between the real datasets (see Table 2), and measured the absolute error when applying the different estimators. Figure 2 shows that there is a range of τ values for which the error is low and stable, allowing us to confidently choose any setting in this range. On synthetic data (Figure 2a), the results show similar results when $\tau \in (4, 50)$. Here, the error is quite small: around 0.01 for the preferred choice of τ, corresponding to a very small deviation from the true distance. On real data (Figure 2b), choices of τ in the range 1–10 are comparable in terms of accuracy. After these tests we use a default of $\tau = 5$ as a robust choice for subsequent experiments.

Figure 3 shows the top-k estimator as we vary κ, which indicates the number of estimates around the $k/2$th largest that we average together (see Section 4.4.2). Although increasing κ from 1 upwards does not significantly change the accuracy, on the synthetic data (Figure 3a), there is a local optimum close to $\kappa = 1000$ (here, $k = 10000$), then there is a rapid fall in accuracy for larger values of κ. In what follows, we use $\kappa = 100$ (taking the average of 100 entries for the top-k estimator). The take-away from these tests is that good results can be obtained from the estimators. While there is not a single optimal choice for the parameters κ and τ, we can find values that are a good choice across a range of inputs.

The space of the sketch depends on k. Each sketch consists of a fixed-size heap tracking the k largest estimated frequencies, and a CountSketch [17] of size proportional to k. Concretely, the heap is an array of exactly k words, and the CountSketch uses $3k$ words, plus some constant overhead. With 32-bit words, the space is $16k$ bytes (plus some bookkeeping), so choosing $k = 10,000$ requires 160KB of space. This is our default k value for the fast sketches.

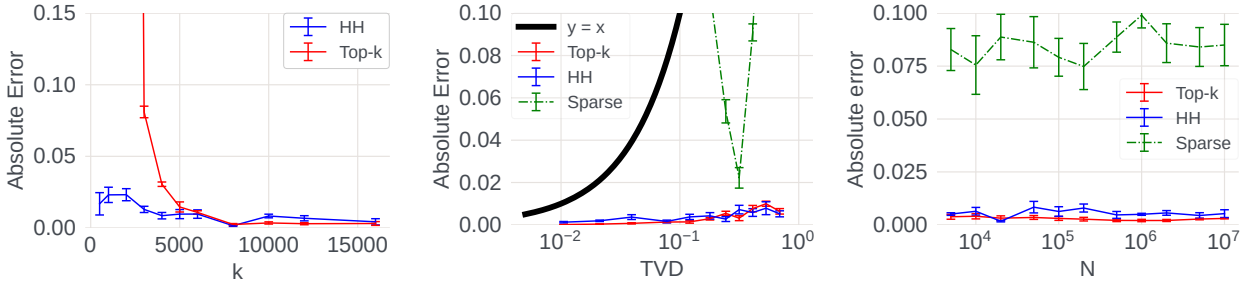
5.2 Comparison to prior work

Figure 4 compares our fast sketches with prior work on time cost and accuracy and shows prior theoretical work is not competitive in practice. For the FEMNIST data, Figure 4a shows time as a function of sketch size k (tests on other datasets showed similar results). Here, the α-sketching approach is up to three orders of magnitude slower even for small k and worse for large k since time costs grow linearly with k. Although sparse sketching approach is much faster, its cost also grows linearly with k, while our fast sketches have a running time independent of k. Figure 4b fixes k for each method



(a) Time to process FEMNIST data **(b) Time as FEMNIST data size increases** **(c) Accuracy on FEMNIST data**

Figure 4: Speed and accuracy comparison with prior work



(a) Varying size parameter k **(b) Varying L_1 difference** **(c) Varying number of samples N**

Figure 5: Testing on synthetic data

and varies the size of the input N. Our fast sketches stay the fastest, and the time cost grows linearly with N.

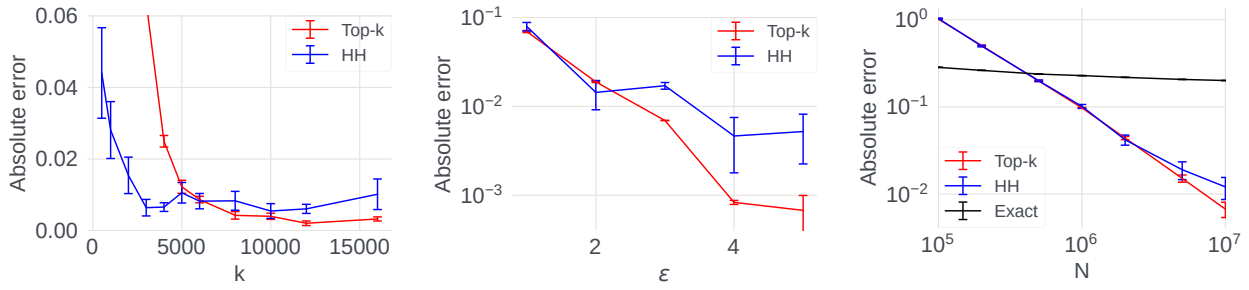
Figure 4c shows that only the fast sketches are able to achieve acceptable accuracy but they require somewhat large values of sketch size k. For small k in the range 10-200, the α-sketches achieve average error of greater than 0.5, which is unusable as D_{TV} takes values between 0 and 1. We did not evaluate the α-sketches for larger k due to their extremely high computation cost. The sparse sketching approach achieves error around 0.25 on this data, which is better but still poor. Furthermore, their accuracy did not show meaningful improvement even as we increased k to 10,000. For k large enough (5,000-10,000), the fast sketches have better accuracy than all other sketches and achieve an acceptable error close to 0. Hence we conclude that the α-sketches are dominated in both speed and accuracy, and so can't be used in practice. The sparse sketch is comparable in terms of time cost for moderate k (around 100), but does not show accuracy improvements for k in the range 10^2-10^4. In what follows, we show a few further tests with the sparse sketch, but the accuracy is rarely strong enough to be competitive.

5.3 Accuracy results on synthetic data

Having established that our fast sketches achieve usable results, we study their accuracy as parameters of the sketches and of the input are varied (Figure 5). Figure 5a shows the impact of varying the sketch size, k. As expected, increasing k tends to improve accuracy with the top-k approach needing a fairly large value of $k \geq 5000$ to obtain good accuracy. The HH estimator also improves as k increases but is less accurate than top-k except for smaller k values.

In Figure 5b, we generate Zipfian distributions with a range of skewness parameters to obtain pairs of distributions with a range of D_{TV} values, from close to zero (very similar) to one (completely disjoint) and show the error of our sketches on a log scale with respect to the true D_{TV} (heavy line). The heavy line is equivalent to ignoring the sketches and always guessing that the distance is zero. The plot demonstrates that the error is small relative to the distance and decreases when the distance is smaller, so that we can accurately determine its magnitude. The absolute values of the error are also small: always below 0.01 for the top-k based method. This gives evidence that we can make good use of the sketches for this task: the error is small enough to accurately use to measure the D_{TV}. From these results, we may lean towards preferring the top-k estimator, since it reliably achieves lower error here and in other tests, at least when k is large enough (here, we use $k = 10,000$).

Figure 5c varies the number of samples, N, and shows that our estimators work for both small and large datasets. Larger values of N are associated with better accuracy, but not dramatically so. The accuracy of the top-k estimator is slightly better than the heavy hitters estimator and has appreciably less variation. The absolute sketch error is often small – always < 0.02, and often < 0.01. This is accurate enough to detect a change in a distribution, sufficient to trigger retraining a model or other use cases discussed in Section 1. The sparse sketch results improve as N increases, but remain much worse than the fast sketches on this data.



(a) Varying k for both estimators **(b) Varying DP parameter privacy ε** **(c) Varying the size of the input N**

Figure 6: Accuracy experiments on Sent140 data with differential privacy

Table 2: Accuracy and timing results on real data sets ($\varepsilon = 3$)

Dataset	D_{TV}	Top-k	HH	Time(s)	Sparse	Time(s)
IMDB-(i)	**0.137**	0.005	0.008	2.335	0.274	5.378
IMDB-(ii)	**0.038**	0.001	0.006	2.354	0.426	5.345
Sent140-(i)	**0.121**	0.002	0.005	3.224	0.410	7.821
Sent140-(ii)	**0.195**	0.003	0.007	4.301	0.318	10.787
Celeb-A-(i)	**0.465**	0.011	0.016	0.467	0.065	0.503
Celeb-A-(ii)	**0.525**	0.011	0.013	0.463	0.065	0.453
Celeb-A-(iii)	**0.413**	0.007	0.013	0.451	0.073	0.492
FEMNIST-(i)	**0.509**	0.013	0.006	0.059	0.207	0.076
FEMNIST-(ii)	**0.272**	0.006	0.006	0.056	0.063	0.081
FEMNIST-(iii)	**0.216**	0.005	0.004	0.066	0.109	0.069

5.4 Real data with differential privacy

Next, we study differential privacy noise (discrete Laplace noise corresponding to $\varepsilon = 3$) on the real datasets. Adding privacy noise does not substantially weaken the accuracy in Figure 6. For the sketch size k, we see a general trend that increasing k improves accuracy (Figure 6a). Similar to the synthetic data case, the top-k estimator requires a sketch that is large enough in order to obtain accurate results. We see that for our preferred sketch size of $k = 10,000$, it obtains very high accuracy, even with added privacy noise. Moreover, the variation in error is lower for the top-k estimator, compared to the HH-based estimator. The absolute error bound is below 0.01 for large k and sufficiently accurate to identify any notable change in the input distribution.

In Figure 6b, we vary the privacy parameter ε, also known as the privacy budget, from 1 to 5. Unsurprisingly, the accuracy improves as ε increases, corresponding to decreased levels of noise. The accuracy of the HH method tends to flatten near 0.01 error on this data even as the noise decreases to close to zero, because there is still the inherent noise of the sketching. Meanwhile, the top-k error decreases to negligible levels, as its sketching error keeps decreasing. This is consistent with results on data without privacy noise, where top-k is the preferred approach for accuracy.

Figure 6c shows the accuracy when varying the size of the input, from a small fraction (100K words) to almost the whole Sent140 dataset (10M words). For both methods, error decreases with N. The reason is that as N increases, the absolute magnitude of privacy noise remains the same, so its relative impact on the L_1 estimation decreases as N increases. For this data, it means that for small inputs (100K words), the privacy noise can drown out the signal, since individual words have small absolute frequency. However, for large enough inputs, the accuracy is sufficient to detect data shift. To highlight this, we additionally plot the true total variation distance on this figure as "exact". For $N = 10^6$, the estimate's error is almost equal to the target distance. Meanwhile, for $N = 10^7$, the error is about an order of magnitude smaller than the distance.

Finally, Table 2 shows results with DP for all the different ways to split the four different real data sets described above. Different splits lead to different D_{TV} values: splitting in half leads to lower differences, while splitting based on labels means the distributions are less alike. In all cases, the top-k estimator achieves the lowest error in the fastest time. The HH error is larger but usable, while the sparse sketch error is too large to be of value here.

6 CONCLUDING REMARKS

This paper introduced the problem of federated data shift distance estimation, and presented fast sketches with proven accuracy. The main experimental takeaway is that the sketch-based approach to federated data shift distance estimation is feasible. With a moderately small sketch (160KB), we can accurately estimate D_{TV} between distributed distributions. The same sketch admits two different estimators. The top-k based estimator is preferred when k is large enough. Our approach is the first to also offer privacy guarantees, as the results are quite robust to the addition of differentially private noise on the item frequencies. The techniques are of most value when the inputs are high dimensional: for lower-dimensional data, it may be preferable to materialize the distributions directly.

Future work is to broaden the applicability of sketching techniques within the federated model, to allow more properties of distributed data to be estimated, in order to support federated computations, such as data preparation and post-processing for modern data management tasks. It will also be of interest to evaluate them in distributed systems, and measure the effect of sketching for downstream tasks. Scenarios with very weak clients (where it is not feasible to store a small sketch) may need a new approach.

ACKNOWLEDGMENTS

This work is supported in part by EPSRC grant EP/V044621/1, the UKRI Prosperity Partnership Scheme (FAIR) under the EPSRC Grant EP/V056883/1, and The Alan Turing Institute.

REFERENCES

[1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep Learning with Differential Privacy. In *ACM SIGSAC Conference on Computer and Communications Security*. ACM, 308–318. https://doi.org/10.1145/2976749.2978318

[2] Amirali Abdullah, Ravi Kumar, Andrew McGregor, Sergei Vassilvitskii, and Suresh Venkatasubramanian. 2016. Sketching, Embedding and Dimensionality Reduction in Information Theoretic Spaces. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS (JMLR Workshop and Conference Proceedings)*, Vol. 51. JMLR.org, 948–956. http://proceedings.mlr.press/v51/abdullah16.html

[3] John M. Abowd. 2018. The U.S. Census Bureau Adopts Differential Privacy. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, (KDD)*. ACM, 2867. https://doi.org/10.1145/3219819.3226070

[4] Naman Agarwal, Peter Kairouz, and Ziyu Liu. 2021. The Skellam Mechanism for Differentially Private Federated Learning. In *Advances in Neural Information Processing Systems*. 5052–5064. https://proceedings.neurips.cc/paper/2021/hash/285baacbdf8fda1de94b19282acd23e2-Abstract.html

[5] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. 2011. Streaming Algorithms via Precision Sampling. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*. 363–372. https://doi.org/10.1109/FOCS.2011.82

[6] Eugene Bagdasaryan, Peter Kairouz, Stefan Mellem, Adrià Gascón, Kallista A. Bonawitz, Deborah Estrin, and Marco Gruteser. 2022. Towards Sparse Federated Analytics: Location Heatmaps under Distributed Differential Privacy with Secure Aggregation. *Proc. Priv. Enhancing Technol.* 2022, 4 (2022), 162–182. https://doi.org/10.56553/POPETS-2022-0104

[7] Victor Balcer and Salil P. Vadhan. 2019. Differential Privacy on Finite Computers. *J. Priv. Confidentiality* 9, 2 (2019). https://doi.org/10.29012/jpc.679

[8] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. 2020. Secure Single-Server Aggregation with (Poly)Logarithmic Overhead. In *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna (Eds.). ACM, 1253–1269. https://doi.org/10.1145/3372297.3417885

[9] Akash Bharadwaj and Graham Cormode. 2022. An Introduction to Federated Computation. In *International Conference on Management of Data (SIGMOD)*. ACM, 2448–2451. https://doi.org/10.1145/3514221.3522561

[10] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. 2012. The Johnson-Lindenstrauss Transform Itself Preserves Differential Privacy. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS*. IEEE Computer Society, 410–419. https://doi.org/10.1109/FOCS.2012.67

[11] Burton H. Bloom. 1970. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commun. ACM* 13, 7 (1970), 422–426.

[12] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM, 1175–1191. https://doi.org/10.1145/3133956.3133982

[13] Vladimir Braverman, Robert Krauthgamer, and Lin F. Yang. 2020. Universal Streaming of Subset Norms. *CoRR* abs/1812.00241 (2020). arXiv:1812.00241 http://arxiv.org/abs/1812.00241

[14] Bo Brinkman and Moses Charikar. 2005. On the impossibility of dimension reduction in l_1. *J. ACM* 52, 5 (2005), 766–788. https://doi.org/10.1145/1089023.1089026

[15] Sebastian Caldas, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. 2018. LEAF: A Benchmark for Federated Settings. *CoRR* abs/1812.01097 (2018). arXiv:1812.01097 http://arxiv.org/abs/1812.01097

[16] Karan N. Chadha, Junye Chen, John C. Duchi, Vitaly Feldman, Hanieh Hashemi, Omid Javidbakht, Audra McMillan, and Kunal Talwar. 2023. Differentially Private Heavy Hitter Detection using Federated Analytics. *CoRR* abs/2307.11749 (2023). https://doi.org/10.48550/arXiv.2307.11749

[17] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. 2004. Finding frequent items in data streams. *Theor. Comput. Sci.* 312, 1 (2004), 3–15. https://doi.org/10.1016/S0304-3975(03)00400-6

[18] Wei-Ning Chen, Ayfer Özgür, and Peter Kairouz. 2022. The Poisson Binomial Mechanism for Unbiased Federated Learning with Secure Aggregation. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (Proceedings of Machine Learning Research)*, Vol. 162. PMLR, 3490–3506. https://proceedings.mlr.press/v162/chen22s.html

[19] Graham Cormode and Hossein Jowhari. 2019. L_p Samplers and Their Applications: A Survey. *ACM Comput. Surv.* 52, 1 (2019), 16:1–16:31. https://doi.org/10.1145/3297715

[20] Graham Cormode, Samuel Maddock, and Carsten Maple. 2021. Frequency Estimation under Local Differential Privacy. *Proc. VLDB Endow.* 14, 11 (2021), 2046–2058. https://doi.org/10.14778/3476249.3476261

[21] Graham Cormode and S. Muthukrishnan. 2005. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms* 55, 1 (2005), 58–75. https://doi.org/10.1016/j.jalgor.2003.12.001

[22] Graham Cormode and Ke Yi. 2020. *Small Summaries for Big Data*. Cambridge University Press.

[23] Differential Privacy Team at Apple. 2017. Learning with Privacy at Scale. https://machinelearning.apple.com/research/learning-with-privacy-at-scale.

[24] Cynthia Dwork and Aaron Roth. 2014. The Algorithmic Foundations of Differential Privacy. *Found. Trends Theor. Comput. Sci.* 9, 3-4 (2014), 211–407. https://doi.org/10.1561/0400000042

[25] Hubert Eichner, Daniel Ramage, Kallista Bonawitz, Dzmitry Huba, Tiziano Santoro, Brett McLarnon, Timon Van Overveldt, Nova Fallen, Peter Kairouz, Albert Cheu, Katharine Daly, Adria Gascon, Marco Gruteser, and Brendan McMahan. 2024. Confidential Federated Computations. arXiv:2404.10764 [cs.CR]

[26] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. 2007. Hyperloglog: The analysis of a near-optimal cardinality estimation algorithm. In *International Conference on Analysis of Algorithms*.

[27] Arik Friedman and Assaf Schuster. 2010. Data mining with differential privacy. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 493–502. https://doi.org/10.1145/1835804.1835868

[28] Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. Project Report, Stanford.

[29] Sudipto Guha, Piotr Indyk, and Andrew McGregor. 2008. Sketching information divergences. *Mach. Learn.* 72, 1-2 (2008), 5–19. https://doi.org/10.1007/S10994-008-5054-X

[30] Farzin Haddadpour, Belhal Karimi, Ping Li, and Xiaoyun Li. 2020. FedSKETCH: Communication-Efficient and Private Federated Learning via Sketching. *CoRR* abs/2008.04975 (2020). arXiv:2008.04975 https://arxiv.org/abs/2008.04975

[31] Jonathan Hehir, Daniel Ting, and Graham Cormode. 2023. Sketch-Flip-Merge: Mergeable Sketches for Private Distinct Counting. In *ICML*.

[32] Charlie Hou, Hongyuan Zhan, Akshat Shrivastava, Sid Wang, Aleksandr Livshits, Giulia Fanti, and Daniel Lazar. 2023. Privately Customizing Prefinetuning to Better Match User Data in Federated Learning. arXiv:2302.09042 [cs.LG]

[33] Dzmitry Huba, John Nguyen, Kshitiz Malik, Ruiyu Zhu, Mike Rabbat, Ashkan Yousefpour, Carole-Jean Wu, Hongyuan Zhan, Pavel Ustinov, Harish Srinivas, Kaikai Wang, Anthony Shoumikhin, Jesik Min, and Mani Malek. 2022. PAPAYA: Practical, Private, and Scalable Federated Learning. In *Proceedings of Machine Learning and Systems 2022, MLSys 2022, Santa Clara, CA, USA, August 29 - September 1, 2022*. mlsys.org. https://proceedings.mlsys.org/paper/2022/hash/f340f1b1f65b6df5b5e3f94d95b11daf-Abstract.html

[34] Piotr Indyk. 2000. Stable Distributions, Pseudorandom Generators, Embeddings and Data Stream Computation. In *Foundations of Computer Science (FOCS)*. IEEE Computer Society, 189–197. https://doi.org/10.1109/SFCS.2000.892082

[35] Piotr Indyk and Rajeev Motwani. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*. ACM, 604–613. https://doi.org/10.1145/276698.276876

[36] Piotr Indyk, Rajeev Motwani, Prabhakar Raghavan, and Santosh S. Vempala. 1997. Locality-Preserving Hashing in Multidimensional Spaces. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*. ACM, 618–625. https://doi.org/10.1145/258533.258656

[37] Sergey Ioffe. 2010. Improved Consistent Sampling, Weighted Minhash and L1 Sketching. In *IEEE International Conference on Data Mining*. IEEE Computer Society, 246–255. https://doi.org/10.1109/ICDM.2010.80

[38] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2021. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.* 14, 1-2 (2021), 1–210. https://doi.org/10.1561/2200000083

[39] Saqib A. Kakvi, Keith M. Martin, Colin Putman, and Elizabeth A. Quaglia. 2023. SoK: Anonymous Credentials. In *Security Standardisation Research (Lecture Notes in Computer Science)*, Vol. 13895. Springer, 129–151. https://doi.org/10.1007/978-3-031-30731-7_6

[40] Ping Li. 2007. Very sparse stable random projections for dimension reduction in lalpha (0 <alpha<=2) norm. In *Proceedings of the 13th ACM SIGKDD International*

Conference on Knowledge Discovery and Data Mining. ACM, 440–449. https://doi.org/10.1145/1281192.1281241

[41] Ping Li. 2008. Estimators and tail bounds for dimension reduction in l_α ($0 < \alpha \leq 2$) using stable random projections. In *ACM-SIAM Symposium on Discrete Algorithms, (SODA)*. SIAM, 10–19. http://dl.acm.org/citation.cfm?id=1347082.1347084

[42] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Large-scale CelebFaces Attributes (CelebA) Dataset. https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html.

[43] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 142–150.

[44] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *International Conference on Artificial Intelligence and Statistics (AISTATS) (Proceedings of Machine Learning Research)*, Vol. 54. PMLR, 1273–1282. http://proceedings.mlr.press/v54/mcmahan17a.html

[45] Jelani Nelson and David P. Woodruff. 2010. Fast Manhattan sketches in data streams. In *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. ACM, 99–110. https://doi.org/10.1145/1807085.1807101

[46] Rasmus Pagh and Nina Mesing Stausholm. 2021. Efficient Differentially Private F_0 Linear Sketching. In *International Conference on Database Theory, (ICDT) (LIPIcs)*, Vol. 186. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 18:1–18:19. https://doi.org/10.4230/LIPIcs.ICDT.2021.18

[47] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. 2020. FetchSGD: Communication-Efficient Federated Learning with Sketching. In *International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research)*, Vol. 119. PMLR, 8253–8265. http://proceedings.mlr.press/v119/rothchild20a.html

[48] Adam D. Smith, Shuang Song, and Abhradeep Thakurta. 2020. The Flajolet-Martin Sketch Itself Preserves Differential Privacy: Private Counting with Minimal Space. In *Advances in Neural Information Processing Systems*. https://proceedings.neurips.cc/paper/2020/hash/e3019767b1b23f82883c9850356b71d6-Abstract.html

[49] Kunal Talwar, Shan Wang, Audra McMillan, Vojta Jina, Vitaly Feldman, Bailey Basile, Áine Cahill, Yi Sheng Chan, Mike Chatzidakis, Junye Chen, Oliver Chick, Mona Chitnis, Suman Ganta, Yusuf Goren, Filip Granqvist, Kristine Guo, Frederic Jacobs, Omid Javidbakht, Albert Liu, Richard Low, Dan Mascenik, Steve Myers, David Park, Wonhee Park, Gianni Parsa, Tommy Pauly, Christian Priebe, Rehan Rishi, Guy Rothblum, Michael Scaria, Linmao Song, Congzheng Song, Karl Tarbe, Sebastian Vogt, Luke Winstrom, and Shundong Zhou. 2023. Samplable Anonymous Aggregation for Private Federated Data Analysis. *CoRR* abs/2307.15017 (2023). https://doi.org/10.48550/arXiv.2307.15017 arXiv:2307.15017

[50] Mikkel Thorup. 2013. Bottom-k and priority sampling, set similarity and subset sums with minimal independence. In *Symposium on Theory of Computing Conference*, Dan Boneh, Tim Roughgarden, and Joan Feigenbaum (Eds.). ACM, 371–380. https://doi.org/10.1145/2488608.2488655

[51] Lun Wang and Dawn Song. 2021. Differentially Private Frequency Moments Estimation with Polylogarithmic Space. In *ICLR*.

[52] Tianhao Wang, Bolin Ding, Jingren Zhou, Cheng Hong, Zhicong Huang, Ninghui Li, and Somesh Jha. 2019. Answering Multi-Dimensional Analytical Queries under Local Differential Privacy. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, Peter A. Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska (Eds.). ACM, 159–176. https://doi.org/10.1145/3299869.3319891

[53] Fuheng Zhao, Dan Qiao, Rachel Redberg, Divyakant Agrawal, Amr El Abbadi, and Yu-Xiang Wang. 2022. Differentially Private Linear Sketches: Efficient Implementations and Applications. *CoRR* abs/2205.09873 (2022). https://doi.org/10.48550/arXiv.2205.09873 arXiv:2205.09873

[54] Wennan Zhu, Peter Kairouz, Brendan McMahan, Haicheng Sun, and Wei Li. 2020. Federated Heavy Hitters Discovery with Differential Privacy. In *International Conference on Artificial Intelligence and Statistics, (AISTATS) (Proceedings of Machine Learning Research)*, Vol. 108. PMLR, 3837–3847. http://proceedings.mlr.press/v108/zhu20a.html