FLAIM: AIM-based Synthetic Data Generation in the Federated Setting

Samuel Maddock
University of Warwick
Coventry, UK
s.maddock@warwick.ac.uk

Graham Cormode
Meta AI & University of Warwick
Coventry, UK
gcormode@meta.com

Carsten Maple
University of Warwick
Coventry, UK
cm@warwick.ac.uk

ABSTRACT

Preserving individual privacy while enabling collaborative data sharing is crucial for organizations. Synthetic data generation is one solution, producing artificial data that mirrors the statistical properties of private data. While numerous techniques have been devised under differential privacy, they predominantly assume data is centralized. However, data is often distributed across multiple clients in a federated manner. In this work, we initiate the study of federated synthetic tabular data generation. Building upon a SOTA central method known as AIM, we present *DistAIM* and *FLAIM*. We first show that it is straightforward to distribute AIM, extending a recent approach based on secure multi-party computation which necessitates additional overhead, making it less suited to federated scenarios. We then demonstrate that naively federating AIM can lead to substantial degradation in utility under the presence of heterogeneity. To mitigate both issues, we propose an augmented FLAIM approach that maintains a private proxy of heterogeneity. We simulate our methods across a range of benchmark datasets under different degrees of heterogeneity and show we can improve utility while reducing overhead.

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**.

KEYWORDS

Synthetic Data, Federated Learning, Differential Privacy

1 INTRODUCTION

Modern computational applications are predicated on the availability of significant volumes of high-quality data. Increasingly, such data is not freely available: it may not be collected in the volume needed, and may be subject to privacy concerns. Recent regulations such as the General Data Protection Regulation (GDPR) restrict the extent to which data collected for a specific purpose may be

processed for some other goal. The aim of *synthetic data generation* (SDG) is to solve this problem by allowing the creation of realistic artificial data that shares the same structure and statistical properties as the original data source. SDG is an active area of research, offering the potential for organisations to share useful datasets while protecting the privacy of individuals [2, 36, 50].

SDG methods fall into two categories: deep learning [15, 25, 53] and statistical models [55, 56]. Nevertheless, without strict privacy measures in place, it is possible for SDG models to leak information about the data it was trained on [19, 38, 46]. It is common for deep learning approaches such as Generative Adversarial Networks (GANs) to produce verbatim copies of training data, breaching privacy [45, 49]. A standard approach to prevent leakage is to use Differential Privacy (DP) [12]. DP is a formal definition which ensures the output of an algorithm does not depend heavily on any one individual's data by introducing calibrated random noise. Under DP, statistical models have become state-of-the-art (SOTA) for tabular data and often outperform deep learning counterparts [14, 29, 47]. Approaches are based on Bayesian networks [56], Markov random fields [33] and iterative marginal-based methods [3, 28, 32].

Private SDG methods perform well in centralized settings where a trusted curator holds all the data. However, in many settings, data cannot be easily centralized. Instead, there are multiple participants each holding a small private dataset who wish to generate synthetic data. Federated learning (FL) is a paradigm that applies when multiple parties wish to collaboratively train a model without sharing data directly [23]. In FL, local data remains on-device, and only model updates are transmitted back to a central aggregator [35]. FL methods commonly adopt differential privacy to provide formal privacy guarantees and is widely used in deep learning [20, 22, 34]. However, there has been minimal focus on federated SDG: we only identify a recent effort of Pereira et al. to distribute Multiplicative Weights with Exponential Mechanism (MWEM) via secure multi-party computation (SMC) [42]. Their work focuses on a distributed setting which assumes a small number of participants are *all* available to secret-share data before the protocol begins. This is not suited for the fully federated setting where there may be thousands of clients and only a small proportion available at a particular round.

In this work, we study generating differentially private tabular data in the federated setting where only a small proportion of clients are available per-round who exhibit strong data heterogeneity. We propose FLAIM, a novel federated analogue to the current SOTA central DP algorithm AIM [32]. We show how an analog to traditional FL training can be formed with clients performing a number of local steps before sending model updates to the server in the form of noisy marginals. We highlight how this naive extension can suffer severely under strong heterogeneity which is exacerbated
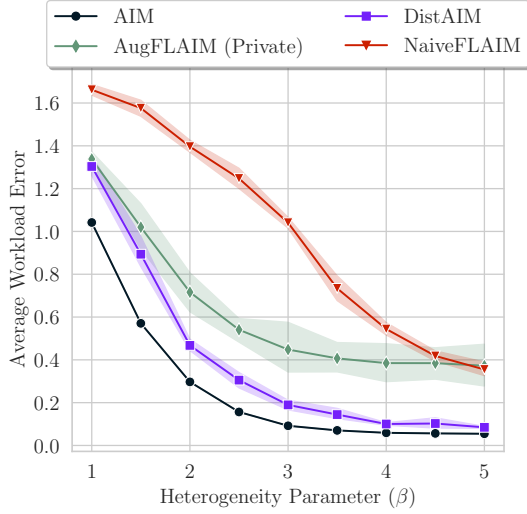
Figure 1: Average error over a workload of marginals for (FL)AIM trained with $\varepsilon = 1$ on a toy federated dataset. β varies client feature skew where large β results in less skew.

when only a few clients participate per round. To circumvent this, we modify FLAIM by replacing components of central AIM with newly-built steps that are better suited to the federated setting, such as augmenting clients' local choices via a private proxy of skew to ensure decisions are not adversely affected by heterogeneity.

EXAMPLE. *Figure 1 presents a federated scenario where 10% of 100 clients participate per round. Each client holds data with varying degrees of feature skew, where a larger β implies less heterogeneity. We use four variations: Centralised AIM (black); Distributed AIM, our adaptation of Pereira et al. [42] (purple); our naive federated AIM approach (red); and our improved federated version (green). We plot the L_1 error over a workload of marginal queries trained with $\varepsilon = 1$. Due to client availability, there is an inevitable utility gap between central and distributed AIM. By naively federating AIM, client decisions made in local training are strongly affected by heterogeneity while distributed AIM is not, resulting in a big loss in utility. This gap is almost closed in high skew scenarios (small β) by penalising clients' local decisions via a private measure of heterogeneity (AugFLAIM).*

Our main contributions are as follows:

- We are the first to study marginal-based methods in the federated setting. We extend the work of Pereira et al. [42] who focus on a strongly synchronized distributed setting with MWEM to instead form a distributed protocol that replaces MWEM with AIM to obtain greater utility (DistAIM).
- Motivated to reduce the overheads present in DistAIM, we propose FLAIM, our federated analogue of AIM [32] that is designed specifically for the federated setting. We propose novel extensions based on augmenting utility scores in AIM decisions via a private proxy that reduces the effect heterogeneity has on local decisions, resulting in increased model performance and smaller overheads.

- We show empirically our FLAIM method outperforms federated deep learning approaches such as DP-CTGAN, which extends conclusions of prior studies to the federated setting.
- We perform an extensive empirical study on 7 realistic tabular datasets. We show FLAIM obtains utility matching DistAIM but reduces the need for heavyweight SMC, resulting in less overhead. Furthermore, we show our FLAIM approaches are resistant to varying levels of heterogeneity[1].

2 PRELIMINARIES

We assume the existence of K participants each holding local datasets $D_1, \ldots, D_K$ over a set of d attributes such that the full dataset is denoted $D := \cup_k D_k$. Additionally, we assume that each attribute is categorical[2]. For a record $\boldsymbol{x} := (x_1, \ldots, x_d) \in D$ we denote x_i as the value of attribute i. For each attribute $i \in [d]$, we define A_i as the set of discrete values that x_i can take. For a subset of attributes $q \subseteq [d]$ we abuse notation and let x_q be the subset of $\boldsymbol{x}$ with attributes in the set q. We are mostly concerned with computing marginal queries over D (or individual D_k). Let $q \subseteq [d]$ and define $A_q := \prod_{i \in q} A_i$, as the set of values q can take and $n_q := |A_q|$ as the cardinality of q.

DEFINITION 2.1 (MARGINAL QUERY). *A marginal query for a subset of features $q \subseteq [d]$ is a function $M_q : \mathcal{D} \to \mathbb{R}^{n_q}$ where each entry is a count of the form $(M_q(D))_j := \sum_{x \in D} \mathbf{1}[x_q = a_j]$, $\forall j \in [n_q], a_j \in A_q$.*

As an example, consider a dataset with two features: unemployment and age where $A_1 = \{0, 1\}$ and $A_2 = \{1, 2, \ldots, 99\}$. The output of the marginal query $q = \{unemployment, age\}$ is a vector where an entry is a count of each record that satisfies a possible combination of feature values e.g., $\{unemp = 0, age = 18\}$. The goal in workload-based synthetic data generation is to generate a synthetic dataset $\hat{D}$ that minimises $\text{Err}(D, \hat{D})$ over a given workload of (marginal) queries Q. We follow existing work and study the average workload error under the L_1 norm [32].

DEFINITION 2.2 (AVERAGE WORKLOAD ERROR). *Denote the workload $Q = \{q_1, \ldots, q_m\}$ as a set of marginal queries where each $q \subseteq [d]$. The average workload error for synthetic dataset $\hat{D}$ is defined $\text{Err}(D, \hat{D}; Q) := \frac{1}{|Q|} \sum_{q \in Q} \|M_q(D) - M_q(\hat{D})\|_1$*

We are interested in producing a synthetic dataset $\hat{D}$ with marginals close to that of D. However, in the federated setting it is often impossible to form the global dataset $D := \cup_k D_k$ due to privacy restrictions or client availability. Instead the goal is to gather sufficient information from local datasets D_k and train a model that learns $M_q(D)$. For any D_k, the marginal query $M_q(D_k)$ and local workload error $\text{Err}(D_k, \hat{D})$ are defined analogously.

Differential Privacy (DP) [11] is a formal notion that guarantees the output of an algorithm does not depend heavily on any individual. We seek to guarantee (ε, δ)-DP, where the parameter ε is called the privacy budget and determines an upper bound on the privacy leakage of the algorithm. The parameter δ defines the probability of failing to meet this, and is set very small. DP has many attractive properties including sequential composition, meaning that if two algorithms are $(\varepsilon_1, \delta_1)$-DP and $(\varepsilon_2, \delta_2)$-DP respectively, then

---

[1]Our code is available at https://github.com/Samuel-Maddock/flaim
[2]We discretize continuous features via uniform binning, see Appendix B.1 for details.

their joint output on a specific dataset satisfies $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-DP. Tighter bounds are obtained via zero-Concentrated DP (zCDP) [7]:

Definition 2.3 (ρ-zCDP). *A mechanism $\mathcal{M}$ is ρ-zCDP if for any two neighbouring datasets D, D' and all $\alpha \in (1, \infty)$ we have $D_\alpha(\mathcal{M}(D)|\mathcal{M}(D')) \leq \rho \cdot \alpha$, where D_α is Renyi divergence of order α.*

One can convert ρ-zCDP to obtain an (ε, δ)-DP guarantee. The notion of "adjacent" datasets can lead to different privacy definitions. We assume example-level privacy, which defines two datasets D, D' to be adjacent if D' can be formed from the addition/removal of a single row from D. To satisfy DP it is common to require bounded sensitivity of the function we wish to privatize.

Definition 2.4 (Sensitivity). *Let $f : \mathcal{D} \to \mathbb{R}^d$ be a function over a dataset. The L_2 sensitivity of f, denoted $\Delta_2(f)$, is defined as $\Delta_2(f) := \max_{D \sim D'} \|f(D) - f(D')\|_2$, where $D \sim D'$ represents the example-level relation between datasets. Similarly, $\Delta_1(f)$ is defined with the L_1 norm as $\Delta_1(f) := \max_{D \sim D'} \|f(D) - f(D')\|_1$.*

We use two foundational DP methods that are core to many DP-SDG algorithms, the Gaussian and Exponential mechanisms [12].

Definition 2.5 (Gaussian Mechanism). *Let $f : \mathcal{D} \to \mathbb{R}^d$, the Gaussian mechanism is defined as $GM(f) = f(D) + \Delta_2(f) \cdot \mathcal{N}(0, \sigma^2 I_d)$. The Gaussian mechanism satisfies $\frac{1}{2\sigma^2}$-zCDP.*

Definition 2.6 (Exponential Mechanism). *Let $u(q; \cdot) : \mathcal{D} \to \mathbb{R}$ be a utility function defined for all $q \in Q$. The exponential mechanism releases q with probability $\mathbb{P}[\mathcal{M}(D) = q] \propto \exp(\frac{\varepsilon}{2\Delta} \cdot u(q; D))$, with $\Delta := \max_q \Delta_1(u(q; D))$. This satisfies $\frac{\varepsilon^2}{8}$-zCDP.*

Iterative Methods (Select-Measure-Generate). Recent methods for private tabular data generation follow the "Select-Measure-Generate" paradigm which is also the core focus of our work. These are broadly known as iterative methods [28] and usually involve training a graphical model via noisy marginals over a number of steps. In this work, we focus on AIM [32], an extension of the classical MWEM algorithm [16], which replaces the multiplicative weight update with a graphical model inference procedure called Private-PGM [33]. PGM learns a Markov Random Field (MRF) and applies post-processing optimisation to ensure consistency in the generated data. PGM can answer queries without directly generating data from the model, thus avoiding additional sampling errors.

In outline, given a workload of queries Q, AIM proceeds as follows (further details are in the full technical report):

(1) At each round t, via the exponential mechanism, **select** a query $q \in Q$ that is worst-approximated by the current synthetic dataset.
(2) Under the Gaussian mechanism **measure** the chosen marginal and update the graphical model via PGM.
(3) At any point, we can **generate** synthetic data via PGM that best explains the observed measurements.

AIM begins round t by computing utility scores for each query $q \in Q$ of the form,

$$u(q; D) = w_q \cdot (\|M_q(D) - M_q(\hat{D}^{(t-1)})\|_1 - \sqrt{\tfrac{2}{\pi}} \cdot \sigma_t \cdot n_q),$$

where $\hat{D}^{(t-1)}$ is the current PGM model. The core idea is to select marginals that are high in error (first term) balanced with the expected error from measuring the query under Gaussian noise with variance σ_t^2 (second term). The utility scores are weighted by $w_q := \sum_{r \in Q} |r \cap q|$, which calculates the overlap of other marginals in the workload with q. The sensitivity of the resulting exponential mechanism is $\Delta = \max_q w_q$ since measuring $\|M_q(D) - M_q(\hat{D}^{(t-1)})\|_1$ has sensitivity 1 which is weighted by w_q. Once a query is selected it is measured by the Gaussian mechanism with variance σ_t^2 and sensitivity 1. An update to the model via PGM is then applied using all observed measurements so far. See Appendix A.1 for full details.

Towards Decentralized Synthetic Data. Given a set of K clients with datasets $D_1, \ldots D_K$ and workload Q, the goal is to learn a synthetic dataset $\hat{D}$ that best approximates $D := \cup_k D_k$ over Q e.g., $M_q(\hat{D}) \approx M_q(D), \forall q \in Q$. However, computing statistics directly from D is not possible as each D_k is private. We make an important distinction here between the highly-synchronized distributed and loosely-coordinated federated settings. In the distributed setting, all participants are available to collaboratively share $M_q(D_k)$ and some central server(s) compute steps of AIM in a strongly synchronized manner, with high communication overhead. This is the original setting of Pereira et al. [42]. Instead we are mainly interested in the federated setting where we assume that participants are more weakly engaged, and may become unavailable or dropout at any moment. We model this by assuming that each participant participates in the current round only with probability p. We also assume each D_k exhibits heterogeneity which could manifest as significant feature-skew or a varying number of samples. We detail how we model heterogeneity in Section 5.

3 DISTRIBUTED AIM

Our first proposal, DistAIM, translates the AIM algorithm directly into the federated setting by having computing servers jointly calculate each step, attempting to mirror what would be computed in the central setting. To do so, computing servers must collaborate privately and securely, such that no one participant's raw query answers, $M_q(D_k)$, are revealed. The "select" and "measure" steps require direct access to private local datasets D_k, and hence we need to implement distributed DP mechanisms for these steps. We present an overview here with full details in Appendix A.2.

Pereira et al. [42], describe one such approach for MWEM. They utilize various secure multi-party computation (SMC) primitives based on secret-sharing [1]. However, a key difference is they assume a distributed setting where all participants first secret-share their workload answers to computing servers before the protocol begins. These computing servers implement secure exponential and Laplace mechanisms over shares of marginals via standard SMC operations [24]. This is a key difference to our federated setting where we assume partial participation of clients over multiple rounds. Their approach also has two drawbacks: first, their cryptographic solution incurs both a computation and communication overhead which may be prohibitive in federated scenarios. Secondly, their approach is based on MWEM which results in a significant loss in utility. Furthermore, MWEM is memory-intensive and does not scale to high-dimensional datasets.

Instead, we apply the framework of Pereira et al. [42] to AIM, and adapt this for our federated setting. Compared to AIM and Pereira et al., our DistAIM approach has important differences:

Client participation: At each round only a subset of participants are available to join the AIM round. For simplicity, we assume clients are sampled with probability p. In expectation, pK clients contribute their secret-shared workload answers e.g., $\{[\![M_q(D_k)]\!] : q \in Q\}$ to computing servers. This is an immediate difference with the setting of Pereira et al. [42], where it is assumed all clients are available to secret-share marginals before training. Instead, in DistAIM the secret-shares from participants are aggregated across rounds and the "select" and "measure" steps are carried out via computing servers over the updated aggregate shares at a particular round. Compared to the central setting, DistAIM incurs additional error due to subsampling.

Select step: A key difficulty extending AIM (or MWEM) to a distributed setting is the use of the exponential mechanism. In order to apply this, the utility scores $u(q; D)$ must be calculated. Following Protocol 2 of Pereira et al. [42], sampling from the exponential mechanism can be done over secret-shares of the marginal $[\![M_q(D_k)]\!]$ since utility scores $u(q; D)$ depend only linearly in $M_q(D_k)$.

Measure step: Once a marginal has been sampled, it must be measured. Protocol 3 in Pereira et al. [42] proposes one way to securely generate Laplace noise between computing servers. This is then added to the aggregate sum of a secret-shared marginal. To remain consistent with AIM, we use Gaussian noise instead.

Estimate step: Under the post-processing properties of DP the computing server(s) are free to use the measured noisy marginals with PGM to update the graphical model, as in the centralized case.

4 FLAIM: FL ANALOG FOR AIM

While DistAIM is one solution, it is not defined within the standard federated paradigm where clients typically perform a number of local steps before sending model updates to a server. Furthermore, the SMC-based approach can have large overheads which is prohibitive for federated clients who have limited bandwidth (we explore this in Section 5.3). This leads us to design an AIM approach that is analogous to traditional Federated Learning (FL), where only lightweight SMC is needed in the form of secure-aggregation (SecAgg) [4]. In FL, the paradigm for training models is to do computation on-device, having clients perform multiple local steps before sending a model update. The server aggregates all client updates and performs an update to the global model [35]. When combined with DP, model updates are aggregated via SecAgg schemes and noise is added either by a trusted server or in a distributed manner. In the case of AIM, we denote our analogous FL approach as FLAIM. In FLAIM, the selection step of AIM is performed locally by clients (across multiple local training steps). Each clients' chosen marginals are then sent to a trusted server via SecAgg and noise is added.

In more detail, FLAIM is outlined in Algorithm 1. We present three variations, with differences highlighted in color. Shared between all variations are the key differences with DistAIM displayed in blue underline. First is NaiveFLAIM, a straightforward translation of AIM into the federated setting. In Section 4.1, we explain the shortcomings of such an approach which stems from scenarios where clients' local data exhibits strong heterogeneity. Motivated by this, Section 4.2 proposes AugFLAIM (Oracle) a variant of FLAIM that assumes oracle access to a measure of skew which can be used to augment local utility scores. This skew measure is non-private

---

Algorithm 1 FLAIM

Input: K participants with data $D_1, \ldots, D_k$, sampling rate p, global rounds T, local rounds s, workload Q, privacy parameters (ε, δ)

1: **for** each global round $t = 1 \ldots T$ **do**
2:     Form P_t by sampling each client k with probability p
3:     **for** each client $k \in P_t$ **do**
4:         **for** each local step $l = 1 \ldots s$ **do**
5:             Filter workload $Q \leftarrow Q \setminus \{|q| = 1 : q \in Q\}$ (Private)

6:             Compute a heterogeneity measure for each $q \in Q$

$$\tilde{\tau}_k(q) \leftarrow \begin{cases} 0 & \text{Naive} \\ \tau_k(q) := \|M_q(D_k) - M_q(D)\|_1 & \text{Oracle} \\ \frac{1}{|q|} \sum_{j \in q} \|M_{\{j\}}(D_k) - M_{\{j\}}(\hat{D}^{(t-1)})\|_1 & \text{Private} \end{cases}$$

7:             **Select** $q_{t+l} \in Q$ using Exp-Mech with utility score(s) $u(q; D_k) :=$
$$w_q \left(\|M_q(D_k) - M_q(\hat{D}^{(t-1)+l})\|_1 - \sqrt{\frac{2}{\pi}} \cdot \sigma_{(t-1)+l} \cdot n_q \underline{-\tilde{\tau}_k(q)} \right)$$

8:             **Measure** marginal $\tilde{M}_q(D_k) := M_q(D_k) + \mathcal{N}\left(0, \sigma_{(t-1)+l}^2 I\right)$

9:             **Estimate** new local model via PGM as
$$\hat{D}_k^{(t-1)+l} \leftarrow \arg\min_{p \in S} \sum_{i=1}^{(t-1)+l} \frac{1}{\sigma_i} \|M_{q_i}(p) - \tilde{M}_{q_i}(D_k)\|_2$$

10:         Share all 1-way marginals under SecAgg,
$$\mathcal{M}_k^1 \leftarrow \{(t, j, [\![M_{\{j\}}(D_k)]\!])\}_{j \in [d]} \text{ (Private only)}$$

11:         Compute $\mathcal{M}_k \leftarrow \{(t, q_{t+l}, [\![M_{q_{t+l}}(D_k)]\!])\}_{l \in [s]} \cup \mathcal{M}_k^1$

12:         Send $\mathcal{M}_k$ to the server via SecAgg

13:     Server updates measurement list $\mathcal{M}^t := \cup_{k \in P_t} \mathcal{M}_k$

14:     **for** each unique $q \in \mathcal{M}^t$ aggregate marginals and add noise
$$\tilde{M}_q^t := \sum_{\{k : M_q \in \mathcal{M}_k\}} [\![M_q(D_k)]\!] + N(0, \sigma_t^2)$$

15:     **for** each $\tilde{M}_q^t$ compute $\alpha_q^t := \begin{cases} 1/\sigma_t, & \text{Naive} \\ N_q^t/\sigma_t, & \text{Oracle} \\ \tilde{N}_q^t/\sigma_t, & \text{Private} \end{cases}$

16:     Server updates measurement list $\mathcal{M}$ with each $(t, q, \tilde{M}_q^t, \alpha_q^t)$ and updates the global model
$$\hat{D}_t \leftarrow \arg\min_{p \in S} \sum_{(t, q, \tilde{M}_q^t, \alpha_q^t) \in \mathcal{M}} \alpha_q^t \|M_q(p) - \tilde{M}_q^t\|_2$$

---

and not obtainable in practice, but provides an idealized baseline. Lastly, Section 4.3 introduces AugFLAIM (Private), which again augments local utility scores but with a private proxy of heterogeneity alongside other heuristics to improve utility.

All FLAIM variants proceed by sampling clients to participate in round t. Each client performs a number of local steps s, which consist of performing a local selection step using the exponential mechanism, measuring the chosen marginal under local noise and updating their local model via PGM. When each client finishes local training, they send back each chosen query q alongside the associated marginal $[\![M_q(D_k)]\!]$, which are aggregated via secure-aggregation and noise is added by the central server. Hence, local training is done under local differential privacy (LDP) to not leak any privacy between steps, whereas the resulting global update is under a form of distributed DP where noise is added by the central

server to the securely-aggregated marginals. We assume all AIM methods run for T global rounds. AIM can also set T adaptively via budget annealing and we explore this in our experiments (see Appendix B.4 for details).

4.1 NaiveFLAIM and Heterogeneous Data

NaiveFLAIM is our first attempt at a SDG in the federated setting, by directly translating the AIM algorithm. However, in federated settings, participants often exhibit strong heterogeneity in their local datasets. That is, clients' local datasets D_k can differ significantly from the global dataset D. Such heterogeneity will affect AIM in both the "select" and "measure" steps. If D_k and D are significantly different then the local marginal $M_q(D_k)$ will differ from the true marginal $M_q(D)$. We quantify heterogeneity for a client k and query $q \in Q$ via the L_1 distance:

$$\tau_k(q) := \|M_q(D_k) - M_q(D)\|_1.$$

This can be viewed as a measure of query skew. In FLAIM, we proceed by clients perform a number of local steps. The first stage involves carrying out a local "select" step based on utility scores of the form $u(q; D_k) \propto \|M_q(D_k) - M_q(\hat{D}^{(t-1)})\|_1$. Suppose for a particular client k there exists a query $q \in Q$ such that $M_q(D_k)$ exhibits strong heterogeneity. If at step t the current model $\hat{D}^{(t-1)}$ is a good approximation of D, then it is probable that client k ends up selecting any query that has high heterogeneity since $u(q; D_k) \propto \|M_q(D_k) - M_q(\hat{D}^{(t-1)})\|_1 \approx \|M_q(D_k) - M_q(D)\|_1 = \tau_k(q)$. This mismatch can harm model performance and is compounded by having many clients select (multiple and possibly differing) skewed marginals and so the model is updated in a way that drifts from D.

4.2 AugFLAIM (Oracle): Tackling Heterogeneity

The difficulty above arises as clients choose marginals via local applications of the exponential mechanism with a score that does not account for underlying skew. We have $u(q; D_k) \propto$

$$\|M_q(D_k) - M_q(\hat{D})\|_1 \leq \|M_q(D) - M_q(\hat{D})\| + \|M_q(D_k) - M_q(D)\|_1$$
$$\propto u(q; D) + \tau_k(q)$$

To circumvent this, we should correct local utility scores by down-weighting marginals based on $\tau_k(q)$ and modify utility scores as:

$$u(q; D_k) \propto \|M_q(D_k) - M_q(\hat{D})\| - \tau_k(q)$$

where $\tau_k(q)$ is an exact L_1 measure of heterogeneity for client k at a marginal q. Unfortunately, measuring $\tau_k(q)$ under privacy constraints is not feasible. That is, $\tau_k(q)$ depends directly on $M_q(D)$, which is exactly what we are trying to learn via AIM. Still, we introduce AugFLAIM (Oracle) as an idealized baseline to compare with. AugFLAIM (Oracle) is a variation of FLAIM that assumes oracle access to $\tau_k(q)$ and augments local utility scores as above.

4.3 AugFLAIM (Private): Heterogeneity Proxy

Since augmenting utility scores directly via $\tau_k(q)$ is not feasible, we seek a proxy $\tilde{\tau}_k(q)$ that is reasonably correlated with $\tau_k(q)$ and can be computed under privacy. This proxy measure can be used to correct local utility scores, penalising queries via $\tilde{\tau}_k(q)$. This helps ensure clients select queries that are not adversely affected

Table 1: Comparison of FLAIM approaches against baselines for negative log-likelihood (NLL), $\varepsilon = 5$. Smaller NLL is better.

Method / Dataset	Adult	Credit	Covtype
Fed DP-CTGAN	37.1	83.8	62.7
FedNaiveBayes	25.33	18.02	44.9
FLAIM (Random)	83.9	47.7	58.4
NaiveFLAIM	29.4	18	45.4
AugFLAIM (Private)	**20.87**	**16.2**	**41.6**
DP-CTGAN	28.6	27.6	45.9
AIM	**19.2**	**15.57**	**40.92**

by heterogeneity. We propose the following proxy

$$\tilde{\tau}_k(q) := \frac{1}{|q|} \sum_{j \in q} \|M_{\{j\}}(D_k) - \tilde{M}_{\{j\}}(D)\|_1$$

Instead of computing a measure for each $q \in Q$, we compute one for each feature $j \in [d]$, where $\tilde{M}_{\{j\}}(D_k)$ is a noisy estimate of the 1-way marginal for feature j. For a particular query $q \in Q$, we average the skew of the associated features contained in q. Such a $\tilde{\tau}_k(q)$ relies only on estimating the distribution of each feature. This estimate can be refined across multiple federated rounds as each participant can measure $M_{\{j\}}(D_k)$ for each $j \in [d]$ and have the server sum and add noise (via SecAgg) to produce a new private estimate $\tilde{M}_{\{j\}}(D)$ each round. We add two further enhancements:
1. Filtering and combining 1-way marginals (Line 10). As we require clients to estimate all features at every round, we remove 1-way marginals from the workload to prevent clients from measuring the same marginal twice. All 1-way marginals that are estimated for $\tilde{\tau}_k(q)$ are fed back into PGM to improve the global model.
2. Weighting Marginals (Line 15). In PGM, measurements are weighted by $\alpha = 1/\sigma_t$, so those that are measured with less noise have more importance in the optimisation of model parameters. Both AugFLAIM variations adopt an additional weighting scheme that includes the total sample size that contributed to a particular marginal q at round t, $N_q^t := \sum_{\{k:M_q^t \in \mathcal{M}_k\}} |D_k|$ where the weight becomes $\alpha_q^t = N_q^t/\sigma_t$. This relies on knowing the number of samples that are aggregated. In some cases, the size of datasets may be deemed private. In such scenarios, it can be estimated from the noisy marginal $\tilde{M}_q$ by summing the counts to produce $\tilde{N}_q$.

The privacy guarantees of all FLAIM variations follow directly from those of AIM. The use of a heterogeneity measure incurs an additional sensitivity cost for the exponential mechanism and AugFLAIM (Private) incurs an additional privacy cost as it measures each of the d features at every round. The following lemma captures this. See Appendix A.3 for the full proof.

LEMMA 4.1. *For any number of global rounds T and local rounds s, FLAIM satisfies (ε, δ)-DP, under Gaussian budget allocation $r \in (0, 1)$ by computing ρ according to Lemma A.2, and setting*

$$\sigma_t = \begin{cases} \sqrt{\frac{Ts+d}{2 \cdot r \cdot \rho}}, & \textit{Naive or Oracle} \\ \sqrt{\frac{T(s+d)}{2 \cdot r \cdot \rho}}, & \textit{Private} \end{cases}, \quad \varepsilon_t = \sqrt{\frac{8 \cdot (1-r) \cdot \rho}{Ts}}$$

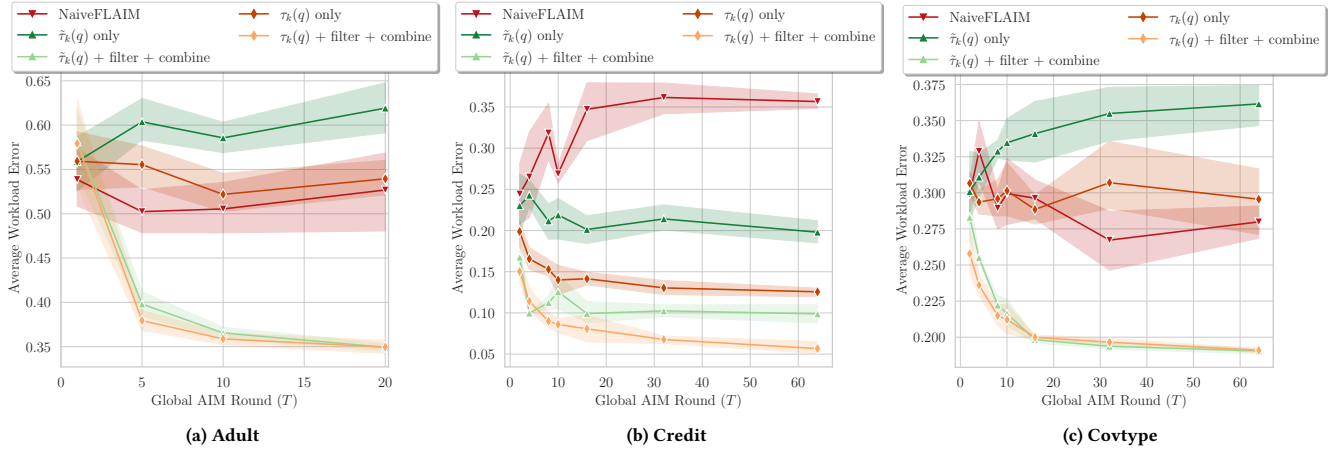*For AugFLAIM methods, the exponential mechanism is applied with sensitivity $\Delta := \max_q 2w_q$.*

Figure 2: Ablation study, comparing utility for FLAIM variations that augment local utility scores, $\varepsilon = 5, T = 10, s = 1, p = 0.1$

5 EXPERIMENTAL EVALUATION

For our experiments, we utilize realistic benchmark tabular datasets from the UCI repository [10]: Adult, Magic, Marketing and Covtype. We further use datasets common for benchmarking synthetic data: Census and Intrusion from the Synthetic Data Vault (SDV) [40]. We also construct a toy dataset with feature-skew denoted SynthFS. Full details on all datasets are contained in Appendix B.1.

We evaluate our methods in three ways: average workload error (as defined in Section 2), average negative log-likelihood (evaluated on test data) and the area under the curve (AUC) of a decision tree model trained on synthetic data and evaluated on test data.

For all datasets, we simulate heterogeneity by forming non-IID splits in one of two ways: The first is by performing dimensionality reduction and clustering nearby points to form client partitions that have strong feature-skew. We call this the "clustering" approach. For experiments that require varying heterogeneity, we form splits via an alternative label-skew method popularized by Li et al. [27]. This samples a label distribution $p_c \in [0, 1]^K$ for each class c from a Dirichlet(β) where larger β results in less heterogeneity. See Appendix B.2 for full details. In the following sections, all experiments have $K = 100$ clients with partitions formed from the clustering approach unless stated otherwise. We train (FL)AIM models on a fixed workload of 3-way marginal queries chosen at random with $|Q| = 64$ and average results over 10 independent runs. Further experiments on datasets besides Adult are contained in Appendix C.

5.1 Comparison with Existing Baselines

We begin with an experiment comparing AugFLAIM (Private) to other federated baselines. One such SOTA approach is CTGAN [53]. We utilise the DP-CTGAN implementation within OpenDP's smartnoise-sdk [39] to compare to AIM. For the federated setting we train CTGAN using DP-FedSGD via FLSim [44]. For details on hyperparameters see Appendix B.4. We further compare AugFLAIM (Private) against two AIM baselines. FLAIM (Random) which takes FLAIM and randomly chooses a query $q \in Q$ without utilising the

exponential mechanism. Instead, all privacy budget is spent on the "Measure" step. The other is FedNaiveBayes which restricts the workload Q to only 1-way marginals and is equivalent to training a NaiveBayes model. In Table 1 we present the negative log-likelihood (NLL) for models trained to an $\varepsilon = 5$ across three datasets. Methods achieving lowest NLL for a particular dataset are in bold.

For the central setting, AIM achieves better performance than DP-CTGAN across each dataset. This confirms prior studies such as [29] that show graphical model approaches achieve better utility than deep learning methods for tabular data. For the federated setting, we note FedNaiveBayes and FLAIM (Random) both perform poorly in comparison to AugFLAIM (Private). This illustrates two main points: utilising the exponential mechanism does result in a substantial increase in utility (i.e., randomly choosing $q \in Q$ is poor) and that utilising a workload of k-way marginals with $k > 1$ gives best utility (i.e., NaiveBayes is poor). Further note, AugFLAIM (Private) has better utility than NaiveFLAIM which shows augmenting utility scores in the Exponential mechanism does improve utility. We explore this further in Section 5.3. Finally, Fed DP-CTGAN performs very poorly compared to AugFLAIM. Even NaiveFLAIM and occasionally FedNaiveBayes outperform it. There are further issues for practitioners: first, Fed DP-CTGAN requires a large number of hyperparameters to be tuned for best utility such as client and server learning rates and the clipping norm. This is in contrast to (FL)AIM methods that only have a single hyperparameter - the total number of global rounds T. Secondly, CTGAN requires a large number of training epochs. In this experiment we train for 50 epochs which is equivalent to $T = 500$ rounds whereas the FLAIM methods achieve better utility in only $T = 10$ rounds. For these reasons, in further experiments, we do not compare to federated CTGAN.

5.2 Ablation Study: Utility of AugFLAIM

To understand what determines the utility of AugFLAIM (Private) we present an ablation study in Figure 2. Here we train FLAIM models with $\varepsilon = 5$ on Adult, Credit and Covtype whilst varying
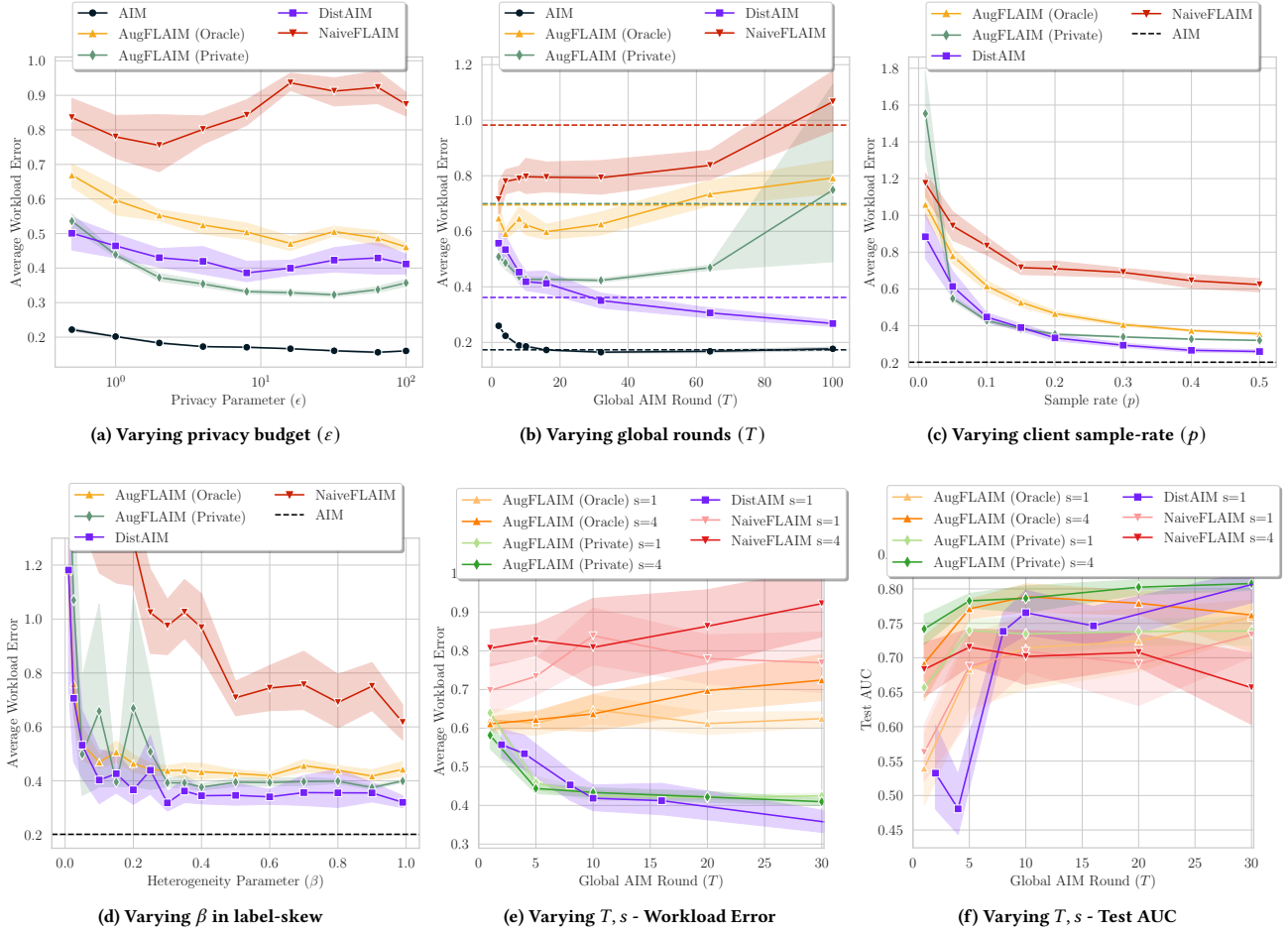
Figure 3: Varying (FL)AIM Parameters on Adult; Unless otherwise stated $T = 10$, $s = 1$, $p = 0.1$, $K = 100$, $\varepsilon = 1$

the global rounds T. We present NaiveFLAIM compared with variations that augment the utility scores of the Exponential mechanism. These are: using the true heterogeneity measure $\tau_k(q)$ only (otherwise denoted AugFLAIM (Oracle)); using $\tau_k(q)$ with the filter-and-combine heuristic; using the private heterogeneity proxy $\tilde{\tau}_k(q)$ only and using $\tilde{\tau}_k(q)$ with the filter and combine heuristic (otherwise denoted AugFLAIM (Private)). On the Credit dataset, using $\tau_k(q)$ or $\tilde{\tau}_k(q)$ only results in a clear improvement over NaiveFLAIM, and when combined with the heuristics the lowest error is obtained. On Adult and Covtype, using only $\tau_k(q)$ or the private proxy $\tilde{\tau}_k(q)$ does not immediately result in lower workload error than Naive-FLAIM. Instead, utilising the filter and combine heuristics results in the best workload error overall. In further experiments, we denote AugFLAIM (Private) as the method which augments utility scores with $\tilde{\tau}_k(q)$ and uses the filter and combine heuristic whereas we denote AugFLAIM (Oracle) as the method that has oracle access to $\tau_k(q)$ only (without further heuristics).

5.3 Parameter Settings

Having concluded that AugFLAIM (Private) achieves the best performance against other federated baselines, we now present a detailed set of experiments comparing FLAIM methods with DistAIM across a variety of federated settings. We compare AIM and DistAIM against NaiveFLAIM and our two variants that augment local utility scores: AugFLAIM (Oracle) using $\tau_k(q)$ only and AugFLAIM (Private) using proxy $\tilde{\tau}_k(q)$ with filtering and combining 1-ways.

Varying the privacy budget (ε). In Figure 3a, we plot the workload error whilst varying ε on Adult, sampling 10% of clients per round and setting $T = 10$. First, we observe a clear gap in performance between DistAIM and central AIM due to the error from subsampling a small number of clients per round. We observe that naively federating AIM gives the worst performance even as ε becomes large. Furthermore, augmenting utility scores makes a clear improvement in workload error, particularly for $\varepsilon > 1$. By estimating feature distributions at each round, AugFlaim (Private) can obtain performance that matches or sometimes improves upon DistAIM for larger values of ε. We further note that AugFLAIM (Private) has lower error than AugFLAIM (Oracle) which may seem

counter-intuitive. However, AugFLAIM (Oracle) is still trained under DP, only oracle access to $\tau_k(q)$ is assumed which is non-private and trained without heuristics as explored in Section 5.2.

Varying the number of global AIM rounds (T). In Figure 3b, we vary the number of global AIM rounds and fix $\varepsilon = 1$. Additionally, we plot the setting where T is chosen adaptively by budget annealing. This is shown in dashed lines for each method. First observe with DistAIM, the workload error decreases as T increases. Since computing servers aggregate secret-shares across rounds, then as T grows large, most clients will have been sampled and the server(s) have workload answers over most of the (central) dataset. For all FLAIM variations, the workload error usually increases when T is large, since they are more sensitive to the increased amount of noise that is added. For NaiveFLAIM, this is worsened by client heterogeneity. Further, we observe that for AugFLAIM (Private), the utility matches that of DistAIM unless the choice of T is very large. At $T = 100$, the variance in utility is high, sometimes even worse than that of NaiveFLAIM. This is since the privacy cost scales in both the number of rounds and features, resulting in too much noise. In the case of annealing, T is chosen adaptively by an early stopping condition (see Appendix B.4). While annealing has good performance in central AIM, it obtains poor utility across all federated methods. For annealing on Adult, AugFLAIM (Private) matches AugFLAIM (Oracle) and both perform better than NaiveFLAIM. Overall, we found choosing T to be small (≤ 30) gives best performance for AugFLAIM and should avoid using budget annealing.

Client-participation (p). In Figure 3c, we plot the average workload error whilst varying the per-round participation rate (p) with $T = 10, \varepsilon = 1$. We observe clearly the gap in performance between central AIM and DistAIM is caused by the error introduced by subsampling and when $p \geq 0.5$ performance is almost matched. For NaiveFLAIM, we observe the performance improvement as p increases is slower than other methods. When p is large, NaiveFLAIM receives many measurements, each likely to be highly heterogeneous and thus the model struggles to learn consistently. For both AugFLAIM variations, we observe the utility improves with client participation but does eventually plateau. AugFLAIM (Private) consistently matches the error of DistAIM except when p is large, but we note this is not a practical regime in FL.

Varying heterogeneity (β). In Figure 3d, we plot the average workload error on the Adult dataset over client splits formed by varying the heterogeneity parameter (β) to produce label-skew. Here, a larger β corresponds to a more uniform partition and therefore less heterogeneity. In the label-skew setting, data is both skewed according to the class attribute of Adult and the number of samples, with only a few clients holding the majority of the dataset. We observe that when the skew is large ($\beta < 0.1$), all methods struggle. As β increases and skew decreases, NaiveFLAIM performs the worst and AugFLAIM (Private) has stable error, close to that of DistAIM.

Varying local rounds (s). A benefit of the federated setting is that clients can perform a number of local steps before sending all measured marginals to the server. However, for FLAIM methods, this incurs an extra privacy cost in the number of local rounds (s). In Figure 3e, we vary $s \in \{1, 4\}$ and plot the workload error. Although there is an associated privacy cost with increasing s, the errors are not significantly different for small T. As we vary T, the associated privacy cost becomes larger and the workload error

increases for methods that perform $s = 4$ local updates. Although increasing the number of local rounds (s) does not result in lower workload error, and in cases where T is misspecified can give far worse performance, it is instructive to instead study the test AUC of a classification model trained on the synthetic data. In Figure 3f we see that performing more local updates can give better test AUC after fewer global rounds. For AugFLAIM (Private), this allows us to match the test AUC performance of DistAIM on Adult.

Comparison across datasets. Table 2 presents results across all datasets with client data partitioned via the clustering approach. We set $\varepsilon = 1, p = 0.1$ and $T = 10$. For each method we present both the average workload error and the negative log-likelihood over a holdout set. The first is a form of training error and the second a measure of generalisation. We observe that on 5 of the 7 datasets AugFLAIM (Private) achieves the lowest negative log-likelihood and workload error. On the other datasets, AugFLAIM (private) closely matches DistAIM in utility but with lower overheads.

Distributed vs. Federated AIM. Table 3 presents the overhead of DistAIM compared to AugFLAIM (Private) including average client throughput (sent and received communication) across protocols. We set $T \in [1, 200]$ that achieves lowest workload error. Observe on Adult, DistAIM requires twice as many rounds to achieve optimal error and results in a large (1300×) increase in client throughput compared to AugFLAIM. However, this results in 2× lower workload error and an 11% improvement in NLL. This highlights one of the chief advantages of FLAIM, where, for a small loss in utility, we can obtain much lower overheads. Furthermore, while a 2× gap in workload error seems significant, we refer back to Figure 3f, which shows the resulting classifier has AUC that is practical for downstream tasks. We note the overhead of DistAIM is significantly larger than FLAIM when queries in the workload have large cardinality (e.g., on Adult and Magic). Datasets with much smaller feature domains still have communication overhead but it is not as significant (e.g., Covtype which has many binary features).

6 RELATED WORK

Synthetic data has gained substantial traction due to its potential to mitigate privacy concerns and address limitations for sharing real-world data. Many generative deep learning approaches exist including GANs [15], VAEs [51] and diffusion models [18]. Recent work has extended these synthetic data generators (SDGs) to satisfy central differential privacy (DP) [30, 48, 52] and whilst results are promising for image data, performance on tabular data remains limited. Only a few generative tabular approaches exist including that of CTGAN [13, 53]. However, recent work has shown that private tabular approaches like DP-CTGAN often fail to provide good utility when compared to simpler models [14, 47]. Indeed, in the central setting of DP many successful methods are based on graphical models such as PrivBayes [56], PrivSyn [57], PGM [33] and AIM [32]. Recent work has shown the class of iterative methods [28] are SOTA on tabular data and we choose to focus on one of these methods, AIM, in our work. Meanwhile, research into SDGs in the federated setting remains limited. Recent federated SDGs are focused on image data such as MD-GAN [17], FedGAN [43] and FedVAE [54]. We do not compare with these in our work as they do not support tabular data or DP. The closest work to ours is that of

Table 2: Performance on datasets $K = 100$, $p = 0.1$, $\varepsilon = 1$, $T = 10$. Results show workload error and negative log-likelihood. Metrics are bold if a federated method achieves lowest on a specific dataset.

Method / Dataset	Adult	Magic	Census	Covtype	Credit	Intrusion	Marketing
NaiveFLAIM	0.8 / 29.28	1.64 / 2587.5	0.72 / 322.29	0.3 / 47.44	0.42 / 20.22	0.7 / 27.92	1.05 / 186.44
AugFLAIM (Oracle)	0.62 / 23.91	1.18 / 62.84	0.61 / 44.5	0.26 / 45.11	0.21 / 16.84	0.48 / 17.92	0.87 / 38.27
AugFLAIM (Private)	0.43 / 21.74	**1.07** / 28.9	**0.48** / **41.87**	**0.17** / 42.61	**0.14** / **16.33**	**0.32** / 15.05	**0.64** / **30.17**
DistAIM	**0.42** / **21.41**	1.08 / 35.04	0.54 / 55.19	0.2 / 45.94	0.18 / 16.9	**0.32** / **13.19**	0.66 / 40.57
AIM	0.2 / 19.3	0.85 / 23.7	0.28 / 34.26	0.06 / 41.06	0.07 / 15.62	0.12 / 9.96	0.24 / 20.87

Table 3: DistAIM vs. FLAIM at optimal T, metrics with ↑ show overhead of DistAIM over FLAIM and ↓ show % improvement of DistAIM over FLAIM. Client throughput is additionally stated in megabytes (MB) for DistAIM vs. FLAIM.

Dataset	$T(\uparrow)$	Throughput (↑)	Err (↓)	NLL (↓)
Adult	2×	1300× (80 / 0.06)	58%	11%
Magic	3.2×	1643× (80 / 0.04)	20%	14%
Census	1.5x	64x (29.6 / 0.46)	79%	33%
Intrusion	2.5x	366x (101 / 0.28)	82%	52%
Marketing	2.0x	97x (18 / 0.19)	77%	35%
Credit	1.0x	167x (93 / 0.55)	45%	6%
Covtype	1.25x	10x (7.6 / 0.76)	64%	3%

Pereira et al. [42] who propose a distributed DP version of MWEM [16] using secure multiparty computation (SMC) to distribute noise generation across computing servers. This approach has two main drawbacks: it assumes all clients are available to secret-share their workload answers and as it is based on MWEM, obtains subpar utility. Our work is motivated to extend their approach to AIM and to study an alternative and more natural federation of these methods. We also note the concurrent work of Pentyala et al. [41] which extends [42] to work with AIM via SMC.

7 CONCLUSION

Overall, we have shown that naively federating AIM under the challenges of FL causes a large decrease in utility when compared to the SMC-based DistAIM. To counteract this, we propose AugFLAIM (Private), which augments local decisions with a proxy for heterogeneity and obtains utility close to DistAIM while lowering overheads. In the future, we plan to extend our approaches to support user-level DP where clients hold multiple data items related to the same individual.

ACKNOWLEDGMENTS

REFERENCES

[1] Toshinori Araki, Jun Furukawa, Yehuda Lindell, Ariel Nof, and Kazuma Ohara. 2016. High-throughput semi-honest secure three-party computation with an honest majority. In *ACM SIGSAC CCS*. Vienna, 805–817.

[2] Samuel A Assefa, Danial Dervovic, Mahmoud Mahfouz, Robert E Tillman, Prashant Reddy, and Manuela Veloso. 2020. Generating synthetic data in finance: opportunities, challenges and pitfalls. In *Proceedings of the First ACM International Conference on AI in Finance*. ACM, New York, 1–8.

[3] Sergul Aydore, William Brown, Michael Kearns, Krishnaram Kenthapadi, Luca Melis, Aaron Roth, and Ankit A Siva. 2021. Differentially private query release through adaptive projection. In *ICML*. PMLR, 457–467.

[4] James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. 2020. Secure single-server aggregation with (poly) logarithmic overhead. In *ACM SIGSAC CCS*. Online, 1253–1269.

[5] Jock Blackard. 1998. Covertype. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C50K5N.

[6] R. Bock. 2007. MAGIC Gamma Telescope. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C52C8B.

[7] Mark Bun and Thomas Steinke. 2016. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*. Springer, Springer, Berlin, 635–658.

[8] Clément L Canonne, Gautam Kamath, and Thomas Steinke. 2020. The discrete gaussian for differential privacy. *Advances in Neural Information Processing Systems* 33 (2020), 15676–15688.

[9] DARPA. 1999. Darpa intrusion detection evaluation,. https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset

[10] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[11] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, Springer, 265–284.

[12] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[13] Mei Ling Fang, Devendra Singh Dhami, and Kristian Kersting. 2022. Dp-ctgan: Differentially private medical data generation using ctgans. In *International Conference on Artificial Intelligence in Medicine*. Springer, 178–188.

[14] Georgi Ganev, Kai Xu, and Emiliano De Cristofaro. 2023. Understanding how Differentially Private Generative Models Spend their Privacy Budget. arXiv:2305.10994 [cs.LG]

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* 63, 11 (2020), 139–144.

[16] Moritz Hardt, Katrina Ligett, and Frank McSherry. 2012. A simple and practical algorithm for differentially private data release. *Advances in neural information processing systems* 25 (2012).

[17] Corentin Hardy, Erwan Le Merrer, and Bruno Sericola. 2019. Md-gan: Multidiscriminator generative adversarial networks for distributed datasets. In *2019 IEEE international parallel and distributed processing symposium (IPDPS)*. IEEE, 866–877.

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.

[19] Florimond Houssiau, James Jordon, Samuel N Cohen, Owen Daniel, Andrew Elliott, James Geddes, Callum Mole, Camila Rangel-Smith, and Lukasz Szpruch. 2022. TAPAS: a toolbox for adversarial privacy auditing of synthetic data. *NeurIPS Workshop on Synthetic Data for Empowering ML Research* (2022).

[20] Dzmitry Huba, John Nguyen, Kshitiz Malik, Ruiyu Zhu, Mike Rabbat, Ashkan Yousefpour, Carole-Jean Wu, Hongyuan Zhan, Pavel Ustinov, Harish Srinivas, et al. 2022. Papaya: Practical, private, and scalable federated learning. *Proceedings of Machine Learning and Systems* 4 (2022), 814–832.

[21] Kaggle. 2017. Credit card fraud dataset. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
[22] Peter Kairouz, Brendan McMahan, Shuang Song, Om Thakkar, Abhradeep Thakurta, and Zheng Xu. 2021. Practical and private (deep) learning without sampling or shuffling. In *ICML*. PMLR, 5213–5225.
[23] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista 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, Zaid 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, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, 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. 2019. Advances and Open Problems in Federated Learning. arXiv:1912.04977 [cs.LG]
[24] Marcel Keller. 2020. MP-SPDZ: A versatile framework for multi-party computation. In *ACM SIGSAC CCS*. 1575–1590.
[25] Diederik P Kingma, Max Welling, et al. 2019. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning* 12, 4 (2019), 307–392.
[26] Ronny Kohavi and Barry Becker. 1996. Adult dataset. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml/nomao
[27] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. 2022. Federated learning on non-iid data silos: An experimental study. In *IEEE ICDE*. 965–978.
[28] Terrance Liu, Giuseppe Vietri, and Steven Z Wu. 2021. Iterative methods for private synthetic data: Unifying framework and new methods. *Advances in Neural Information Processing Systems* 34 (2021), 690–702.
[29] Yucong Liu, Chi-Hua Wang, and Guang Cheng. 2022. On the Utility Recovery Incapability of Neural Net-based Differential Private Tabular Training Data Synthesizer under Privacy Deregulation. arXiv:2211.15809 [cs.LG]
[30] Saiyue Lyu, Michael F Liu, Margarita Vinaroz, and Mijung Park. 2023. Differentially private latent diffusion models. *arXiv preprint arXiv:2305.15759* (2023).
[31] Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).
[32] Ryan McKenna, Brett Mullins, Daniel Sheldon, and Gerome Miklau. 2022. AIM: An Adaptive and Iterative Mechanism for Differentially Private Synthetic Data. *arXiv preprint arXiv:2201.12677* (2022).
[33] Ryan McKenna, Daniel Sheldon, and Gerome Miklau. 2019. Graphical-model based estimation and inference for differential privacy. In *ICML*. PMLR, 4435–4444.
[34] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, PMLR, 1273–1282.
[35] Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963* (2017).
[36] Ofer Mendelevitch and Michael D Lesh. 2021. Fidelity and privacy of synthetic medical data. *arXiv preprint arXiv:2101.08658* (2021).
[37] Sérgio Moro, Paulo Cortez, and Paulo Rita. 2014. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems* 62 (2014), 22–31.
[38] Sasi Kumar Murakonda, Reza Shokri, and George Theodorakopoulos. 2021. Quantifying the privacy risks of learning high-dimensional graphical models. In *AISTATS*. PMLR, PMLR, 2287–2295.
[39] OpenDP. 2021. smartnoise-sdk. https://github.com/opendp/smartnoise-sdk
[40] Neha Patki, Roy Wedge, and Kalyan Veeramachaneni. 2016. The Synthetic data vault. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, Canada, 399–410. https://doi.org/10.1109/DSAA.2016.49
[41] Sikha Pentyala, Mayana Pereira, and Martine De Cock. 2024. CaPS: Collaborative and Private Synthetic Data Generation from Distributed Sources. *arXiv preprint arXiv:2402.08614* (2024).
[42] Mayana Pereira, Sikha Pentyala, Anderson C. A. Nascimento, Rafael T. de Sousa Jr., and Martine De Cock. 2022. Secure Multiparty Computation for Synthetic Data Generation from Distributed Data. *CoRR* abs/2211.07332 (2022). https://doi.org/10.48550/ARXIV.2211.07332 arXiv:2211.07332
[43] Mohammad Rasouli, Tao Sun, and Ram Rajagopal. 2020. Fedgan: Federated generative adversarial networks for distributed data. *arXiv preprint arXiv:2006.07228* (2020).
[44] Facebook Research. 2021. FLSim. https://github.com/facebookresearch/FLSim
[45] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. 2017. Veegan: Reducing mode collapse in GANs using implicit variational learning. *Advances in neural information processing systems* 30 (2017).
[46] Theresa Stadler, Bristena Oprisanu, and Carmela Troncoso. 2022. Synthetic data–anonymisation groundhog day. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX, Vancouver, 1451–1468.
[47] Yuchao Tao, Ryan McKenna, Michael Hay, Ashwin Machanavajjhala, and Gerome Miklau. 2022. Benchmarking differentially private synthetic data generation algorithms. In *Workshop on Privacy-Preserving Artificial Intelligence, AAAI 2022*. AAAI, Vancouver.
[48] Reihaneh Torkzadehmahani, Peter Kairouz, and Benedict Paten. 2019. Dp-cgan: Differentially private synthetic data and label generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 0–0.
[49] Boris van Breugel, Hao Sun, Zhaozhi Qian, and Mihaela van der Schaar. 2023. Membership Inference Attacks against Synthetic Data through Overfitting Detection. *arXiv preprint arXiv:2302.12580* (2023).
[50] Boris van Breugel and Mihaela van der Schaar. 2023. Beyond Privacy: Navigating the Opportunities and Challenges of Synthetic Data. *arXiv preprint arXiv:2304.03722* (2023).
[51] Zhiqiang Wan, Yazhou Zhang, and Haibo He. 2017. Variational autoencoder based synthetic data generation for imbalanced learning. In *2017 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 1–7.
[52] Benjamin Weggenmann, Valentin Rublack, Michael Andrejczuk, Justus Mattern, and Florian Kerschbaum. 2022. DP-VAE: Human-readable text anonymization for online reviews with differentially private variational autoencoders. In *Proceedings of the ACM Web Conference 2022*. 721–731.
[53] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling tabular data using conditional gan. *Advances in neural information processing systems* 32 (2019).
[54] Haomiao Yang, Mengyu Ge, Kunlan Xiang, Xuejun Bai, and Hongwei Li. 2023. FedVAE: Communication-Efficient Federated Learning With Non-IID Private Data. *IEEE Systems Journal* (2023).
[55] Jim Young, Patrick Graham, and Richard Penny. 2009. Using Bayesian networks to create synthetic data. *Journal of Official Statistics* 25, 4 (2009), 549.
[56] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. 2017. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)* 42, 4 (2017), 1–41.
[57] Zhikun Zhang, Tianhao Wang, Jean Honorio, Ninghui Li, Michael Backes, Shibo He, Jiming Chen, and Yang Zhang. 2021. Privsyn: Differentially private data synthesis. (2021).

Algorithm 2 AIM [32]

Input: Dataset $D \in \mathbb{R}^{N \times d}$, workload Q, privacy parameters ε, δ, Maximum model size S

Output: Synthetic dataset $\hat{D}$

1: Initialise the zCDP budget $\rho_{\text{total}} \leftarrow \rho(\varepsilon, \delta)$ via Lemma A.2
2: Set $\sigma_0^2 \leftarrow \frac{16d}{0.9 \cdot \rho_{\text{total}}}, \varepsilon_0 \leftarrow \sqrt{8 \cdot 0.1 \cdot \rho_{\text{total}}/16d}$
3: Set $t \leftarrow 0$ and $Q \leftarrow \text{Completion}(Q)$
4: For each marginal $q \in Q$ initialise weights $w_q := \sum_{r \in Q} |q \cap r|$
5: **while** $\rho_{\text{used}} < \rho_{\text{total}}$ **do**
6:    $t \leftarrow t + 1$
7:    **if** $t = 0$ **then**   ▷ Initialise with one-way marginals
8:       Filter current rounds workload as
$$Q_0 \leftarrow \{q \in Q : |q| = 1\}$$
9:       Measure $\tilde{M}_q(D) \leftarrow M_q(D) + N(0, \sigma_0^2 I), \forall q \in Q_0$ and use PGM to estimate $\hat{D}_0$
10:       $\rho_{\text{used}} \leftarrow \rho_{\text{used}} + \frac{d}{2\sigma_t^2}$
11:    **else**
12:       Filter the workload
$$Q_t \leftarrow \{q \in Q : \text{ModelSize}(\hat{D}_{t-1}, q) \leq \frac{\rho_{\text{used}}}{\rho_{\text{total}}} \cdot S\}$$
13:       **Select** $q_t \in Q_t$ using the exponential mechanism with parameter ε_t and utility function
$$u(q; D) \leftarrow w_q \cdot \left(\|M_q(D) - M_q(\hat{D}_{t-1})\|_1 - \sqrt{\frac{2}{\pi}} \cdot \sigma_t \cdot n_q \right)$$
14:       **Measure** the chosen marginal q_t with the Gaussian mechanism i.e.,
$$\tilde{M}_{q_t}(D) \leftarrow M_{q_t}(D) + \mathcal{N}(0, \sigma_t^2 I)$$
15:       **Estimate** the new model via PGM [33]
$$\hat{D}_t \leftarrow \arg\min_{p \in \mathcal{S}} \sum_{i=1}^{t} \frac{1}{\sigma_i} \|M_{q_i}(p) - \tilde{M}_{q_i}(D)\|_2$$
16:       $\rho_{\text{used}} \leftarrow \rho_{\text{used}} + \frac{\varepsilon_t^2}{8} + \frac{1}{2\sigma_t^2}$
17:       **if** $\|M_{q_t}(\hat{D}_t) - M_{q_t}(\hat{D}_{t-1})\|_1 \leq \sqrt{2/\pi} \cdot \sigma_t \cdot n_{q_t}$ **then**   ▷ Budget Annealing
18:          Set $\sigma_{t+1} \leftarrow \sigma_t/2, \varepsilon_{t+1} \leftarrow 2 \cdot \varepsilon_t$
19:       **if** $(\rho_{\text{total}} - \rho_{\text{used}}) \leq 2(1/2\sigma_{t+1}^2 + \frac{1}{8}\varepsilon_{t+1}^2)$ **then** ▷ Final round
20:          Set $\sigma_{t+1}^2 \leftarrow 1/(2 \cdot 0.9 \cdot (\rho_{\text{total}} - \rho_{\text{used}})), \varepsilon_{t+1} \leftarrow \sqrt{8 \cdot 0.1 \cdot (\rho_{\text{total}} - \rho_{\text{used}})}$
21: **return** $\hat{D}_t$

A ALGORITHM DETAILS

A.1 AIM

The current SOTA method, and the core of our federated algorithms is AIM, introduced by McKenna et al. [32]. AIM extends the main ideas of MWEM [16] but augments the algorithm with an improved utility score function, a graphical model-based inference approach (via Private-PGM) and more efficient privacy accounting with zero-Concentrated Differential Privacy (zCDP). The full details of AIM are outlined in Algorithm 2. We refer to this algorithm as

'Central AIM', to distinguish it from the distributed and federated versions we consider in the main body of the paper. It is important to highlight the following details:

- **zCDP Budget Initialisation:** In central AIM, the number of global rounds T is set adaptively via budget annealing. To begin, $T := 16 \cdot d$ where d is the number of features. This is the maximum number of rounds that will occur in the case where the annealing condition is never triggered. This initialisation occurs in Line 2.

- **Workload Filtering:** The provided workload of queries, Q, is extended by forming the completion of Q. That is to say, all lower order marginals contained within any $q \in Q$ are also added to the workload. Furthermore, for the first round the workload is filtered to contain only 1-way marginals to initialise the model. This occurs in Line 8. In subsequent rounds, the workload is filtered to remove any queries that would force the model to grow beyond a predetermined maximum size S. This occurs at Line 12.

- **Weighted Workload:** Each marginal $q \in Q$ is assigned a weight via $w_q = \sum_{r \in Q} |q \cap r|$. Thus, marginals that have high overlap with other queries in the workload are more likely to be chosen. This is computed in Line 4.

- **Model Initialisation**: Instead of initialising the synthetic distribution uniformly over the dataset domain, the synthetic model is initialised by measuring each 1-way marginal in the workload W and using PGM to estimate the initial model. This corresponds to measuring each feature's distribution once before AIM begins and occurs in Lines 7-10.

- **Query Selection:** A marginal query is selected via the exponential mechanism with utility scores that compare the trade-off between the current error and the expected error when measured under Gaussian noise. The utility scores and selection step occur at Line 13.

- **Query Measurement:** Once a query has been chosen, it is measured under the Gaussian mechanism. This occurs at Line 14.

- **PGM model estimation:** The current PGM model is updated by adding the newly measured query to the set of previous measurements. The PGM model parameters are then updated by a form of mirror descent for a number of iterations. The precise details of PGM can be found in [33]. This occurs at Line 15.

- **Budget Annealing:** At the end of every round, the difference between the measured query of the new model and that of the previous model is taken. If this change is smaller than the expected error under Gaussian noise, the noise parameters are annealed by halving the amount of noise. This occurs at Line 17. If after this annealing there is only a small amount of remaining privacy budget left, the noise parameters can instead be calibrated to perform one final round before finishing. This occurs at Line 20.

A.2 DistAIM

We describe in full detail the DistAIM algorithm introduced in Section 3 and outlined in Algorithm 3. The algorithm can be seen as an adaptation of [42] who propose a secure multi-party computation

---

Algorithm 3 DistAIM

Input: Participants $P_1, \ldots P_k$ with local datasets $D_1, \ldots, D_k$, privacy parameters (ε, δ)

1: Initialise AIM parameters as in Lines 1-4 of Algorithm 2
2: **for** each round t **do**
3:     Sample participants $P_t \subseteq [k]$ with probability p and remove those who have already participated
4:     For each $k \in P_t$ who have not participated before, secret-share the workload answers $\{[\![M_q(D_k)]\!] : q \in W\}$ to the compute servers [1]
5:     **Aggregate:** The compute servers aggregate shares of the received answers and combine with previously received shares
$[\![M_q(\tilde{D}_t)]\!] \coloneqq \sum_{i=1}^{t-1} \sum_{k \in P_i} [\![M_q(D_k)]\!] + \sum_{k \in P_t} [\![M_q(D_k)]\!]$
6:     **Select:** Compute servers select $q_t \in Q$ using the exponential mechanism over secret shares $[\![M_q(\tilde{D}_t)]\!]$ via Protocol 2 in [42] with AIM utility scores

$$u(q; D_t) \coloneqq w_q \cdot (\|M_q(\tilde{D}_t) - M_q(\hat{D}_{t-1})\|_1 - \sqrt{\frac{2}{\pi}} \cdot \sigma_t \cdot n_q)$$

7:     **Measure:** q_t is measured using $[\![M_{q_t}(\tilde{D}_t)]\!]$ under a variation of Protocol 3 in [42], replacing Laplace noise with Gaussian to produce $\tilde{M}_{q_t}(\tilde{D}_t)$
8:     **Estimate** the new model via PGM using the received noisy measurements e.g.

$$\hat{D}_t \leftarrow \underset{p \in \mathcal{S}}{\arg\min} \sum_{i=1}^{t} \frac{1}{\sigma_i} \|M_{q_i}(p) - \tilde{M}_{q_i}(\tilde{D}_i)\|_2$$

(SMC) approach for distributing MWEM. The key differences are that we replace MWEM with AIM and consider a distributed setting where not all participants are available at any particular round. The approach relies on participants secret-sharing their query answers to compute servers who then perform a number of SMC operations over these shares to train the model. The resulting algorithm is identical to AIM in outline but has a few subtle differences:

- **Secret Sharing:** Participants must secret-share the required quantities to train AIM. In [42], it is assumed that the full workload answers $\{[\![M_q(D)]\!] : q \in Q\}$ have already been secret-shared between a number of compute servers. In DistAIM, we assume that clients sampled to participate at a particular round contribute their secret-shared workload answers $\{[\![M_q(D_k)]\!] : q \in Q\}$ which are aggregated with the shares of current and past participants from previous rounds. Thus, as the number of global rounds T increases, the secret-shared answers approach that of the central dataset. We assume the same SMC framework as [42] which is a 3-party scheme based on [1].
- **Client participation:** At each round only a subset of the participants are available to join the AIM round. In expectation pK clients will contribute their local marginals $[\![M_q(D_k)]\!]$ in the form of secret-shares. Compared to the central setting, DistAIM incurs additional error due to this subsampling.
- **Select step:** One key obstacle in extending AIM to a distributed setting is the exponential mechanism. Since each client holds a local dataset D_k, they cannot share their data

with the central server. Instead the quality functions $u(q; D)$ must be computed in a distributed manner between the compute servers who hold shares of the workload answers.
- **Measure step:** Once the marginal q_t has been selected by a secure exponential mechanism, it must be measured. As [42] utilise MWEM, they measure queries under Laplace noise which can be easily generated in an SMC setting. AIM instead uses Gaussian noise and this is also what we use in DistAIM. In practice, one can also implement this under SMC e.g., using the Box-Muller method.

A.3 FLAIM: Privacy Guarantees

In this section, we present and prove the privacy guarantees of the FLAIM approach. For completeness, we provide additional definitions and results, starting with the definition of (ε, δ)-Differential Privacy.

DEFINITION A.1 (DIFFERENTIAL PRIVACY [12]). *A randomised algorithm $\mathcal{M} \colon \mathcal{D} \to \mathcal{R}$ satisfies (ε, δ)-differential privacy if for any two adjacent datasets $D, D' \in \mathcal{D}$ and any subset of outputs $S \subseteq \mathcal{R}$,*

$$\mathbb{P}(\mathcal{M}(D) \in S) \le e^{\varepsilon} \mathbb{P}(\mathcal{M}(D') \in S) + \delta.$$

While we work using the more convenient formulation of ρ-zCDP (Definition 2.3), it is common to translate this guarantee to the more interpretable (ε, δ)-DP setting via the following lemma.

LEMMA A.2 (zCDP TO DP [8]). *If a mechanism $\mathcal{M}$ satisfies ρ-zCDP then it satisfies (ε, δ)-DP for all $\varepsilon > 0$ with*

$$\delta = \min_{\alpha > 1} \frac{\exp((\alpha - 1)(\alpha\rho - \varepsilon))}{\alpha - 1} \left(1 - \frac{1}{\alpha}\right)^{\alpha}$$

We restate the privacy guarantees of FLAIM and its variations.

LEMMA A.3 (LEMMA 4.1 RESTATED). *For any number of global rounds T and local rounds s, FLAIM satisfies (ε, δ)-DP, under Gaussian budget allocation $r \in (0, 1)$ by computing ρ according to Lemma A.2, and setting*

$$\sigma_t = \begin{cases} \sqrt{\frac{Ts+d}{2 \cdot r \cdot \rho}}, & \text{\textcolor{red}{Naive} or \textcolor{orange}{Oracle}} \\ \sqrt{\frac{T(s+d)}{2 \cdot r \cdot \rho}}, & \text{\textcolor{green}{Private}} \end{cases}, \quad \varepsilon_t = \sqrt{\frac{8 \cdot (1 - r) \cdot \rho}{Ts}}$$

For AugFLAIM methods, the exponential mechanism is applied with sensitivity $\Delta \coloneqq \max_q 2w_q$.

PROOF. For NaiveAIM, the result follows almost directly from AIM, since T rounds in the latter correspond to $T \cdot s$ in the former. We then apply the existing privacy bounds for AIM. Similarly, for AugFLAIM (Private), the 1-way marginals of every feature are included in the computation, thus increasing the number of measured marginals under Gaussian noise to $T \cdot (s + d)$. In all variations, the exponential mechanism is only applied once for each local round and thus Ts times in total. For AugFLAIM, the augmented utility scores $u(q; D_k)$ lead to a doubling of the sensitivity compared to AIM, since $M_q(D_k)$ is used twice in the utility score and thus $\Delta \coloneqq 2 \cdot \max_q w_q$. □

Table 4: Datasets - Those marked * have been subsampled for computational reasons.

Dataset	# of training samples	# of features	# of classes
Adult [26]	43,598	14	2
Credit [21]	284,807	30	2
Covtype* [5]	116,203	55	7
Census* [40]	89,786	41	2
Intrusion*[9]	197,608	40	5
Marketing [37]	41,188	21	2
Magic [6]	17,118	11	2
SynthFS (see Appendix B.1.1)	45,000	10	N/A

B EXPERIMENTAL SETUP

B.1 Datasets

In our experiments we use a range of tabular datasets from the UCI repository [10] and others available directly from the Synthetic Data Vault (SDV) package [40]. Additionally, we use one synthetic dataset that we construct ourselves. A summary of all datasets in terms of the number of training samples, features and classes is detailed in Table 4. All datasets are split into a train and test set with 90% forming the train set. From this, we form clients' local datasets via a partitioning method (see Appendix B.2). In more detail:

- **Adult** — A census dataset that contains information about adults and their occupations. The goal of the dataset is to predict the binary feature of whether their income is greater than $50,000. The training set we use contains 43,598 training samples and 14 features.
- **Credit** — A credit card fraud detection dataset available from Kaggle. The goal is to predict whether a transaction is fraudulent.
- **Covtype** — A forest cover type prediction dataset available from the UCI repository. We subsampled the dataset for computational reasons. Our train and test sets were formed from 20% of the original dataset.
- **Census** — US census dataset available through the synthetic data vault (SDV) package. This dataset was subsampled for computational reasons. Our training and test sets were formed from 30% of the original dataset.
- **Intrusion** — The DARPA network intrusion detection dataset containing network logs, available through the synthetic data vault (SDV) package. This was subsampled for computational reasons. Our training and test sets were formed from 40% of the original dataset.
- **Marketing** — A bank marketing dataset available from the UCI repository. The goal is to predict whether a client will subscribe to a term deposit.
- **Magic** — A dataset on imaging measurements from a telescope. The classification task is to predict whether or not the measurements are signal or background noise. The training set we use contains 17,118 samples and 11 features.
- **SynthFS** — A synthetic dataset formed from sampling features from a Gaussian distribution with different means. The precise construction is detailed in Appendix B.1.1. In our

experiments, the training set contains 45,000 samples with 10 features.

All continuous features are binned uniformly between the minimum and maximum which we assume to be public knowledge. We discretize our features with 32 bins, although experiments varying this size presented no significant change in utility. This follows the pre-processing steps taken by prior work [3, 32].

B.1.1 SynthFS. In order to simulate feature-skew in an ideal setting for FLAIM, we construct a synthetic dataset that we denote SynthFS. To create SynthFS, we draw independent features from a Gaussian distribution where the mean is chosen randomly from a Zipfian distribution whose parameter β controls the skew. This is done in the following manner:

- For each client $k \in [K]$ and feature $m \in [d]$ sample mean $\mu_m^k \sim \text{Zipf}(\beta, n_{\text{zipf}})$
- For each feature $m \in [d]$, sample n/K examples for client k from $N(\mu_m^k, 1)$

In our experiments we set $n = 50,000$ such that for $K = 100$ each client is assigned 500 samples. In order to form a test set we sample 10% from the dataset and assign the rest to clients. We fix $d = 10$ and $n_{\text{zipf}} = 40$ in all constructions. We highlight this process for $\beta \in \{1, 2, 3, 5\}$ in Figure 4, with $d = 2$ features for visualization purposes only. By increasing β, we decrease the skew of the means being sampled from the Zipf distribution. Hence, for larger β values, each client's features are likely to be drawn from the same Gaussian and there is no heterogeneity. Decreasing β increases the skew of client means and each feature is likely to be drawn from very different Gaussian distributions, as shown when $\beta = 1$.

B.2 Heterogeneity: Non-IID Client Partitions

In order to simulate heterogeneity on our benchmark datasets, we take one of the tabular datasets outlined in Appendix B.1 and form partitions for each client. The aim is to create client datasets that exhibit strong data heterogeneity by varying the number of samples and inducing feature-skew. We do this in two ways:

- **Clustering Approach** — In the majority of our experiments, we form client partitions via dimensionality reduction using UMAP [31]. An example of this process is shown in Figure 5 for the Adult dataset. Figure 5a shows a UMAP embedding of the training dataset in two-dimensions where each client partition (cluster) is highlighted a different color. To form these clusters we simply use K-means where $K = 100$ is

(a) $\beta = 5$ **(b) $\beta = 3$** **(c) $\beta = 2$** **(d) $\beta = 1$**
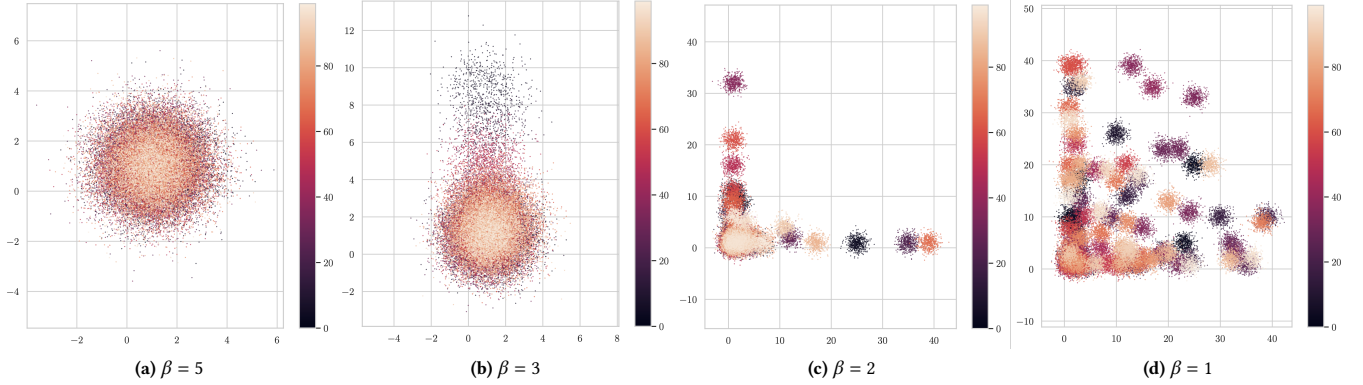
Figure 4: SynthFS: Synthetic dataset constructed with feature skew, varying $\beta \in \{1, 2, 3, 5\}$

the total number of clients we require. In Figures 5b-5d, we display the same embedding but colored based on different feature values for age, hours worked per-week and income > 50k. We observe, for instance, the examples that are largest in age are concentrated around $x = 10$ while those who work more hours are concentrated around $y = -7$. Thus clients that have datasets formed from clusters in the area of $(10, -7)$ will have significant feature-skew with a bias towards older adults who work more hours. These features have been picked at random and other features in the dataset have similar skew properties. The embedding is used only to map the original data to clients, and the raw data is used when training AIM models.

- **Label-skew Approach** — While the clustering approach works well to form non-IID client partitions, there is no simple parameter to vary the heterogeneity of the partitions. In experiments where we wish to vary heterogeneity, we follow the approach outlined by [27]. For each value the class variable can take, we sample the distribution $p_C \sim$ Dirichlet$(\beta) \subseteq [0, 1]^K$ and assign examples with class value C to the clients using this distribution. This produces client partitions that are skewed via the class variable of the dataset, where a larger β decreases the skew and reduces heterogeneity.

Table 5 presents the average heterogeneity for a fixed workload of queries across the Adult and Magic dataset with different partition methods for $K = 100$ clients. We look at the following methods: IID sampling, clustering approach, label-skew with $\beta = 0.1$ (large-skew) and label-skew with $\beta = 0.8$ (small-skew). Observe in all cases that our non-IID methods have higher heterogeneity than IID sampling. Specifically, the clustering approach works well to induce heterogeneity and can result in twice as much skew across the workload. Note also that increasing β from 0.1 to 0.8 decreases average heterogeneity and at $\beta = 0.8$, the skew is close to IID sampling. This confirms that simulating client partitions in this way is useful for experiments where we wish to vary heterogeneity, since we can vary β accordingly and $\beta \in (0, 1]$ in experiments is well-chosen.

Table 5: Average heterogeneity over a workload of uniform queries computed as $\frac{1}{K} \sum_k \sum_{q \in Q} \tau_k(q)$ whilst varying different client partition methods with $K = 100$ total clients.

Dataset / Partition	IID	Clustering	Label-skew $(\beta = 0.1)$	Label-skew $(\beta = 0.8)$
Adult	0.241	0.525	0.531	0.332
Magic	0.538	0.792	0.767	0.603

B.3 Evaluation

In our experiments we evaluate our methods with three different metrics:

1. Average Workload Error. We mainly evaluate (FL)AIM methods via the average workload error. For a fixed workload of marginal queries Q, we measure $\mathrm{Err}(D, \hat{D}; Q) := \frac{1}{|Q|} \sum_{q \in Q} \|M_q(D) - M_q(\hat{D})\|_1$ where $D := \cup_k D_k$. This can be seen as a type of training error since the models are trained to answer the queries in Q.

2. Negative Log-likelihood. An alternative is the (mean) negative log-likelihood of the synthetic dataset sampled from our (FL)AIM models when compared to a heldout test set. This metric can be viewed as a measure of generalisation, since the metric is agnostic to the specific workload chosen.

3. Test ROC-AUC. In some cases we evaluate our models by training a gradient boosted decision tree (GBDT) on the synthetic data it produces. We test the performance of the classifier on a test set and evaluate the ROC-AUC.

B.4 Experiment Hyperparameters

B.4.1 CTGAN. In our baseline comparisons in Section 5.1 we use the DP-CTGAN implementation contained in the synthetic data vault (SDV) package [40]. We performed a hyperparameter search over epochs, learning rates and gradient clipping norm. We found training for 20 epochs, with a gradient norm of 1, batch size of 128, discriminator LR of $1e-3$ and generator LR of $1e-5$ gave best performance. For the federated setting we train the DP-CTGAN using DP-FedSGD implemented via the FLSim framework [44]. We found training for 50 epochs with a local batch size of 128, clipping

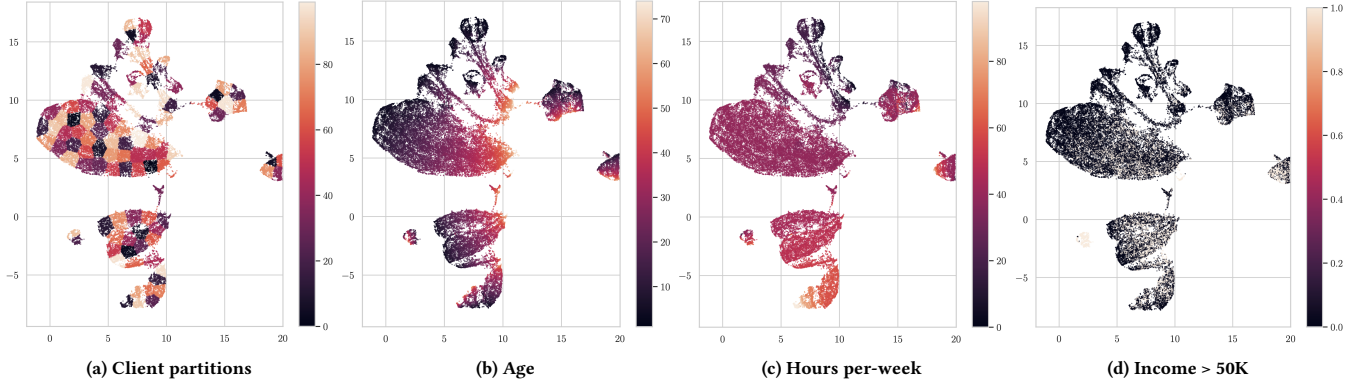**(a) Client partitions** **(b) Age** **(c) Hours per-week** **(d) Income > 50K**

Figure 5: Clustering approach to form non-IID splits on Adult dataset, $K = 100$ clients. All plots show the same embedding formed from UMAP, with Figure 5a showing each client's local dataset formed by clustering in the embedding space. Figures 5b-5d show the same embedding but colored based on three features: age, hours worked per-week and income > 50k. The embedding is used only to map examples to clients, and AIM models are trained on the raw data.

norm of 0.5, server LR of 0.5 and discriminator/generator LRs of $1e-4$ performed best.

B.4.2 (FL)AIM. **PGM Iterations:** The number of PGM iterations determines how many optimisation steps are performed to update the parameters of the graphical model during training. AIM has two parameters, one for the number of training iterations between global rounds of AIM and one for the final number of iterations performed at the end of training. We set this to 100 training iterations and 1000 final iterations. This is notably smaller than the default parameters used in central AIM, but we verified that there is no significant impact on utility.

Model Initialisation: We follow the same procedure as in central AIM, where every 1-way marginal is estimated to initialise the model. Instead in our federated settings, we take a random sample of clients and have them estimate the 1-way marginals and initialise the model from these measurements.

Budget Annealing Initialisation: When using budget annealing, the initial noise is calibrated under a high number of global rounds. In central AIM, initially $T = 16 \cdot d$ results in a large amount of noise until the budget is annealed. We instead set this as $T = 8 \cdot d$ since empirically we have verified that a smaller number of global rounds is better for performance in the federated setting.

Budget Annealing Condition: In central AIM, the budget annealing condition compares the previous model estimate with the new model estimate of the current marginal. If the annealing condition is met, the noise parameters are decreased. In the federated setting, it is possible that PGM receives multiple new marginals at a particular round. We employ the same annealing condition, except we anneal the budget if at least one of the marginals received from the last round passes the check.

C FURTHER EXPERIMENTS

Varying ε: In Figure 6, we vary ε across our datasets under a clustering partition with $K = 100$ clients and $\varepsilon = 1$. These plots replicate Figure 3a across the other datasets. We observe similar patterns

to that of Figure 3a with NaiveFLAIM performing worst across all settings, and our AugFLAIM methods helping correct this to closely match the performance of DistAIM and in some cases even exceed it with lower workload error. There are however some consistent differences when compared to the Adult datasets. For example, on the Magic dataset, AugFLAIM (Private) performance comes very close to DistAIM but there is a consistent gap in workload error. This is in contrast to the Adult dataset where AugFLAIM (Private) shows a more marked improvement.

Varying T: In Figure 7, we vary the global rounds T while fixing $\varepsilon = 1$ and $K = 100$ clients under a clustering partition. This replicates Figure 3b but over the other datasets. Across all figures we plot dashed lines to show the mean workload error under the setting where T is chosen adaptively via budget annealing. On datasets other than Adult, we observe more clearly the choice of T is very significant to the performance of AugFLAIM (Private) and choosing $T > 30$ can result in a large increase in workload error for some datasets (marketing, covtype, intrusion, census). In contrast, increasing T for DistAIM often gives an improvement to the workload error. Recall, DistAIM has participants secret-share their workload answers and these are aggregated over a number of rounds. Hence, as T increases the workload answers DistAIM receives approaches that of the central dataset. For budget annealing, on 3 of the 6 datasets, AugFLAIM (Private) has improved error over NaiveFLAIM but does not always result in performance that matches DistAIM. Instead, it is recommended to choose $T \in [5, 30]$ which has consistently good performance across all of the datasets.

Varying p: In Figure 8 we vary the participation rate p while fixing $\varepsilon = 1, T = 10$ and $K = 100$ clients under a clustering partition. This replicates Figure 3c but across the other datasets. We observe similar patterns as we did on Adult. DistAIM approaches the utility of central AIM as p increases. We note that for NaiveFLAIM, often the workload error does not increase as p increases. Again, as in Figure 3c the likely cause for this is local skew. For AugFLAIM the workload error decreases as p increases and often matches that of DistAIM, except on Magic and Marketing where it stabilises for

Table 6: Budget annealing ranking across workload error and negative log-likelihood. Ranks are averaged across each dataset, with each method repeated 10 times. T is set adaptively via annealing.

Method / ε =	1	2	3	5
NaiveFLAIM	4.65 / 4.75	4.875 / 4.9	4.975 / 4.95	5.0 / 5.0
AugFLAIM (Oracle)	3.6 / 3.3	3.85 / 3.525	3.9 / 3.775	3.925 / 3.6
AugFLAIM (Private)	3.75 / 3.45	3.25 / 3.15	3.125 / 3.125	3.05 / 3.25
DistAIM	2.0 / 2.5	2.025 / 2.425	2.0 / 2.15	2.025 / 2.15
AIM	1.0 / 1.0	1.0 / 1.0	1.0 / 1.0	1.0 / 1.0

Table 7: Total overhead of DistAIM vs AugFLAIM (Private) measure via the average client throughput (total received and sent communication) for $\varepsilon = 1$ and $T = 4, 32, 96$.

Method	$T = 4$	$T = 32$	$T = 96$
Adult	3990x	1223x	410x
Magic	2467x	746x	267x
Intrusion	2313x	630x	233x
Marketing	603x	174x	66x
Covtype	199x	57x	15x
Credit	2363x	714x	220x
Census	832x	221x	77x

$p > 0.3$. Generally, when p is large, DistAIM is preferable but we note this does not correspond to a practical federated setting where sampling rates are typically much smaller ($p < 0.1$) and in this regime DistAIM and AugFLAIM performance is matched.
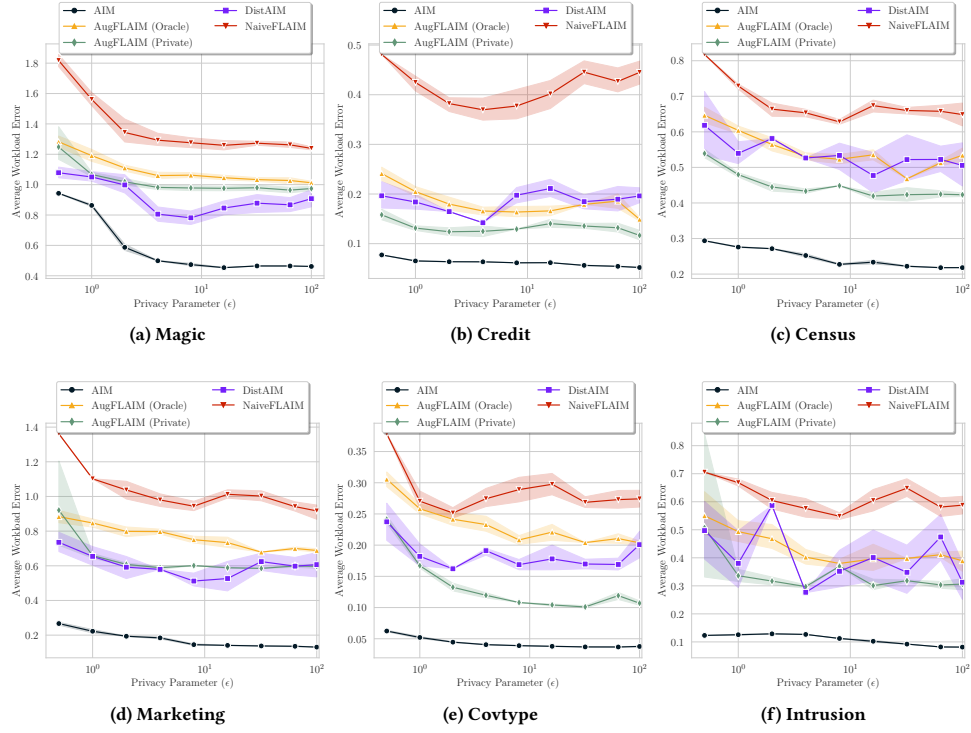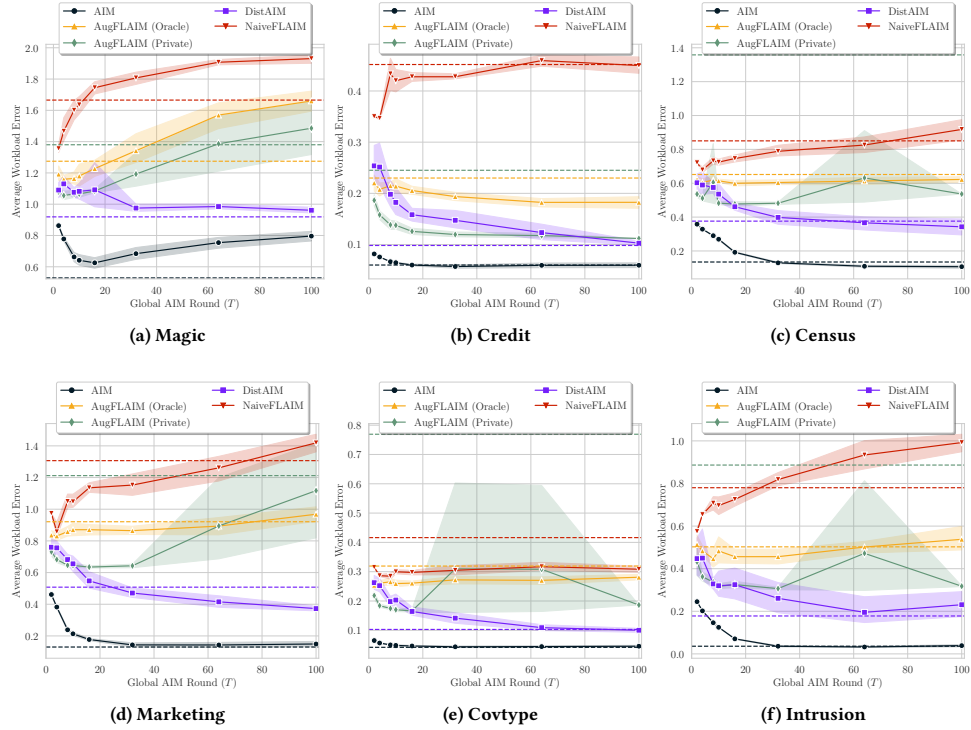
Varying β: In Figure 9, we vary the label-skew partition across datasets via the parameter β. A larger β results in less label-skew and so less heterogeneity. These experiments replicate that of Figure 3d. As before, we clearly observe that NaiveFLAIM is subject to poor performance and that this is particularly the case when there is high skew (small β) in participants' datasets. We can see that the AugFLAIM methods help to stabilise performance and when skew is large ($\beta < 0.1$) can help match DistAIM across the datasets.
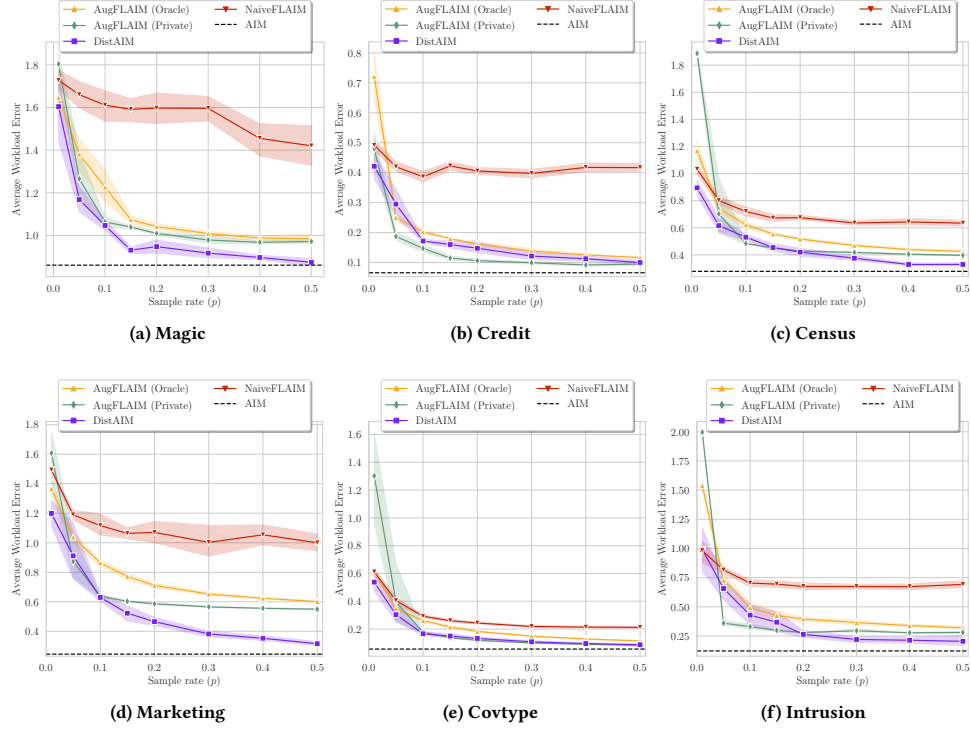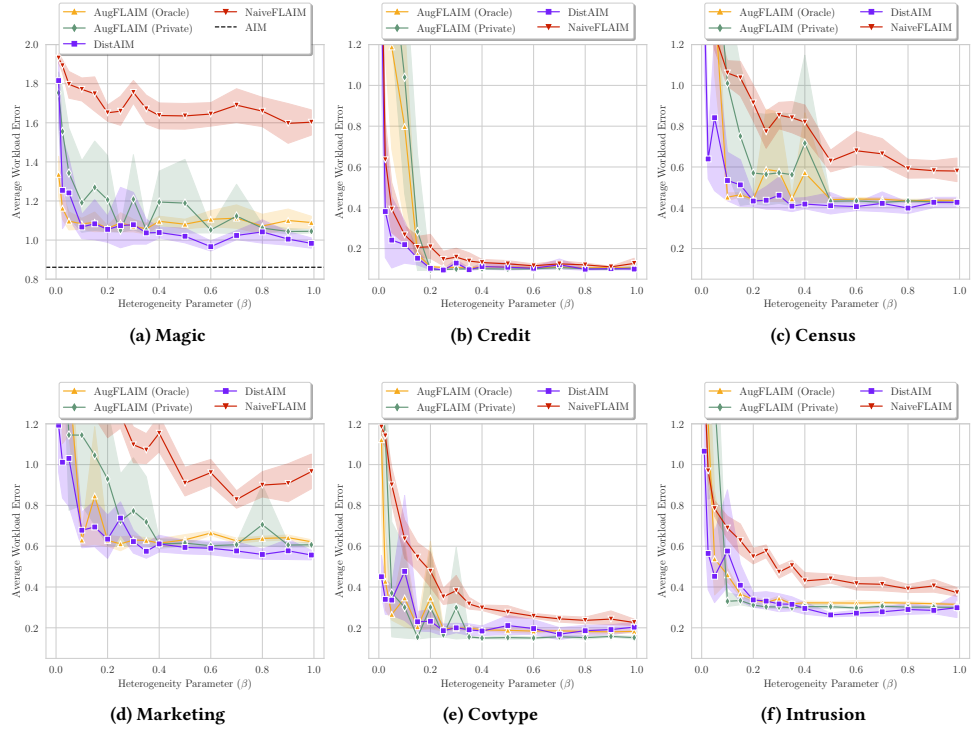
Local updates: In Figures 10 and 11 we vary the local updates $s \in \{1, 4\}$ while fixing $\varepsilon = 1, T = 10$ and $K = 100$. This replicates Figure 3e and 3f but across the other datasets. When using $s = 4$ local rounds, the workload error across methods often increases for NaiveFLAIM and AugFLAIM methods. However, when looking at the test AUC performance, taking $s = 4$ local updates often gives better AUC performance than $s = 1$ on the Census, Magic and Credit datasets. This results in AUC that is closer to that of DistAIM than the other FLAIM methods.
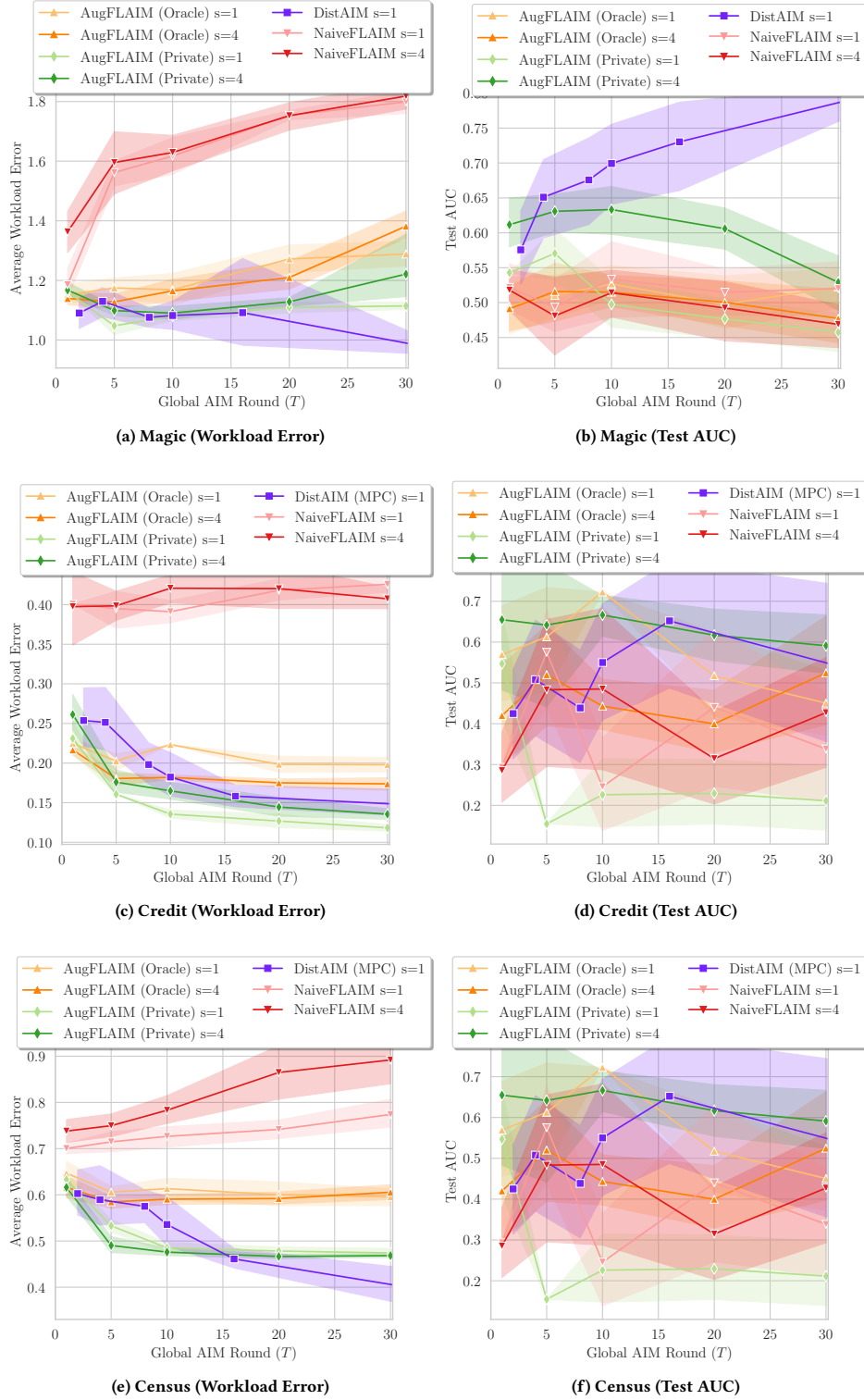
Budget Annealing: In Table 6 we present the average rank of methods across all datasets. We rank based on two metrics: workload error and negative log-likelihood. The number of rounds T is set adaptively via budget annealing. We vary $\varepsilon \in \{1, 2, 3, 4, 5\}$ with the goal of understanding how annealing affects utility across methods. DistAIM achieves the best rank across all settings when using budget annealing, only beaten by central AIM. When ε is small, AugFLAIM (Oracle) achieves a better average ranking across both metrics when compared to AugFLAIM (Private). However, as
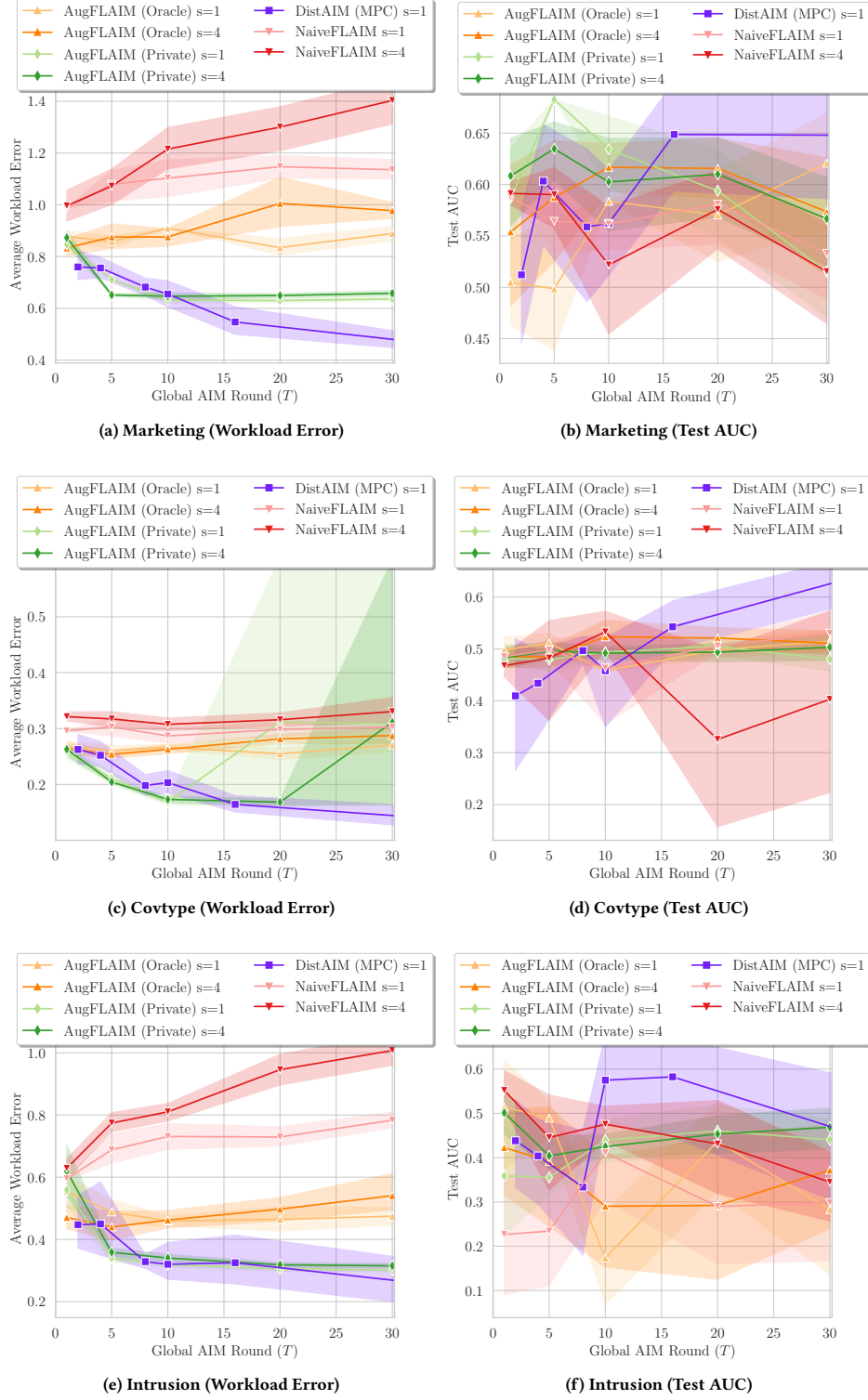
ε increases, AugFLAIM (Private) achieves better rank, only beaten by DistAIM. AugFLAIM (Private) can achieve better performance by choosing T reasonably small ($T < 30$) as previously mentioned. **DistAIM vs. FLAIM Communication:** In Table 7, we present the overhead of DistAIM vs. AugFLAIM (Private) in terms of the average client throughput (total sent and received communication) for $T = 4, 32, 96$. In DistAIM, the amount of communication a client sends is constant no matter the value of T, since they only send secret-shared answers once (when they participate in a round). In the case where the total dimension of a workload is large, the gap in client throughput between AugFLAIM and DistAIM is also large. For example on Adult, clients must send 140Mb in shares whereas AugFLAIM is an order of magnitude smaller. Sending 140Mb of shares may not seem prohibitive but this size quickly scales in the dimensions of features and in practice could be large e.g., on datasets with many continuous features discretized to a reasonable number of bins. Note that if T increases to be very large, eventually AugFLAIM would meet or exceed the communication of DistAIM. However, this would not occur in practice since the best utility is obtained when T is small (e.g., $T < 100$), as observed in Figure 3b.

AugFLAIM (Private) communication is mostly consistent across each dataset for a particular value of T e.g., at $T = 4$ average client throughput is 0.035MB up to 0.5MB at $T = 96$. This is in contrast to DistAIM which varies between 7MB of communication (on Covtype) up to 140MB (on Adult) with the dominating factor for DistAIM being the total dimension of the workload e.g., datasets that have many high cardinality marginals will have large communication overheads under DistAIM.

Figure 6: Varying ε

Figure 7: Varying T

Samuel Maddock, Graham Cormode, and Carsten Maple

Figure 8: Varying p

Figure 9: Varying β

(a) **Magic (Workload Error)**

(b) **Magic (Test AUC)**

(c) **Credit (Workload Error)**

(d) **Credit (Test AUC)**

(e) **Census (Workload Error)**

(f) **Census (Test AUC)**

Figure 10: Varying local rounds $s \in \{1, 4\}$ as in Figure 3e but on alternative datasets.

(a) Marketing (Workload Error)

(b) Marketing (Test AUC)

(c) Covtype (Workload Error)

(d) Covtype (Test AUC)

(e) Intrusion (Workload Error)

(f) Intrusion (Test AUC)

Figure 11: Varying local rounds $s \in \{1, 4\}$ as in Figure 3e but on alternative datasets.