DP-Hype: Federated Differentially Private Hyperparameter Search

Johannes Liebenow
University of Lübeck
Lübeck, Germany
j.liebenow@uni-luebeck.de

Thorsten Peinemann
University of Lübeck
Lübeck, Germany
t.peinemann@uni-luebeck.de

Esfandiar Mohammadi
University of Lübeck
Lübeck, Germany
esfandiar.mohammadi@uni-luebeck.de

Abstract

The tuning of hyperparameters in federated machine learning can substantially impact model performance. When the hyperparameters are tuned on sensitive data, privacy becomes an important challenge and to this end, differential privacy has emerged as the de facto standard for provable privacy. A standard setting when performing federated learning tasks is that clients agree on a shared setup, i.e., find a compromise from a set of hyperparameters, like a model's learning rate. Yet, prior work on privacy-preserving hyperparameter tuning is tailored to specific learning tasks, does not take into account the privacy leakage of aggregated results, or offers a sub-optimal privacy-utility trade-off.

In this work, we present our algorithm DP-Hype, which performs a federated and privacy-preserving hyperparameter search by conducting a federated voting based on local hyperparameter evaluations of clients. In this way, DP-Hype selects hyperparameters that lead to a compromise supported by a majority of clients, while maintaining scalability and independence from specific learning tasks. We prove that DP-Hype preserves the strong notion of differential privacy called client-level differential privacy and, importantly, show that its privacy guarantees do not depend on the number of hyperparameters. We also provide bounds on its utility guarantees, that is, the probability of finding good hyperparameters, and implement DP-Hype as a submodule in the popular Flower framework for federated machine learning. In addition, we evaluate performance on multiple benchmark data sets in iid as well as multiple non-iid settings and demonstrate high utility of DP-Hype even under small privacy budgets.

Keywords

Differential Privacy, Federated Learning, Hyperparameters, Privacy-Preserving Machine Learning

1 Introduction

Federated learning has made significant advances, yet many learning methods have to be fine-tuned to acceptable hyperparameters before achieving strong utility. However, even the process of finding suitable hyperparameters introduces a privacy risk. Outliers can noticeably influence a hyperparameter search, and an adversary can use this information to infer whether certain data points were part of the process or not [33]. Consequently, privacy needs to be preserved. For provable privacy guarantees, differential privacy has become the de facto standard. Conducting a federated and scalable hyperparameter search while preserving differential privacy is a challenging task: The key challenge is that usually a lot of hyperparameters have to be evaluated, but revealing the results of the evaluations to other parties can leak information about training data, thereby deteriorating the differential privacy (DP) guarantees. While in the centralized setting, several techniques [24, 28, 33] have been proposed that are independent of the number of hyperparameters, these techniques rely on one crucial assumption: The adversary is not able to see intermediate evaluation results, but only sees the final hyperparameter that was chosen. This assumption is not satisfied in the federated setting, and it is not clear how to do so in a scalable and federated manner, i.e., without relying on computation parties or trusted third parties, and without paying the well-known $\sqrt{\#\text{clients}}$-price of local DP techniques [4].

In the federated setting, there are essentially two scenarios concerning the data distribution: either the local data distributions are sufficiently similar such that looking for a shared set of hyperparameters makes sense, or the local data distributions are vastly different such that each party (or group of parties) have to search for their own hyperparameter set. In this work, we focus on the first scenario where we aim to find a compromise among clients in terms of hyperparameters, e.g. the model learning rate or its momentum. Importantly, finding a compromise does not imply that clients follow the same data distribution (iid). In reality, their data

Table 1: Overview of related work for privacy-preserving federated hyperparameter search in comparison with DP-Hype. Task Agnostic: The hyperparameter search can be applied to any learning task with some sort of loss function. DP Notion: If the algorithm satisfies differential privacy (DP), which notion does it satisfy exactly. DP Accounting indep. of #HPs: The accounting for the privacy budget does not depend on the (evaluated) number of hyperparameters. Parentheses indicate that the specific property is not fully fulfilled.

Algorithm	Task Agnostic	DP Notion	DP Accounting indep. of #HPs
PrivTuna [32]	✔	✗	-
LDP-DSBO [15]	✗	local	(✔)
DP-FTS-DE [16]	(✔)	central	(✔)
Feathers [36]	✔	(local)	✗
DP-Hype (Ours)	✔	client-level	✔

Proceedings on Privacy Enhancing Technologies 2026(3), 139–159

https://doi.org/10.56553/popets-2026-0075

often differs significantly (non-iid), which means that a federated hyperparameter search algorithm must be able to cope with these variations in local distributions to a reasonable extent.

In this context, there exist some algorithms from prior work that aim for a similar goal (see Table 1). The first solution is PrivTuna [32], which hides intermediate results by using secure multiparty computation (SMPC). However, PrivTuna does not take into account privacy leakage of the final output. Furthermore, LDP-DSBO [15] is designed only to optimize certain continuous hyperparameters in learning setups that require federated model training via gradient descent, and this approach is tailored for local DP, which typically requires adding substantial noise for reasonable privacy budgets. DP-FTS-DE [16] has to rely on a trusted third party to achieve central DP. In addition, each client converges to different hyperparameters, which is why this algorithm is not fully task agnostic as an additional phase would be needed to agree on a shared setup. Feathers [36] aims to find a hyperparameter based on loss differences compared to other candidates. However, taking a closer look at the algorithm reveals that Feathers uses an unbounded loss function without any clipping and its privacy accounting does not include the number of iterations. Thus, local DP cannot be preserved. All prior differentially private algorithms have in common that their privacy accounting, the process of tracking where and how much of the privacy budget is spent, is somewhat dependent on the hyperparameters. For both DP-FTS-DE and LDP-DSBO, properties of the hyperparameter search space affect privacy indirectly by influencing how many iterations are needed to reach a target utility. In LDP-DSBO the dimensionality of the released hyperparameter can also affect privacy directly through sensitivity. However, Feathers has the strongest dependence as the number of evaluated candidates immediately requires composition.

To close this gap, we introduce DP-Hype, a new algorithm for a privacy-preserving federated hyperparameter search. Our approach is based on the observation that if the data distribution per client does not vary too much then local evaluations correlate with the performance of hyperparameters evaluated on the entire data set [32]. Thus, DP-Hype operates in a cross-silo setting where clients have enough computation power to train machine learning models locally. On the technical side, we leverage the voting approach of [45] in combination with a special privacy accounting [23]. Clients evaluate each proposed hyperparameter locally and are allowed to vote for a few of them. Votes are aggregated using a secure summation protocol, which enables DP-Hype to satisfy client-level DP. This counterintuitive trade, i.e., exchanging the expressiveness of global evaluations with local ones, results in a massive privacy amplification. Local training does not have to preserve DP, and we can reduce the required privacy budget to a minimum while still preserving utility. In particular, the privacy guarantees of DP-Hype are independent of the total number of hyperparameters, which is important because the set of hyperparameters is usually large and having to account for each candidate would deteriorate utility quickly. Satisfying client-level DP also offers DP-Hype an improved privacy-utility trade-off as an adversary only sees aggregated results, which effectively improves the signal-to-noise ratio for a growing number of clients. In addition, communication overhead is minimal, as the output of each client only scales linearly in the number of hyperparameters, and our voting approach can be realized with a single invocation of secure summation.

In summary, we make the following contributions:

- We present DP-Hype, the first algorithm for a federated hyperparameter search that is task agnostic, preserves client-level DP, and whose privacy guarantees are independent of the number of hyperparameters. In addition, the simple yet efficient design, including a secure summation protocol, makes DP-Hype scale to many clients.
- We show that DP-Hype satisfies client-level DP and quantify the probability of achieving a good compromise. In addition, we provide insights on the behavior of DP-Hype by simulating various scenarios.
- DP-Hype is implemented as a practical and ready-to-use submodule in Flower, a state-of-the-art framework for federated machine learning.[1]
- We perform an evaluation on three benchmark data sets and show that DP-Hype performs well in iid as well as non-iid scenarios, even under small privacy budgets.

The remainder of this work is structured as follows: In Section 2, we introduce differential privacy in combination with secure summation, our threat model as well as some notation. In Section 3, we present the setting of differentially private federated hyperparameter search, the intuition behind our approach, and the technical details of our algorithm DP-Hype. This is followed by a privacy proof (Section 4) and a utility analysis (Section 5) in combination with an ablation study based on simulated clients. Then, we present our empirical evaluation of DP-Hype on benchmark data sets (Section 6). Afterward, we discuss important design choices of DP-Hype (Section 8). We conclude with an overview of the related work (Section 7) and our conclusion (Section 9).

2 Preliminaries

In this section, we introduce the key concepts used in this work, namely differential privacy as well as secure summation, and present our threat model including some notation.

2.1 Differential Privacy

Differential privacy (DP) was first introduced by [19] and provides a mathematical framework to protect personal information of individuals when applying an algorithm to sensitive data. If an algorithm preserves differential privacy, then each client can plausibly deny that their data was part of the input when the attacker observes a specific output. In a setting where the data set is federated among clients and the server is trusted, there are essentially two scenarios called client-level and record-level DP. Client-level DP aims to hide the influence of an entire client's data, in contrast to record-level DP, where an adversary knows all data points of all clients except for one data point of the targeted client. In this work, we consider client-level DP to achieve the strongest privacy protection.

In mathematical terms, it means that the ratio of output distributions for specific inputs when replacing data from a client is bounded by ε except for a probability mass δ. In this context, ε is called the privacy budget and regulates the degree of privacy

[1] The source code of our implementation is publicly available at https://github.com/UzL-PrivSec/dp-hype.

protection. DP naturally introduces a privacy-utility trade-off as more privacy budget means that the output of the corresponding algorithm can depend more on the input to achieve a higher degree of utility but also less privacy protection, and vice versa.

Let $\mathcal{D}$ be the set of all data sets. Given data sets $D = D_1 \cup \cdots \cup D_n \in \mathcal{D}$ and $D' = (D \setminus D_i) \cup D_i'$ for some client i, the data sets D and D' are called *neighboring*.

Definition 2.1 (Differential Privacy [19]). *Given $\varepsilon \geq 0$ and $\delta \in [0, 1]$, then a randomized algorithm $\mathcal{M} : \mathcal{D} \to \mathcal{Y}$ preserves differential privacy if for all neighboring data sets $D, D' \in \mathcal{D}$ and all measurable sets $S \subseteq \mathcal{Y}$:*

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

Two very useful properties of DP are post-processing and sequential composition. DP is closed under any kind of post-processing as long as it is not based on sensitive data. The composition of multiple DP algorithms also results in a DP algorithm where the exact guarantees depend on the setting, either adaptive or non-adaptive, and on the composition theorem itself [19]. In the worst case, however, ε scales roughly by the square root of the number of compositions while δ composes additively.

We use an alternative flavor of differential privacy, called Rényi-DP (RDP), first introduced by [31]. RDP is based on the Rényi divergence of the output distribution of an algorithm regarding neighboring data sets, and naturally translates to DP.

Definition 2.2 (Rényi Differential Privacy [31]). *A randomized algorithm $\mathcal{M}$ preserves ε-RDP of order α if for all neighboring data sets $D, D' \in \mathcal{D}$:*

$$\mathcal{D}_{\alpha}(\mathcal{M}(D) \parallel \mathcal{M}(D')) \leq \varepsilon$$

where $\mathcal{D}_{\alpha}(. \parallel .)$ is the Rényi divergence.

RDP can be transformed back to approximate DP for any fixed α. To the best of our knowledge, the currently tightest conversion is the following:

Theorem 2.1 (From RDP to DP [3]). *If a randomized algorithm $\mathcal{M}$ satisfies (α, ε)-RDP then it also satisfies $(\varepsilon + \log((\alpha - 1)/\alpha) - (\log \delta + \log \alpha)/(\alpha - 1), \delta)$-DP for any $\delta \in (0, 1)$.*

On this basis, we introduce the Gaussian mechanism, which is an important building block in this work. The mechanism adds Gaussian noise to the output of a function f calibrated on the L2 sensitivity $\Delta_2 f$ of that function and privacy parameters ε, δ. In this context, $\Delta_2 f = \max_{D,D'} ||f(D) - f(D')||_2$ is the worst-case change in the L2 norm in the output that can be observed by the adversary regarding all possible pairs of neighboring data sets.

Definition 2.3 (Gaussian Mechanism [31]). *Let $\mathcal{M} : D \to \mathbb{R}^d$ be defined as $\mathcal{M}(D) = f(D) + \mathbf{z}$ for some function $f : D \to \mathbb{R}^d$ with $\mathbf{z} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_d)$. Then $\mathcal{M}$ satisfies $(\alpha, \frac{\alpha(\Delta_2 f)^2}{2\sigma^2})$-RDP.*

Simulated Central Scenario using SMPC. In the federated learning scenario, without any additional tools, clients have to preserve privacy on their own, which is called local DP (LDP). However, LDP usually requires a lot of noise and can drastically reduce utility [4]. Clients could also entrust the server with their sensitive data, yet this is an unrealistic trust assumption. Consequently, secure multi-party computation (SMPC) is often used. With SMPC, clients can compute a function securely without having to reveal their inputs to the adversary, except for the final result of the computation. Hiding intermediate results can be a massive privacy amplification, as the central scenario can be simulated. In terms of DP, if the adversary can only learn the final result of the computation and cannot single out sensitive information of individual clients, then there is no difference to the central scenario, where the adversary by definition can only observe the final output of the algorithm. However, most SMPC solutions often come at the cost of being scalable to no more than a handful of clients [20, 22, 39, 43]. But in the special case of computing a federated sum, secure summation protocols have proven to be highly scalable with low resource overhead [8]. Throughout this work, we use the terms secure aggregation or secure summation interchangeably.

If each of the n clients locally adds noise drawn from $\mathcal{N}(0, \sigma^2/n)$, then by applying secure summation, the guarantees of the Gaussian mechanism hold although the noise is added partially by each client. This directly follows from the fact that the sum of n independent normal random variables with variance σ^2/n is normal with variance $n \cdot (\sigma^2/n) = \sigma^2$.

2.2 Threat & System Model

We assume a data set $D = D_1 \cup \cdots \cup D_n \in \mathcal{D}$ that is federated among n clients, where each client i holds their own private subset D_i. Important for DP, each user only contributes data to a single client's data set. For ease of presentation, we assume the availability of a server that coordinates the protocol. However, such a server is not mandatory for our algorithm and the same role could also be fulfilled by one or more clients. When results derived from sensitive data need to be shared, clients interact with the server only through a secure summation protocol.

We are in the cross-silo setting, where clients are assumed to be capable of running part of the learning algorithm locally, such as training a model on their private data. This is a standard setting, especially when, for example, gradients are shared with a server, as this already requires local training [29]. The server does not require special computational resources since most of the overhead is caused by the secure summation protocol, which is considered to entail only a small overhead.

We assume an honest-but-curious model for both the server and clients, which is in line with recent work [21, 30]. Our algorithm is explicitly designed to preserve differential privacy without relying on a trusted third party and the adversary is assumed to reside on the server side.

2.3 Notation

In this work, we adopt the following notation. The normal distribution with mean μ and standard deviation σ is denoted by $\mathcal{N}(\mu, \sigma^2)$. The normal CDF with $\mu = 0$ is denoted as Φ_{σ^2}. Vectors are represented using small bold letters. A random vector $\mathbf{z} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_d)$ indicates that $\mathbf{z}$ is sampled from a d-dimensional multivariate normal distribution, where $\mathbf{I}_d \in \mathbb{R}^{d \times d}$ denotes the identity matrix of dimension d. The dth component of a vector $\mathbf{v}$ is accessed as $\mathbf{v}[d]$. Furthermore, we say that $[t]$ denotes the index set $\{1, \ldots, t\} \subseteq \mathbb{N}$.

3 DP-Hype

In this section, we introduce DP-Hype, an algorithm to perform a federated privacy-preserving hyperparameter search. First, we discuss the setting of such a hyperparameter search. Then, we explore the high-level idea behind DP-Hype and present technical details as well as the pseudocode.

3.1 Problem Statement

Hyperparameter search, also referred to as hyperparameter optimization or tuning, is the task of finding hyperparameters for a specific learning problem that maximize the utility of this task. Formally, let $H \subseteq \mathcal{H}$ denote a set of candidate hyperparameters from the space of possible hyperparameters $\mathcal{H}$. The goal is to identify a hyperparameter $H_{j^*} \in H$ that minimizes a loss function $\mathcal{L} : \mathcal{H} \times \mathcal{D} \to \mathbb{R}$ on a given data set $D \in \mathcal{D}$ such that

$$H_{j^*} = \underset{H_j \in H}{\operatorname{argmin}}\ \mathcal{L}(D, H_j). \tag{1}$$

This is the centralized formulation of hyperparameter search, but it becomes substantially more challenging in federated settings. In federated learning, hyperparameter search can manifest in two fundamentally different ways. If the data across clients is highly non-iid, then different subpopulations typically emerge, each requiring their own hyperparameters, which may differ significantly from those of other clients. In such cases, an additional phase is needed to identify subpopulations and assign suitable hyperparameters to them. On the other hand, when the clients' data does not differ too much, hyperparameter search reduces to finding common hyperparameters that allow clients to perform a learning task jointly. Many federated algorithms such as FedAvg [29] or FedAdam [34] rely on clients sharing a similar setup, which essentially requires a compromise over hyperparameters. Such a scenario, where clients must agree on shared hyperparameters in a federated manner, is the focus of this work.

Before introducing our approach, it is important to understand why naively optimizing Equation (1) in a federated context is infeasible in a realistic scenario, i.e., when the number of hyperparameters is large. Without structural knowledge about the behavior of $\mathcal{L}(D, H)$, such as monotonicity or the number and locations of optima, the standard strategy is to train a model in a federated and differentially private manner on every hyperparameter candidate and select the best one. However, this strategy suffers from severe privacy constraints: The privacy budget must be divided among iterations, roughly scaling with the square root of the number of iterations [19]. Since differentially private federated algorithms already consume a large privacy budget to maintain utility [21], even few iterations lead to an impractically small budget due to composition. Moreover, not every loss function can be published out-of-the-box in a private way, as a bounded sensitivity is mandatory. Even if a bound is enforced, e.g., through clipping, it does not automatically guarantee a meaningful privacy-utility trade-off. Furthermore, private selection algorithms like [28] or its extension [33], which allow the privacy accounting to be independent of the number of iterations, cannot be applied either. This is because they share the assumption that an adversary does not know when the algorithm has stopped and whether this happened due to random stopping or because the number of iterations or a threshold was reached. Consequently, straightforward iterative hyperparameter search cannot be directly applied in federated and privacy-preserving settings.

3.2 Approach

Evaluating each hyperparameter on the entire data set in a federated setting is infeasible under strict privacy constraints. To address this, we relax the formulation in Equation (1) and instead consider the following objective:

$$H_{j^*} = \underset{H_j \in H}{\operatorname{argmax}} \left|\{i \in [n] \mid H_j = \underset{H_t \in H}{\operatorname{argmin}}\ \mathcal{L}(D_i, H_t)\}\right|. \tag{2}$$

Clients evaluate each hyperparameter $H_t \in H$ locally on their data set D_i and select the hyperparameter that reaches a minimum loss such that H_{j^*} is the hyperparameter most clients choose locally. Throughout this work, we refer to those hyperparameters as good. As we show later, optimizing Equation (2) instead of Equation (1) yields strong privacy amplification and produces high-quality hyperparameters that closely align with those obtained by optimizing the global objective. A related idea was introduced in [32], where hyperparameters were locally tuned and averaged by the server, but that approach did not analyze any privacy implications.

A closer inspection of Equation (2) reveals a key advantage regarding privacy: we no longer require training a global model or computing the global loss. The only information needed is whether clients independently favor a particular hyperparameter. This naturally transforms the problem into a voting task, where clients act as voters and hyperparameters represent the candidates. This perspective provides an avenue to leverage well-established techniques from federated, differentially private voting [45].

Voting offers several notable benefits in this setting. First, differential privacy is naturally supported: if each client is restricted to vote for at most k hyperparameters, then the L2-sensitivity, defined as the maximum change in the output due to replacing one client's data, only grows with $\sqrt{k}$. Second, communication overhead remains low, since clients only need to send a scalar per candidate. However, a naive implementation would still require allocating privacy budget separately for each candidate, which is prohibitive when the number of hyperparameters is large. To overcome this, we combine voting with the privacy accounting framework of [23, 45], which allows us to account for the aggregated voting result as a whole rather than per candidate. This results in additional privacy amplification, since the privacy cost depends only on the number of votes k and not on the total number of hyperparameters. The final ingredient is secure aggregation: The privacy guarantees from voting and special accounting hold only if no individual votes are revealed to the server. To ensure this, we employ a secure summation protocol such as [8] that computes the aggregated vote counts per hyperparameter without exposing individual contributions.

3.3 DP-Hype

We now introduce our algorithm DP-Hype, presented in Algorithm 1, which implements a federated and privacy-preserving hyperparameter search. The algorithm receives as input the federated data set D, the set of hyperparameters $H = \{H_1, \ldots, H_p\}$, a loss function $\mathcal{L} : \mathcal{D} \times \mathcal{H} \to \mathbb{R}$, privacy parameters ε, δ, and the number of local votes k. Its output is a single hyperparameter

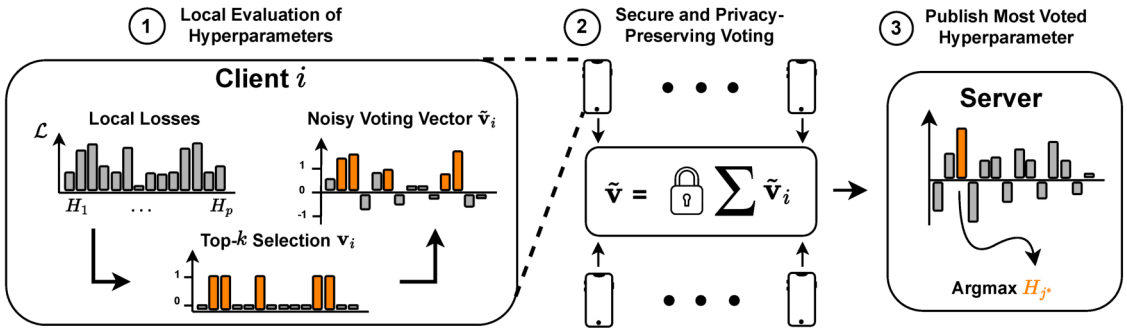


Figure 1: A conceptual overview of our privacy-preserving algorithm for federated hyperparameter search DP-Hype. The set of hyperparameters $H = \{H_1, \ldots, H_p\}$ and the loss function $\mathcal{L}$ for the given learning task are publicly available. First, each client locally computes the loss for each hyperparameter and selects the top-k hyperparameters with the smallest loss. A voting vector $\mathbf{v}_i$ is created, with a 1 on each position corresponding to the top-k hyperparameters and a 0 elsewhere. Each $\mathbf{v}_i$ is noised and entry-wise aggregated on the server side via secure summation to obtain $\tilde{\mathbf{v}}$. The server then outputs the hyperparameter with the most noisy votes.

$H_{j^*} \in H$ that obtains the largest number of noisy votes aggregated across clients. The procedure consists of two phases: a client phase and a server phase. In the client phase, each client i evaluates all hyperparameters H_j on their local data D_i to compute the losses $(\mathcal{L}(D_i, H_1), \ldots, \mathcal{L}(D_i, H_p))$. A local voting vector $\mathbf{v}_i$ of length p is initialized with zeros, and the entries corresponding to the k smallest losses are set to one. To ensure differential privacy, each local voting vector is perturbed by Gaussian noise scaled by σ, yielding a noisy version $\tilde{\mathbf{v}}_i$. The noise scale σ is based on k, ε, δ and can be determined using Theorem 2.1. After this step, clients jointly execute a secure summation protocol with the server, which ensures that only the aggregated noisy vote vector $\tilde{\mathbf{v}}$ is revealed, while no individual votes are exposed. Since the sum of Gaussian random variables is Gaussian, the added noise guarantees differential privacy. Finally, the server selects the hyperparameter H_{j^*} with the maximum number of aggregated votes.

All k local votes are equally weighted and are not ranked on purpose. A ranking would either increase the sensitivity or smaller ranked votes would quickly be drowned by noise. We empirically show the downsides of such a weighted voting in Appendix F. Locally evaluating each hyperparameter introduces a computational overhead for clients, but in Appendix H we show that this overhead is mandatory for achieving a good performance.

DP-Hype is agnostic to the specific learning task, requiring only that local and global evaluations of a loss function are available. For ease of presentation, we describe it in terms of minimizing the loss, but the procedure can equivalently be applied to maximization. Both the set of hyperparameters and their order are assumed to be public, enabling standard search strategies such as random or grid search by reordering candidates. Importantly, the local learning process does not need to satisfy differential privacy and can be performed without any modification, thereby avoiding unnecessary privacy-utility trade-offs. Noise is only introduced when information leaves the clients, during the voting stage. Moreover, communication overhead is minimal, as DP-Hype requires only a single invocation of secure summation, making it lightweight in practice.

The modular design of DP-Hype makes it independent of any specific secure aggregation protocol, as long as it supports summation. This reduces the deployment overhead of DP-Hype and allows selecting a protocol that fits the individual requirements best. A discussion of why we do not rely on a specific protocol can be found in Section 8. Possible extensions for DP-Hype regarding malicious clients or server and robustness are moved to Appendix A due to space limitations.

Algorithm 1: DP-Hype

Data: Data set $D = D_1 \cup \cdots \cup D_n$, hyperparameter set $H = \{H_1, \ldots, H_p\}$, loss function $\mathcal{L} : \mathcal{D} \times \mathcal{H} \to \mathbb{R}$, DP-parameters ε, δ, number of local votes k

Result: $H_{j^*} \in H$

Clients:

Determine σ based on ε, δ, k (Theorem 2.1).
for i **in** $[n]$ **:**
  for j **in** $[p]$ **:**
    $\text{loss}_j = \mathcal{L}(D_i, H_j)$
  $\mathbf{v}_i = (0, \ldots, 0) \in \mathbb{R}^p$
  while $||\mathbf{v}_i||_1 < k$ **do**
    $j^* = \underset{j \in [p] \wedge \mathbf{v}_i[j]=0}{\text{argmin}} \text{loss}_j$
    $\mathbf{v}_i[j^*] = 1$
  $\tilde{\mathbf{v}}_i = \mathbf{v}_i + \mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(0, \frac{\sigma^2}{n}\mathbf{I}_p)$

Server:

$\tilde{\mathbf{v}} = (\sum_{i=1}^n \tilde{\mathbf{v}}_i[1], \ldots, \sum_{i=1}^n \tilde{\mathbf{v}}_i[p])$ with SecSum
$H_{j^*} = \underset{j \in [p]}{\text{argmax}}\, \tilde{\mathbf{v}}[j]$
return H_{j^*}

4 Privacy Proof

In this section, we prove that DP-Hype satisfies client-level (α, ε')-RDP. Guarantees for client-level (ε, δ)-DP can then be derived using Theorem 2.1. Before starting with the proof, we give an overview of the proof structure.

4.1 Proof Outline

Denote DP-Hype by $\mathcal{M}$, to prove that $\mathcal{M}$ satisfies RDP we have to bound the Rényi divergence $\mathcal{D}_\alpha(\mathcal{M}(D) \parallel \mathcal{M}(D')) \leq \varepsilon'$ for some α and ε', and for any pair of neighboring data sets D, D'. From an algorithmic point of view, $\mathcal{M}$ outputs a single hyperparameter $\mathcal{M}(D) = H_{j^*} \in H$. However, from the perspective of an adversary, in our scenario the server, the output consists of much more. Since clients use secure summation to vote for each hyperparameter individually, the adversary actually learns $\mathcal{M}(D) = \tilde{\mathbf{v}} \in \mathbb{R}^p$.

So when analyzing the privacy guarantees of our algorithm, we have to take into account that the adversary can observe the entire noisy voting vector $\tilde{\mathbf{v}}$, i.e., the voting result for every single hyperparameter. To avoid doing the accounting for every single hyperparameter and thereby run into composition, we consider the vector as a whole. The Gaussian mechanism allows us to do this if we consider the L2 sensitivity. This is the maximum change in the L2 norm of the vector that can happen in the worst case regarding all possible pairs of neighboring data sets.

4.2 Proof

We start by analyzing the L2 sensitivity of the voting vector $\mathbf{v}$ without any noise and show that due to the nature of the binary voting system and a fixed number of k votes per client $\Delta_2\mathbf{v} = \sqrt{2k}$.

Lemma 4.1 (L2-Sensitivity of $\mathbf{v}$). *Given the fixed number of local votes k, then the voting vector $\mathbf{v}$ without any noise has L2-sensitivity $\Delta_2\mathbf{v} = \sqrt{2k}$.*

Proof. Let $\mathbf{v} \in \mathbb{R}^p$ be the aggregated voting vector without any noise and denote DP-Hype as $\mathcal{M}$. Further, let p denote the number of hyperparameters, n the number of clients and let $\mathbf{v}_i[j] \in \{0, 1\}$ be the individual vote of client i for hyperparameter j before adding noise. In abuse of notation, we write $\mathbf{v}, \mathbf{v}'$ when based on D, D', respectively.

$$
\begin{aligned}
\Delta_2\mathbf{v} &= \max_{D\sim D'} ||\mathcal{M}(D) - \mathcal{M}(D')||_2 \\
&= \max_{D\sim D'} ||\mathbf{v} - \mathbf{v}'||_2 = \max_{D\sim D'} \sqrt{\sum_{j=1}^{p}\sum_{i=1}^{n}(\mathbf{v}_i[j] - \mathbf{v}'_i[j])^2} \\
&\overset{(a)}{=} \max_{D\sim D'} \sqrt{\sum_{j=1}^{p}(\mathbf{v}_u[j] - \mathbf{v}'_u[j])^2} \\
&\overset{(b)}{=} \max_{D\sim D'} \sqrt{\sum_{j=1}^{2k}(\mathbf{v}_u[j] - \mathbf{v}'_u[j])^2} \\
&\leq \sqrt{k\cdot(-1)^2 + k\cdot(1)^2} = \sqrt{2k}
\end{aligned}
$$

(a) By the definition of neighboring data sets, the entire data set D_u of a single client u can be replaced. (b) As for each client's voting vector $\mathbf{v}$ it holds that $||\mathbf{v}_i||_1 = k$, $2k$ entries in $\mathbf{v}$ change at maximum. Without loss of generality, we can assume that those entries are the first $2k$. The last inequality comes from the fact that clients can only choose between 0 and 1 for each hyperparameter and that in the worst case, k ones flip to zeros and k zeros to ones. □

Based on the fact that the output of DP-Hype has a bounded L2-sensitivity, we can now prove that DP-Hype satisfies RDP.

Theorem 4.1. *Given $k \in \mathbb{N}$ and let $\alpha > 1$ then DP-Hype denoted as $\mathcal{M}$ satisfies client-level (α, ε')-RDP with*

$$\varepsilon' = \mathcal{D}_\alpha(\mathcal{M}(D) \parallel \mathcal{M}(D')) = \frac{\alpha k}{\sigma^2}$$

for variance σ^2 and all neighboring data sets D, D'.

Proof. The proof follows directly from Definition 2.3 by setting $\Delta_2 = \sqrt{2k}$. □

The noise scale σ used in line 9 of Algorithm 1 can then be determined based on Theorem 2.1. To obtain (ε, δ)-DP, Theorem 2.1 can be used.

5 Utility Analysis

In this section, we first quantify the probability of DP-Hype achieving a good compromise and then go into detail about the parameter k by simulating clients under various scenarios.

5.1 Selecting a Good Hyperparameter

When considering the set of hyperparameters provided as input to DP-Hype, we first note that most learning tasks do not admit a single hyperparameter that performs well while all others perform poorly. Instead, there typically exists a subset of candidates that yield good utility, though this subset is not too large, otherwise hyperparameter search would be trivial and random guessing would suffice. The objective of DP-Hype is therefore to select one of these good hyperparameters such that a compromise is achieved across a majority of clients. However, because differential privacy requires that each client perturbs their local voting vector $\mathbf{v}_i$, there remains the possibility that the added noise results in selecting a bad hyperparameter, particularly under tight privacy budgets. To formalize this, we denote by $H_{\text{good}} \subseteq H$ the set of hyperparameters with good utility, and by $H_{\text{bad}} \subseteq H \setminus H_{\text{good}}$ the set of bad ones. Importantly, it is not necessary to return exactly the hyperparameter with the most votes; any hyperparameter with a similar vote count is also acceptable, as it is likely to yield a comparable compromise. We therefore define the gap between the minimum number of votes for any good and the maximum number of votes for any bad hyperparameter based on $\mathbf{v}$, the aggregated votes without noise, as

$$\gamma := \min_{H_i \in H_{\text{good}}} (\mathbf{v}[i]) - \max_{H_j \in H_{\text{bad}}} (\mathbf{v}[j]). \tag{3}$$

Here, the ith component of $\mathbf{v}$ corresponds to the total number of votes for hyperparameter i without noise. Based on this definition, the utility of DP-Hype, denoted by $\mathcal{M}$, can be expressed as $\Pr\big[\mathcal{M}(D) \in H_{\text{good}}\big]$, i.e., the probability that the algorithm outputs a good hyperparameter, when conditioning on the aggregated votes $\mathbf{v}$. Building on prior work on private federated voting [45], we adapt their analysis and obtain the following result:

Theorem 5.1 (Selecting a Good Hyperparameter). *Given a data set D, a set of hyperparameters H containing a subset of bad candidates H_{bad}, and defining γ as in Equation* (3)*, then conditioned on the aggregated votes, DP-Hype, denoted as $\mathcal{M}$, with noise scale σ outputs a good hyperparameter with probability*

$$\Pr\big[\mathcal{M}(D) \in H_{\text{good}}\big] \geq 1 - \frac{|H_{\text{bad}}|\sigma}{\gamma\sqrt{\pi}} \exp\left(-\frac{\gamma^2}{4\sigma^2}\right).$$

Proof. The proof follows by modeling the aggregated noisy voting vector as $\tilde{\mathbf{v}} = \mathbf{v} + \mathbf{z}$ with $\mathbf{z} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_p)$, leveraging the guarantees of the secure summation protocol. If DP-Hype outputs a bad hyperparameter, then $\mathbf{z}[i] - \mathbf{z}[j^*] \geq \gamma$ for some $H_i \in H_{\text{bad}}$ and $j^* = \text{argmin}_{H_j \in H_{\text{good}}} \mathbf{v}[j]$. Using the fact that $(\mathbf{z}[i] - \mathbf{z}[j^*]) \sim \mathcal{N}(0, 2\sigma^2)$ and using a standard Gaussian tail bound, we get that $1 - \Phi_{2\sigma^2}(\gamma) \leq \frac{\sigma}{\gamma\sqrt{\pi}} e^{-\frac{\gamma^2}{4\sigma^2}}$ where Φ denotes the CDF of the normal distribution with variance $2\sigma^2$. Applying a union bound over all bad candidates concludes the proof. □

Importantly, the event that the noise overcomes γ at least once is a necessary condition that must occur whenever DP-Hype outputs a bad hyperparameter, even though it is not a sufficient condition on its own. Thus, we can safely use this event in the proof to upper-bound the failure probability and derive a lower bound for the success probability of DP-Hype. An immediate consequence of this analysis is that the number of clients does not affect the noise magnitude due to secure summation. Furthermore, the probability of DP-Hype selecting a good hyperparameter grows exponentially with the gap γ, which aligns with intuition: When the separation between good and bad hyperparameters is large, noise is unlikely to reverse the outcome, even under limited privacy budgets. Conversely, the success probability decreases as the number of bad hyperparameters grows, since each additional candidate introduces another chance for noise to shift the outcome incorrectly. Note that σ is directly influenced by k as k contributes to the sensitivity of $\mathbf{v}$, which is why the trade-off between privacy and utility is also visible indirectly in this bound.

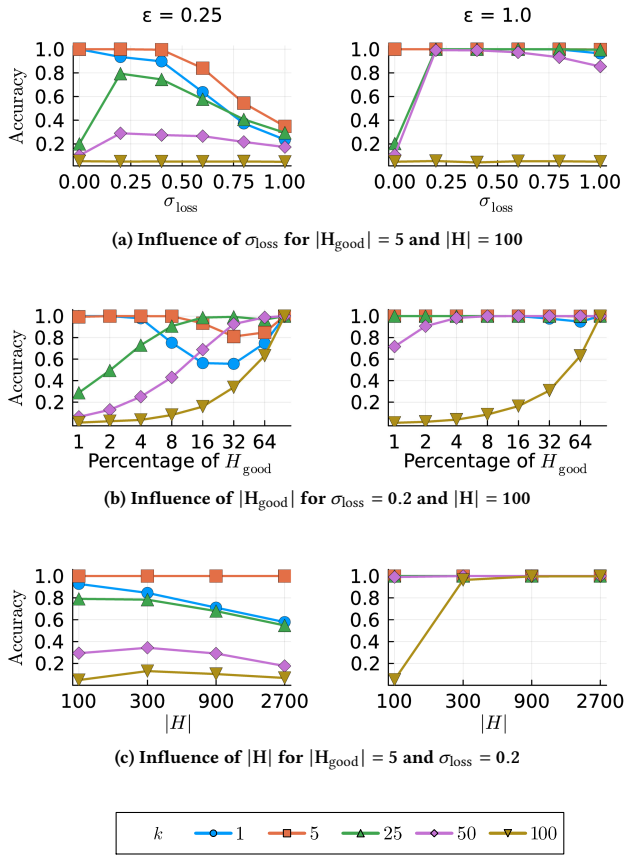
5.2 The Effect of the Maximum Number of Local Votes

We now turn to the role of the parameter k in DP-Hype. Intuitively, k influences the utility of the algorithm but, as shown in Section 4, it also directly correlates with privacy leakage, thereby introducing a privacy–utility trade-off. In the case of iid data distributions, the choice of k has little impact, since a majority of clients will naturally agree on a specific hyperparameter, and a single vote per client would suffice. In such settings, the gap γ from Equation (3) is typically close to n, which ensures robustness even under small privacy budgets. The situation becomes more interesting in non-iid settings, where distinct subpopulations of clients may prefer different hyperparameters. In this case, a hyperparameter representing a compromise might not receive the highest number of votes but could still perform well overall, for example by being the second- or third-best choice for many clients. The parameter k is designed to capture precisely these scenarios by allowing clients to vote for multiple good candidates rather than only their top choice. However, the effectiveness of this mechanism depends heavily on the underlying data distributions and client-specific losses, making it difficult to provide general theoretical guarantees.

To illustrate the impact of k, we perform simulations under controlled conditions. We assume that the local losses of good and bad hyperparameters follow normal distributions with means $\mu_1 = 0$ and $\mu_2 = 1$, respectively, and equal standard deviation σ_{loss}. Each client draws $|H|$ local losses, $|H_{\text{good}}|$ good and $|H_{\text{bad}}|$ bad ones. The federated voting part of DP-Hype is executed on the local losses without local training. After 5000 repetitions, we estimate the success probability from Theorem 5.1 by counting how often DP-Hype selects a hyperparameter from H_{good} and divide it by the total number of repetitions. The experimental setup consists of $n = 250$ clients in total and varying values of k. In addition, we vary the privacy budget $\varepsilon \in \{0.25, 1\}$ while fixing $\delta = 10^{-5}$. We choose such small privacy budgets on purpose as most effects of DP-Hype like the privacy-utility trade-off only become visible for $\varepsilon \leq 1$. These simulations aim to provide empirical evidence of how k impacts the privacy-utility trade-off in iid and non-iid regimes.

IID vs. Non-IID. We begin by studying the effect of varying the degree of non-iid, while deferring the analysis of different sample sizes and class distributions per client to the evaluation section. To simulate this setting, we increase the variance σ^2_{loss} of the local loss distributions gradually, causing the losses of good and bad hyperparameters to overlap more. In other words, the losses of good and bad hyperparameters become harder to distinguish, which effectively simulates scenarios where clients do not vote unanimously. We fix the number of good hyperparameters to $|H_{\text{good}}| = 5$, and the results are reported in Figure 2a. The experiments show that larger values of k improve accuracy when the overlap between good and bad hyperparameters increases. However, as k grows too large, the privacy–utility trade-off becomes apparent: although a higher k may capture more compromise solutions, it also induces stronger noise and forces clients to vote for bad hyperparameters when good ones are scarce. This effect is most pronounced for $k = 100$, where accuracy drops nearly to zero since only five hyperparameters are good and their votes are overwhelmed by those for bad candidates. Importantly, the privacy–utility trade-off manifests only for very small values of ε, which demonstrates that DP-Hype can still achieve high accuracy under tight privacy budgets.

Fraction of Good Hyperparameters. Next, we investigate how the number of good hyperparameters influences the performance of DP-Hype. To this end, we fix $\sigma = 0.2$ to simulate a more practical scenario without only unanimous votes and vary the number of good hyperparameters as $|H_{\text{good}}| = 2^i$ for $i \in [7]$. The results are shown in Figure 2b. The findings confirm that the privacy–utility trade-off becomes relevant only for very small values of ε. Furthermore, for $k = 100$ the accuracy does not exceed random guessing (appearing as a flat curve rather than a diagonal due to the log scale), which is expected since clients are forced to vote for all hyperparameters, leaving the server's argmax decision dominated by noise. For intermediate choices of k, the privacy–utility trade-off becomes visible. In addition, larger choices of k only perform well for a growing $|H_{\text{good}}|$. This is because if the number of good hyperparameters is much smaller than k then clients are forced to vote for bad hyperparameters. Also note the dip observable around $k = 1$ with $|H_{\text{good}}| > 4$ and slightly smaller for $k = 5$ with $|H_{\text{good}}| > 16$. If there



(a) Influence of σ_{loss} for $|\mathbf{H}_{\text{good}}| = 5$ and $|\mathbf{H}| = 100$

(b) Influence of $|\mathbf{H}_{\text{good}}|$ for $\sigma_{\text{loss}} = 0.2$ and $|\mathbf{H}| = 100$

(c) Influence of $|\mathbf{H}|$ for $|\mathbf{H}_{\text{good}}| = 5$ and $\sigma_{\text{loss}} = 0.2$

Figure 2: Simulating (a) a varying overlap between good and bad losses with $|H_{\text{good}}| = 5, |H| = 100$, (b) a varying percentage of good hyperparameters with $\sigma_{\text{loss}} = 0.2, |H| = 100$ and (c) a varying number of hyperparameters in total with $|H_{\text{good}}| = 5, \sigma_{\text{loss}} = 0.2$. All under varying privacy budgets ε with $\delta = 10^{-5}$ and $n = 250$. We sample $|H_{\text{good}}|$ and $|H_{\text{bad}}|$ local losses ($|H|$ in total) from $\mathcal{N}(0, \sigma^2_{\text{loss}})$ and $\mathcal{N}(1, \sigma^2_{\text{loss}})$, respectively. To show the impact of k on utility, we measure the accuracy of DP-Hype, i.e., how often it selects a loss pointing to a good hyperparameter. A varying σ_{loss} creates different degrees of overlapping of both loss distributions, which uncovers the advantage of $k > 1$. Varying H_{good} reveals that smaller k are favorable in case of a small number of good hyperparameters. Lastly, increasing the set of hyperparameters while leaving $|H_{\text{good}}|$ fixed, shows that the privacy-utility trade-off is only marginally influenced by the number of hyperparameters.

are a lot of good hyperparameters then the votes are distributed among these candidates, which in turn decreases the maximum votes for a single hyperparameter and thus decreases accuracy. The accuracy only increases again when bad hyperparameters are becoming rarer.

Total Number of Hyperparameters. Lastly, we study how the total number of hyperparameter candidates affects performance by varying $|H| \in \{100, 300, 900, 2700\}$ while keeping $|H_{\text{good}}| = 5$ and $\sigma_{\text{loss}} = 0.2$ fixed. The results are shown in Figure 2c. For privacy budget $\varepsilon = 1$, only $k = 100$ is noticeably affected, because forcing clients to vote across essentially all candidates becomes close to random guessing, yielding an expected accuracy near $5/100 = 0.02$. For smaller privacy budgets, a smaller k is preferable since fewer votes require less noise, and the results clearly indicate that the privacy–utility trade-off is largely independent of the total number of hyperparameters. Moreover, for $k > 5$ the accuracy initially worsens up to $|H| = 100$, because many clients spend a substantial fraction of their votes on the same few bad candidates, increasing the probability that a bad candidate is selected. This effect vanishes as $|H|$ grows and votes for bad candidates become more dispersed. The remaining slight downward trend in accuracy for larger $|H|$ can be attributed to an increased chance that, as more local losses are sampled, some bad candidate attains a loss close to that of a good candidate, making local ranking and thus voting more error-prone. Interestingly, if k aligns with $|H_{\text{good}}|$ then the accuracy does not decrease at all. However, beyond some point, as $|H|$ grows, it becomes increasingly likely that a poorly performing configuration attains a bad loss that falls within the range of the good losses. As a result, the accuracy of DP-Hype eventually decreases.

Selecting k. DP-Hype is deliberately designed to minimize the number of its own hyperparameters, since in differentially private hyperparameter search each additional parameter would itself require tuning, consuming part of the privacy budget and leaving less available for the actual search and learning task. Our simulations show that the choice of k becomes critical only in very small privacy budget regimes and that choosing a k that is slightly off the potential best choice does not break DP-Hype but only decreases utility by a small amount. Figure 2 clearly shows that as long as k is small DP-Hype has a high probability of selecting a good hyperparameter.

6 Evaluation

In this section, we provide further insights on how DP-Hype is implemented in the federated learning framework Flower, and present our empirical evaluation on benchmark data sets in iid and non-iid scenarios. These experiments are focused on the privacy-utility trade-off and the resource consumption of DP-Hype. Additional ablations are moved to Appendices F to H due to space limitations.

6.1 Flower Implementation

In general, Flower [10] simulates each client as a separate process, eliminating the need for managing many physical devices for clients. The server performs multiple iterations, and in each iteration a so-called strategy manages the configuration of clients, calls clients when their participation is needed, and processes the clients' results. Each client locally trains on their local share of data and sends their results to the server. We implement DP-Hype as such a strategy, a submodule in Flower, and we implement clients to be compatible with other learning tasks in addition to the ones we evaluate here. DP-Hype outputs a hyperparameter configuration that is then passed to the subsequent federated learning task. Inside Flower, DP-Hype is implemented completely using PyTorch for model training. The source code of our implementation is publicly available at https://github.com/UzL-PrivSec/dp-hype.

Table 2: An overview of the three evaluated data sets including the number of data points used for training, the number of attributes and the number of classes.

Data Set	# Data Points	Data Point Size	# Classes
MNIST [26]	60,000	28×28 (grayscale)	10
Cifar-10 [25]	60,000	32×32 (RGB)	10
Adult [7]	$\approx$ 50,000	14 (mixed attributes)	2

Table 3: The set of hyperparameters to be evaluated is the cross-product of 10 learning rate, 5 learning rate decay and 2 momentum candidates, resulting in 100 candidates in total.

Hyperparameter	Candidates
Learning rate	$\{0.5, 0.1, 0.05, 1 \times 10^{-3}, 5 \times 10^{-3}, 1 \times 10^{-5}, 1 \times 10^{-6}, 5 \times 10^{-6}, 5 \times 10^{-7}, 1 \times 10^{-7}\}$
Learning rate decay	$\{0.0, 0.1, 0.25, 0.99, 1.0\}$
Momentum	$\{0, 0.9\}$
Total combinations	$\mathbf{10 \cdot 5 \cdot 2 = 100}$

6.2 Experimental Setup

Data Sets and Learning Tasks. We use the following data sets to evaluate the performance of DP-Hype. The MNIST [26] data set which contains gray-scale images of handwritten digits, Cifar-10 [25], also an image data set, but with three color channels and images containing objects and animals, and Adult [7], which is of tabular form with a mix of categorical and continuous census attributes. An overview of the data sets and their properties is shown in Table 2. For MNIST we use a small convolutional model and for Cifar-10 a slightly larger one, to maintain a reasonable balance between accuracy and training time. In both cases, data are standardized, and the models are trained on a classification task to differentiate between 10 classes. For Adult, the data is also standardized, categorical attributes are one-hot-encoded and the attribute income is mapped into a binary attribute indicating whether income is above 50, 000. We then train a small MLP with dropout to learn the binary classification task for the attribute income. We refer to Appendix B for details on model architectures. In all learning tasks, we use a train-test-split, such that 20% of the data are used for testing and the rest for training. The training was performed on an NVIDIA A100-20C GPU with 20GB VRAM by using the simulation module of Flower to train clients in parallel.

Hyperparameters. We use a specific set of hyperparameters to mirror a realistic setup for training a neural network via SGD, we focus on the three parameters learning rate (10 values), learning rate decay (5 values) and momentum (2 values). The cross-product of candidates for all the parameters results in a total number of 100 different hyperparameters. An overview of the individual candidates can be seen in Table 3. Smaller learning rates are chosen on purpose to challenge DP-Hype with a lot of bad hyperparameters that attain model accuracy at best as good as guessing. The remaining learning rates in combination with the other parameters lead to a few excellent hyperparameters and a medium-sized number of mediocre performing hyperparameters. We explicitly avoid adopting the same setup as prior work, because oftentimes the hyperparameter set consists of only a few candidates in total. On the one hand, this is a priori beneficial for privacy-preserving algorithms as the number of potential compositions remains low and on the other hand, a large fraction of good candidates makes randomly guessing the right hyperparameter a viable strategy.

Evaluated Algorithms. As a first non-private baseline, which we call Opt, we investigate the accuracy of the best-performing hyperparameter. For every hyperparameter candidate we train a model using federated learning and then select the best-performing hyperparameter as the optimal baseline. As another baseline and for lower bounding expected results of our DP-Hype, we also report the average of accuracies over all candidates of hyperparameters which we call RandGuess. This average represents the expected accuracy one would get from randomly selecting a hyperparameter, a strategy that does not have any privacy leakage since it is independent of sensitive data. We then present all candidates to DP-Hype and allow it to find a good compromise across all clients in a federated manner. The clients' local loss function is set to the standard accuracy metric of their model. As for the maximum number of local votes, we set it to $k = 5$ as we observe that it works well across all data sets, so it does not need to be tuned based on the specific data set. So every client votes for the five hyperparameter candidates attaining the highest accuracy on their local model. With the hyperparameter that attains the most noisy votes, we then train the global model in a federated manner as well to assess the performance of the hyperparameter choice of DP-Hype. For performance metrics, we report the average test accuracy and the corresponding 95% confidence interval resulting from 20 runs. Every federated training is based on FedAvg [29] and the number of local as well as federated training runs is set to 5 with a batch size of 64. Except for Cifar-10 where 10 epochs are performed locally and globally. We specifically avoid private training, using DP-SGD [1] for instance, to evaluate the performance of hyperparameters, as this would add much more noise to the training pipeline and introduce additional bias to the results. Also, this allows us to better control a priori that a certain number of hyperparameters results in bad model performance to further challenge our algorithm.

Concerning a comparison with related work, the privacy-preserving federated hyperparameter tuning that comes closest to our work is Feathers [36]. Feathers selects hyperparameters based on loss differences between hyperparameter candidates. A closer inspection reveals that Feathers does not preserve differential privacy. While it adds noise to the loss values, it does so without applying clipping, which is necessary when dealing with unbounded quantities. Moreover, Feathers performs multiple iterations that are not included in its privacy accounting. Accounting for all iterations severely increases the leakage, making it impossible for Feathers to maintain competitive utility under strict privacy budgets. We compare DP-Hype against a version of Feathers where we corrected the privacy accounting in Appendix G.

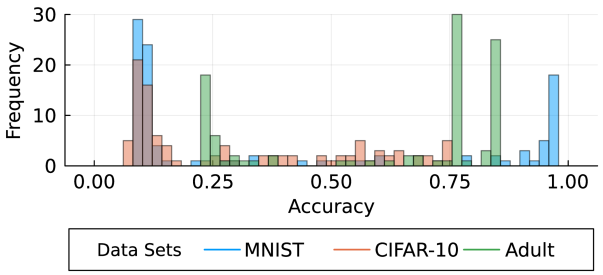


Figure 3: Histogram over individual accuracy values when training a global model on each data set using SGD on the set of hyperparameters shown in Table 3. As expected, the accuracies vary drastically depending on the data set.

6.3 Privacy-Utility Trade-Off

To capture the privacy-utility trade-off of DP-Hype, we evaluate the performance in iid as well as non-iid scenarios of RandGuess, Opt, and DP-Hype on three benchmark data sets: MNIST, Cifar-10, and Adult. For the number of clients, we consider $n \in \{50, 100, 250\}$. For privacy, we fix $\delta = 10^{-5}$ while varying the privacy budget $\varepsilon \in \{0.1, 0.25, 0.5, 1, 3, \infty\}$ resulting in noise scales for the Gaussian mechanism $\sigma^2 \in \{103, 46, 24, 12.5, 4.7, 0\}$, respectively.

In the iid scenario, each client is assumed to have local training data sampled from the same underlying distribution. To construct this setup, we randomly divide the training data into equal portions such that every client receives the same amount of data and the distribution of labels remains nearly identical across clients. The results are summarized in Figure 4. For non-iid scenarios, clients differ both in the size of their local training data and in the distribution of labels. We simulate such scenarios using a Dirichlet partitioning approach. More specifically, for each label l in the data set, we draw proportions from an n-dimensional Dirichlet distribution with concentration parameter β, which determines how samples with label l are split across clients. Usually, the letter α is used in the literature for non-iid scenarios, but we use β to avoid a conflict with α from Rényi DP. Once these proportions are fixed, the data set is shuffled and federated accordingly. Smaller values of β result in highly skewed distributions, potentially assigning nearly all samples of a given label to a single client, while larger values of β yield more iid-like partitions. This setup aligns with prior work and effectively combines two common non-iid conditions: label skewness and variation in client data sizes. For our experiments, we investigate $\beta \in \{30.0, 5.0, 0.5\}$. The results are in Figure 5 as well as in Appendix E. We visualize label distribution and size of clients' local data sets in Appendix C. In this setting, all models are trained globally for 5 training runs.

IID Scenario. First, we take a closer look at Figure 4. The results highlight two key effects: the positive impact of increasing the number of clients and the inherent privacy-utility trade-off. The accuracy improvement due to more clients can be explained by our utility analysis through the parameter γ, which measures the gap in votes between good and bad hyperparameters. In an iid setting, more clients amplify the votes for strong hyperparameters, making it harder for noise to distort the selection process. This effect is especially visible in the MNIST and Cifar-10 results, while on Adult, DP-Hype occasionally favors second-best hyperparameters, though the trend remains consistent. As expected, decreasing the privacy budget lowers utility, but this loss can be partially compensated by increasing the number of clients, since more clients produce more votes and thus improve the signal-to-noise ratio.

Table 4: Resource consumption of DP-Hype for the local hyperparameter evaluation and the federated voting. The former is measured per client individually, and captures the GPU vRAM overhead in MB and the runtime (RT) in seconds for each data set. The voting part shows the individual up- and download bandwidth in KB and the runtime in seconds including all clients. All measurements are averaged over five runs with $n \in \{50, 100, 250\}$.

	Local HP Evaluation			Federated Voting	
n	Data set	vRAM	RT	Bandwidth	RT
50	MNIST	68.29	120.22	33.78 / 29.72	16.66
	Cifar-10	199.44	131.36		
	Adult	18.29	50.52		
100	MNIST	68.29	63.78	61.02 / 49.36	28.36
	Cifar-10	199.44	71.78		
	Adult	18.29	27.05		
250	MNIST	68.29	26.53	147.30 / 111.38	193.13
	Cifar-10	199.44	29.51		
	Adult	18.31	12.74		

Non-IID. When comparing Figures 5, 12 and 13, increasing the number of clients reduces the number of data points available per client, which explains why the utility of all algorithms decreases. However, it also reduces the noise contributed per client, so DP-Hype benefits from larger client populations. A higher degree of non-iidness further decreases utility overall and also changes the privacy–utility trade-off of DP-Hype, because votes are spread across more hyperparameter candidates, diluting the evidence for any single choice. It is therefore natural that DP-Hype cannot consistently match the optimal utility, as purely local evaluations cannot fully replace global evaluation. In an iid setting, FedAvg performs nearly as well as centralized training on the pooled data, whereas DP-Hype may miss globally best candidates because they might not materialize in any client's local evaluation and thus cannot be selected. In the non-iid setting, FedAvg no longer maintains the same utility and, in addition, higher variance across clients produces more strong candidates. Overall, these results indicate that trading global hyperparameter evaluation for local evaluation yields a substantial privacy improvement while sacrificing some optimality (which also carries over to utility), and that DP-Hype remains effective under smaller privacy budgets than typically required in privacy-preserving federated learning tasks.

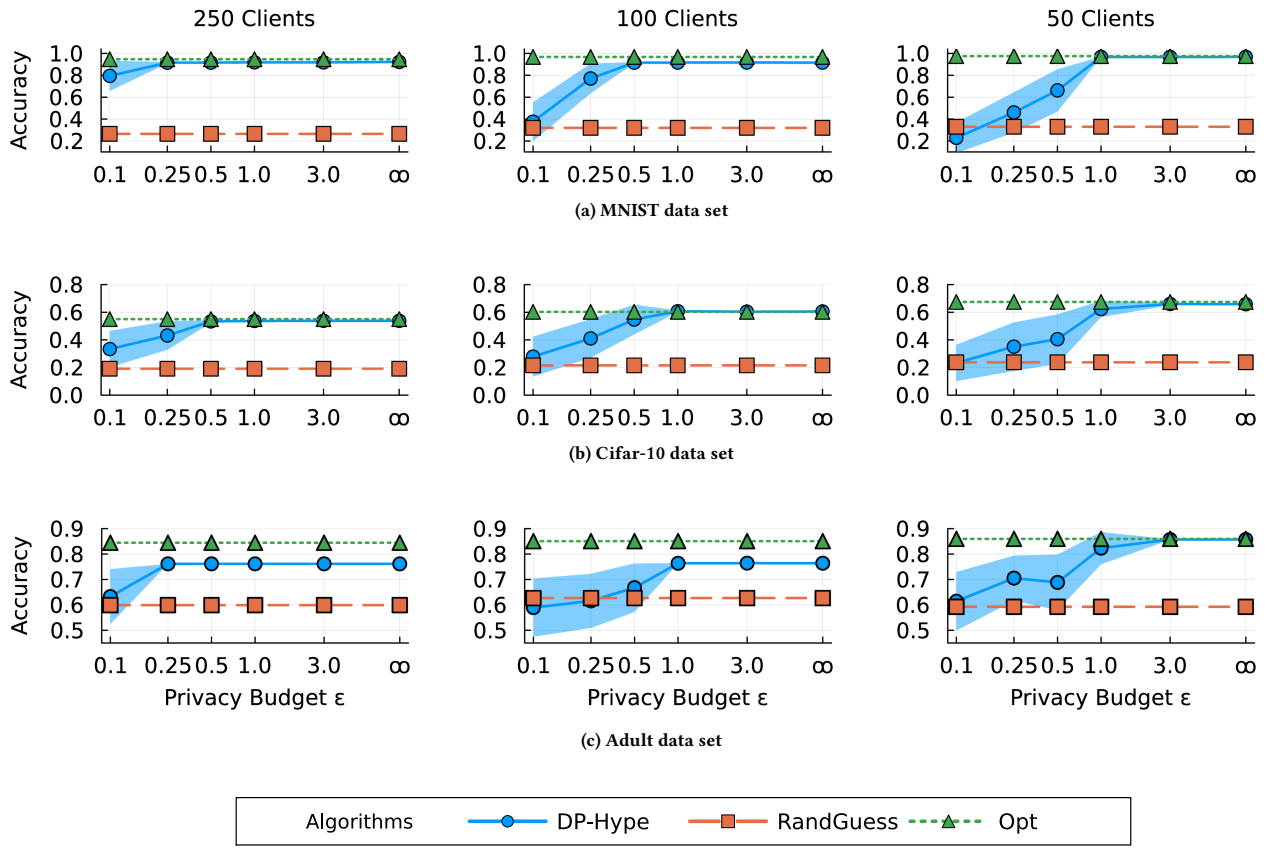


Figure 4: Empirical results in terms of Accuracy for evaluating DP-Hype, RandGuess and Opt on the data sets MNIST, Cifar-10 and Adult for a varying number of clients and different privacy budgets. It can be seen that the set of hyperparameters contains a lot of bad candidates as RandGuess is far away from Opt. Yet, DP-Hype is very close to the best possible result even for small privacy budgets and this effect gets even stronger with a growing number of clients.

6.4 Resource Consumption

We also evaluate the resource consumption of DP-Hype in terms of GPU memory footprint, bandwidth and runtime. Overall, the computational cost decomposes naturally into two phases: (i) local training and evaluation performed independently by each client for every considered hyperparameter, and (ii) a federated, privacy-preserving voting step, which requires a single invocation of a secure summation protocol.

For the local hyperparameter search, we report the per-client GPU memory consumption (in MB) by using the PyTorch built-in function *torch.cuda.max_memory_allocated*, and the wall-clock runtime (in seconds). For the voting phase of DP-Hype, we use the Flower implementation of SecAgg++ [8]. The protocol exposes two main parameters: the number of shares (i.e., the number of secret shares created per client) and the reconstruction threshold, which specifies how many shares are required to reconstruct an encoded value. We set the number of shares to $0.3n$, which provides reasonable security. While a logarithmic number of shares would suffice in principle, for $n < 1000$ this would amount to only a few shares. Since we do not consider client dropouts in this setting, we set the reconstruction threshold equal to the number of shares (dropout-tolerance is still possible with DP-Hype, but is not evaluated here). We fix the voting vector size to 100, consistent with the other experiments, and measure per-client upload and download traffic (in KB) as well as the overall runtime (in seconds). These experiments are conducted five times in the iid setting for $n \in \{50, 100, 250\}$ across all three data sets, and the results are shown in Table 4.

We observe that both GPU RAM consumption and runtime are smallest for Adult, followed by MNIST, and highest for Cifar-10. This trend is expected, as both model complexity and input dimensionality increase in this order. Moreover, as the number of clients increases, the GPU RAM consumption remains constant while the overall runtime decreases substantially. The former follows directly because model complexity and batch size do not depend on n. The latter is explained by the fact that data points are federated across clients. Hence, more clients imply fewer local training examples per client and, consequently, shorter local training times. In all configurations, the per-client GPU memory usage remains in the

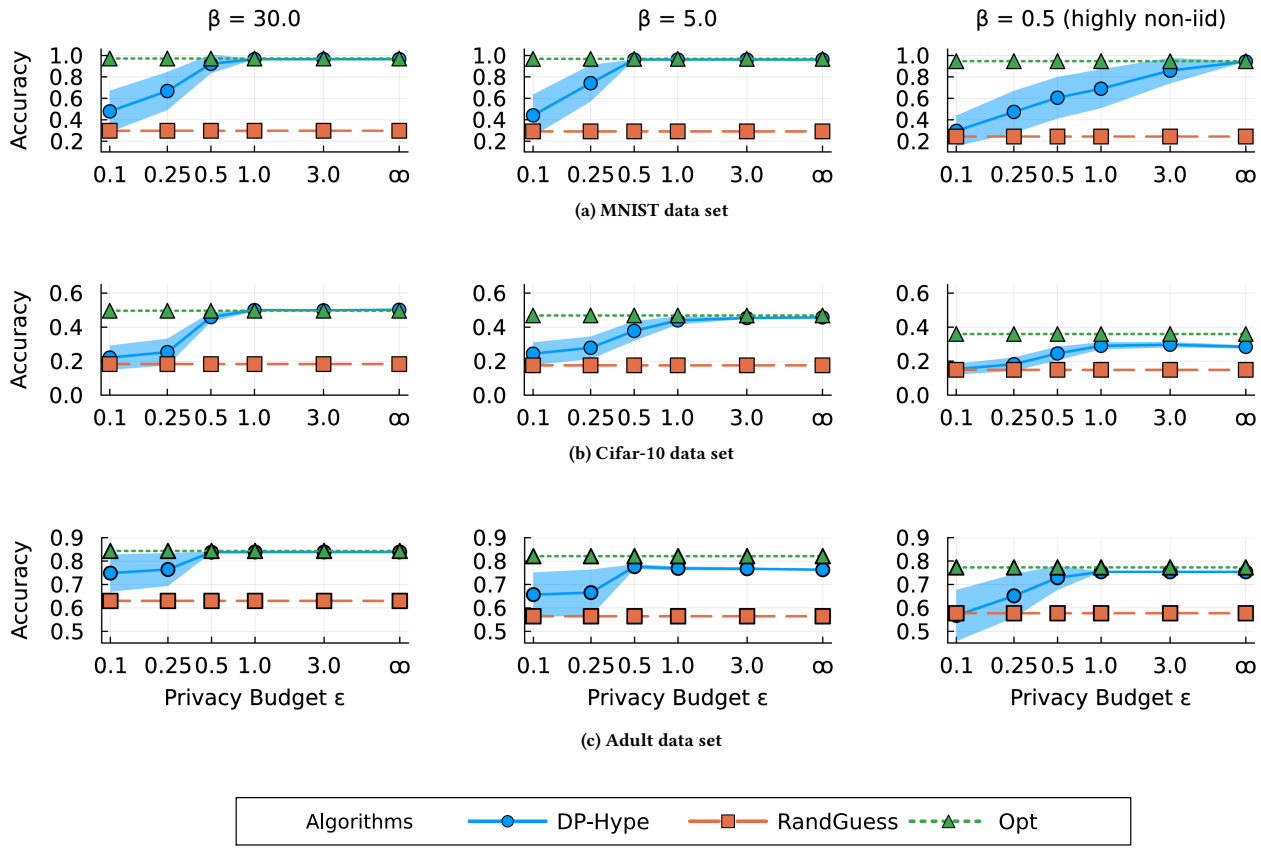


Figure 5: Empirical results in terms of Accuracy for evaluating DP-Hype, RandGuess and Opt on the data sets MNIST, Cifar-10 and Adult for $n = 100$ and different privacy budgets in the non-iid scenario. The degree of non-iid, denoted as β, increases from left to right, which means that the data heterogeneity increases as well. On MNIST, DP-Hype again reaches peak utility, which decreases slightly for the harder $\beta = 0.5$ scenario. On Cifar-10 the effect of non-iid scenarios is clear, as the utility of Opt shows. Still, DP-Hype can uphold a high degree of utility even for those scenarios.

MB range and the end-to-end runtime stays within a few minutes. The measurements of the federated voting are independent of the data set, because the size of the voting vector does not change. As expected, SecAgg++ introduces a clear overhead. Nevertheless, the bandwidth overhead per client remains within KB, and the runtime increases only to slightly above three minutes even for $n = 250$. This efficiency stems from the fact that the voting phase communicates only a single vector per client.

7 Related Work

Centralized DP Hyperparameter Search. In the central setting, the trusted aggregator has access to the entire data set and the adversary can only observe the final output of an algorithm. The naive approach would be to pay some part of the privacy budget for each hyperparameter separately. The individual privacy budget can be calculated based on composition theorems. Although these theorems are becoming increasingly tight, the individual privacy budget still decreases roughly in $\sqrt{r}$ with r compositions [31], which is just too small for most privacy-preserving federated learning algorithms [30]. There is a line of work that shows how hyperparameter search can be done more efficiently in DP. It starts with the sparse vector technique [19] that answers queries simply by outputting whether a noisy threshold exceeds the noisy query answer without any influence of the privacy budget on the number of queries. This is extended by [28] and the follow-up works [24, 33], which show that the search for hyperparameters, with and without a given threshold for utility, can be performed by only paying twice or three times the privacy budget as for a single hyperparameter. However, these techniques cannot be directly applied in a federated scenario. This is because the main privacy amplification comes from random stopping and strict assumptions, such as the adversary can only observe the final output and not intermediate results, which requires a trusted third party.

Federated DP Hyperparameter Search. Conducting a hyperparameter search on sensitive data in the federated setting is far less studied in contrast to the central setting, as privacy-preserving

federated learning by itself is a difficult task. Some solutions like LDP-DSBO [15] are specialized on specific learning algorithms, which leverage some properties of gradients used to train models. Others like PrivTuna [32] focus on simulating the central setting with SMPC by paying the price of high communication overhead that only works with a few clients without real privacy protection as the results are just published without noise. When it comes to a federated privacy-preserving hyperparameter search in general, there exist essentially two algorithms. First, DP-FTS-DE [16] for privacy-preserving federated Bayesian optimization and second, Feathers [36] combining neural architecture and hyperparameter search based on loss differences between hyperparameters. However, DP-FTS-DE tackles a different problem than DP-Hype, as it aims to help clients with the convergence of local hyperparameters. Moreover, DP-FTS-DE relies on a trusted third party, and there is no guarantee that the hyperparameters of one client work for another client. Additionally, the privacy analysis does not account for the hyperparameters to be published. Feathers, on the other hand, comes closest to our approach. However, Feathers fails to preserve DP, because it uses the cross-entropy loss with additive noise without any clipping, even though this loss function is unbounded and could theoretically be drastically influenced when exchanging data of a single client. Furthermore, Feathers does not take into account that each hyperparameter that is evaluated increases the leakage. In addition, Feathers does not perform well under realistic assumptions, i.e., small privacy budgets and a set of hyperparameters containing a lot of bad-performing candidates [36].

We intentionally focus on classic hyperparameter search, in contrast to Feathers, where a differentiable neural architecture search (NAS) is additionally conducted. NAS is typically studied as a distinct problem setting due to its discrete and structured search space, specialized algorithms, and substantially different compute–accuracy trade-offs [9, 38]. DP-Hype can be extended to NAS treating architectural choices as hyperparameters or additionally conducting differentiable NAS.

Differentially Private Voting. In the absence of a trusted aggregator or a secure aggregation protocol, local differential privacy (LDP) serves as the standard model to realize voting, requiring each client to enforce privacy individually. Several algorithms build on this notion: [18] evaluate binary voting with well-known mechanisms such as randomized response and RAPPOR; [41] proposes a multi-stage protocol based on randomized response; and [40] applies local votes for labeling unlabeled data. Despite their practicality, LDP mechanisms generally suffer from high noise: they require large client populations to average it out, and small privacy budgets can significantly reduce utility. Similar to DP-Hype, other work employs secure aggregation as well to simulate the central model. [44] introduce noiseless voting for heavy hitters, [2] construct global histograms to identify frequently visited locations, and [45] leverage differentially private voting to label unlabeled data points based on local contributions. On the other hand, DP-Hype extends this voting approach with a top-k mechanism, translates it towards a differentially private hyperparameter search and requires only a single invocation of the secure summation protocol.

8 Discussion

In a federated setting where each party has a list of items, *heavy hitter protocols* have the goal of selecting those items that occur most often. We could in principle deploy heavy hitter protocols to identify the most frequently chosen hyperparameters. Below, we discuss why we decided against deploying existing heavy hitter protocols. In Appendix A, we go into detail about the robustness of DP-Hype and how malicious parties can be handled.

We decided against local DP heavy hitter protocols [6, 13, 14, 27, 37, 42], to achieve a strong privacy-utility trade-off. LDP protocols do not rely on SMPC, but make the output of each client differentially private. Prior work [4] shows that they have an additional factor of $\sqrt{n}$ privacy leakage in terms of ε. In the shuffle model [4], this overhead can be reduced, but it assumes a mixing protocol, which again introduces trust assumptions that impede deployment.

There is a rich body of literature about SMPC-based heavy hitter protocols that rely on a small number (typically 2 or 3) of computation parties [5, 11, 12, 17] or SMPC-based private selection protocols [17]. As these protocols become very inefficient if the number of computation parties increases, this number has to be small in practice. In a multilateral deployment scenario with more than 10 parties, agreeing on how to set up a handful of computation servers can become a point of friction that significantly delays deployment. In addition, usually at least one of the computation parties has to be a trusted third party. So, we decided against relying on computation parties and designed DP-Hype such that it solely relies on secure summation, as this is a much simpler requirement, for which scalable protocols exist. However, the modularity of DP-Hype allows replacing the secure aggregation part easily if the use of one of the above protocols is required.

9 Conclusion

In this work, we introduced DP-Hype, a privacy-preserving algorithm to perform a federated hyperparameter search. DP-Hype is designed to find a compromise among clients that minimizes the loss of a learning task locally as much as possible. The core idea is to evaluate hyperparameters locally on each client's private data and let each client be part of a federated voting protocol with k votes per client. Then, the hyperparameter with the most votes is chosen for the learning task. The aggregation of local votes is implemented using a secure summation protocol such that the server only sees the aggregated votes per hyperparameter. Each client adds a small amount of noise to their private voting vector locally such that, in sum, the noise suffices to preserve differential privacy.

Compared to previous work, DP-Hype is agnostic to the specific learning task, preserves the strong notion of client-level differential privacy and its privacy guarantees are independent of the total number of hyperparameters. In addition, we formally quantify the probability that DP-Hype outputs a good hyperparameter. This probability converges exponentially to 1 with a growing gap between votes for good and for bad hyperparameters. We implemented DP-Hype in the popular federated learning framework Flower and evaluated it on three benchmark data sets. The results from a simulation and on the data sets demonstrate the high degree of utility that DP-Hype is able to uphold even for small privacy budgets, and in iid as well as non-iid scenarios.

Acknowledgments

This research has been conducted within the AnoMed project (https://anomed.de/) funded by the BMFTR (German Bundesministerium für Forschung, Technologie und Raumfahrt) and by the European Union – NextGenerationEU.

Throughout this work, we used generative AI-based tools to revise the text, improve flow and correct any typos, grammatical errors, and awkward phrasing. These tools were only used for small editorial purposes, and we inspected all outputs to ensure accuracy and originality. Importantly, all content is our own work and is not based on AI tools.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.

[2] Eugene Bagdasaryan, Peter Kairouz, Stefan Mellem, Adrià Gascón, Kallista Bonawitz, Deborah Estrin, and Marco Gruteser. 2021. Towards Sparse Federated Analytics: Location Heatmaps under Distributed Differential Privacy with Secure Aggregation. *arXiv preprint arXiv:2111.02356* (2021).

[3] Borja Balle, Gilles Barthe, Marco Gaboardi, Justin Hsu, and Tetsuya Sato. 2020. Hypothesis Testing Interpretations and Renyi Differential Privacy. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2496–2506.

[4] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. 2019. The Privacy Blanket of the Shuffle Model. In *Advances in Cryptology – CRYPTO 2019*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer International Publishing, 638–667.

[5] Borja Balle, James Bell-Clark, Albert Cheu, Adria Gascon, Jonathan Katz, Mariana Raykova, Phillipp Schoppmann, and Thomas Steinke. 2025. Hash-Prune-Invert: Improved Differentially Private Heavy-Hitter Detection in the Two-Server Model . In *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2903–2918. https://doi.org/10.1109/SP61157.2025.00186

[6] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Thakurta. 2017. Practical locally private heavy hitters. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., 2285–2293.

[7] Barry Becker and Ronny Kohavi. 1996. Adult. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20.

[8] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. 2020. Secure Single-Server Aggregation with (Poly)Logarithmic Overhead. In *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*. ACM, 1253–1269.

[9] Hadjer Benmeziane, Imane Hamzaoui, Zayneb Cherif, and Kaoutar El Maghraoui. 2024. Medical neural architecture search: Survey and taxonomy. In *International joint conference on artificial intelligence*.

[10] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Hei Li Kwing, Titouan Parcollet, Pedro PB de Gusmão, and Nicholas D Lane. 2020. Flower: A Friendly Federated Learning Research Framework. *arXiv preprint arXiv:2007.14390* (2020).

[11] Jonas Böhler and Florian Kerschbaum. 2021. Secure Multi-party Computation of Differentially Private Heavy Hitters. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, New York, NY, USA, 2361–2377. https://doi.org/10.1145/3460120.3484557

[12] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. 2021. Lightweight Techniques for Private Heavy Hitters. In *2021 IEEE Symposium on Security and Privacy (SP)*. 762–776. https://doi.org/10.1109/SP40001.2021.00048

[13] Mark Bun, Jelani Nelson, and Uri Stemmer. 2019. Heavy Hitters and the Structure of Local Privacy. *ACM Trans. Algorithms* 15, 4, Article 51 (Oct. 2019), 40 pages. https://doi.org/10.1145/3344722

[14] Karan Chadha, Junye Chen, John Duchi, Vitaly Feldman, Hanieh Hashemi, Omid Javidbakht, Audra McMillan, and Kunal Talwar. 2024. Differentially Private Heavy Hitter Detection using Federated Analytics. In *2024 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*. 512–533. https://doi.org/10.1109/SaTML59370.2024.00032

[15] Ziqin Chen and Yongqiang Wang. 2024. Locally Differentially Private Decentralized Stochastic Bilevel Optimization with Guaranteed Convergence Accuracy. In *Forty-first International Conference on Machine Learning*.

[16] Zhongxiang Dai, Bryan Kian Hsiang Low, and Patrick Jaillet. 2021. Differentially Private Federated Bayesian Optimization with Distributed Exploration. *Advances in Neural Information Processing Systems* 34 (2021), 9125–9139.

[17] Ivan Damgård, Hannah Keller, Boel Nelson, Claudio Orlandi, and Rasmus Pagh. 2024. Differentially Private Selection from Secure Distributed Computing. In *Proceedings of the ACM Web Conference 2024* (Singapore, Singapore) *(WWW '24)*. Association for Computing Machinery, New York, NY, USA, 1103–1114. https://doi.org/10.1145/3589334.3645435

[18] Bernardo David, Rosario Giustolisi, Victor Mortensen, and Morten Pedersen. 2023. Local Differential Privacy in Voting.. In *ITASEC*.

[19] Cynthia Dwork. 2006. Differential Privacy. In *International colloquium on automata, languages, and programming*. Springer, 1–12.

[20] Idoia Gamiz, Cristina Regueiro, Oscar Lage, Eduardo Jacob, and Jasone Astorga. 2025. Challenges and Future Research Directions in Secure Multi-Party Computation for Resource-Constrained Devices and Large-Scale Computations. *International Journal of Information Security* 24, 1 (2025), 1–29.

[21] Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially Private Federated Learning: A Client Level Perspective. *arXiv preprint arXiv:1712.07557* (2017).

[22] Changhui Hu, Jin Li, Zheli Liu, Xiaojie Guo, Yu Wei, Xuan Guang, Grigorios Loukides, and Changyu Dong. 2021. How to Make Private Distributed Cardinality Estimation Practical, and Get Differential Privacy for Free. In *30th USENIX security symposium (USENIX Security 21)*. 965–982.

[23] Peter Kairouz, Ziyu Liu, and Thomas Steinke. 2021. The Distributed Discrete Gaussian Mechanism for Federated Learning with Secure Aggregation. In *International Conference on Machine Learning*. PMLR, 5201–5212.

[24] Antti Koskela and Tejas D Kulkarni. 2023. Practical Differentially Private Hyperparameter Tuning with Subsampling. *Advances in Neural Information Processing Systems* 36 (2023), 28201–28225.

[25] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning Multiple Layers of Features from Tiny Images. (2009).

[26] Yann LeCun, Corinna Cortes, and CJ Burges. 2010. MNIST Handwritten Digit Database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

[27] Xiaochen Li, Weiran Liu, Jian Lou, Yuan Hong, Lei Zhang, Zhan Qin, and Kui Ren. 2024. Local Differentially Private Heavy Hitter Detection in Data Streams with Bounded Memory. *Proc. ACM Manag. Data* 2, 1, Article 30 (March 2024), 27 pages. https://doi.org/10.1145/3639285

[28] Jingcheng Liu and Kunal Talwar. 2019. Private Selection from Private Candidates. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 298–309.

[29] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Jerry Zhu (Eds.). PMLR, 1273–1282. https://proceedings.mlr.press/v54/mcmahan17a.html

[30] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017. Learning Differentially Private Recurrent Language Models. *arXiv preprint arXiv:1710.06963* (2017).

[31] Ilya Mironov. 2017. Rényi Differential Privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*. IEEE, 263–275.

[32] Natalija Mitic, Apostolos Pyrgelis, and Sinem Sav. 2025. Privacy-Preserving Hyperparameter Tuning for Federated Learning. *IEEE Transactions on Privacy* (2025).

[33] Nicolas Papernot and Thomas Steinke. 2021. Hyperparameter Tuning with Renyi Differential Privacy. *arXiv preprint arXiv:2110.03620* (2021).

[34] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. 2020. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295* (2020).

[35] César Sabater, Aurélien Bellet, and Jan Ramon. 2022. An accurate, scalable and verifiable protocol for federated differentially private averaging. *Mach. Learn.* 111, 11 (Nov. 2022), 4249–4293. https://doi.org/10.1007/s10994-022-06267-9

[36] Jonas Seng, Pooja Prasad, Martin Mundt, Devendra Singh Dhami, and Kristian Kersting. 2022. Feathers: Federated Architecture and Hyperparameter Search. *arXiv preprint arXiv:2206.12342* (2022).

[37] Tianhao Wang, Ninghui Li, and Somesh Jha. 2021. Locally Differentially Private Heavy Hitter Identification . *IEEE Transactions on Dependable and Secure Computing* 18, 02 (2021), 982–993. https://doi.org/10.1109/TDSC.2019.2927695

[38] Xin Wang and Wenwu Zhu. 2024. Advances in neural architecture search. *National Science Review* 11, 8 (2024), nwae282.

[39] Felix Nikolaus Wirth, Tobias Kussel, Armin Müller, Kay Hamacher, and Fabian Prasser. 2022. EasySMPC: A Simple but Powerful No-Code Tool for Practical Secure Multiparty Computation. *BMC bioinformatics* 23, 1 (2022), 531.

[40] Yukai Xu, Jingfeng Zhang, and Yujie Gu. 2024. Privacy-Preserving Heterogeneous Federated Learning for Sensitive Healthcare Data. In *2024 IEEE Conference on Artificial Intelligence (CAI)*. IEEE, 1142–1147.

[41] Ziqi Yan, Jiqiang Liu, and Shaowu Liu. 2019. DPWeVote: Differentially Private Weighted Voting Protocol for Cloud-Based Decision-Making. *Enterprise Information Systems* 13, 2 (2019), 236–256.

[42] Yuemin Zhang, Qingqing Ye, and Haibo Hu. 2025. Federated Heavy Hitter Analytics with Local Differential Privacy. *Proc. ACM Manag. Data* 3, 1, Article 42

(Feb. 2025), 27 pages. https://doi.org/10.1145/3709739
[43] Ian Zhou, Farzad Tofigh, Massimo Piccardi, Mehran Abolhasan, Daniel Franklin, and Justin Lipman. 2024. Secure Multi-Party Computation for Machine Learning: A Survey. *IEEE Access* (2024).
[44] Wennan Zhu, Peter Kairouz, Brendan McMahan, Haicheng Sun, and Wei Li. 2020. Federated Heavy Hitters Discovery with Differential Privacy. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 3837–3847.
[45] Yuqing Zhu, Xiang Yu, Yi-Hsuan Tsai, Francesco Pittaluga, Masoud Faraki, Yu-Xiang Wang, et al. 2020. Voting-Based Approaches for Differentially Private Federated Learning. *arXiv preprint arXiv:2010.04851* (2020).

A Practical Considerations

In this section, we state the advantages of a majority vote when dealing with skewed data or dropouts. Afterward, we discuss how the utility and privacy of DP-Hype could be affected if the server or some clients are malicious and may deviate from the protocol.

A.1 Robustness

For practical deployment, it is important to consider how DP-Hype behaves under skewed data and client dropouts during a protocol run. Notably, DP-Hype can also operate in a fully federated setting without a central server, depending on the underlying secure summation protocol. Since the algorithm itself does not require central coordination, each client can independently process the aggregated result once the final sum is available, and any client could take over the aggregation task without additional computational cost. With respect to data distribution, the majority-based design of DP-Hype is both a strength and a limitation. On the one hand, the algorithm assumes that data is sufficiently similar among clients, ensuring that most clients reach comparable conclusions when evaluating hyperparameters. On the other hand, this same reliance on majority decisions makes DP-Hype robust to a small fraction of clients with highly skewed data, as their influence on the global outcome is limited.

A second challenge arises from client dropouts, a common issue in federated learning where clients may disconnect during the protocol due to failures or unstable connections. Whether dropouts can be tolerated without disrupting the aggregation depends on the secure summation protocol, and dropout resistance is an important feature of many of such protocols. In the case of DP-Hype, dropouts would normally lead to insufficiently noised voting vectors $\mathbf{v}$, undermining differential privacy guarantees. A key advantage of DP-Hype is that it can tolerate dropouts by design with only minimal adjustments. Specifically, if the algorithm is configured to tolerate up to a fraction ξ of dropouts, each client replaces the denominator n in the variance scaling of the Gaussian noise with $(1-\xi)\cdot n$. This ensures that even if up to $\xi \cdot n$ clients drop out, the aggregated vector still receives sufficient noise. While this approach slightly worsens the privacy–utility trade-off, it provides a practical safeguard against unpredictable client failures and has also been applied in related work.

A.2 Malicious Clients and Server

So far, we have assumed that all clients and the server are honest-but-curious, meaning they follow the protocol faithfully, but may still try to infer sensitive information from observed messages. A natural extension is to ask how DP-Hype behaves if some participants are malicious and actively deviate from the protocol. One straightforward attack against privacy would be for clients to avoid the addition of noise to their local voting vectors. This issue can be mitigated similarly to the dropout scenario by introducing additional noise to compensate for potentially malicious clients. In that case, even if a certain fraction of clients refuses to add noise, the remaining honest participants are sufficient to ensure differential privacy. Since DP-Hype can operate successfully under small privacy budgets, this approach remains feasible and the effect of individual malicious clients becomes negligible as the total number of participants grows. Moreover, clients cannot observe the votes of others by design, which prevents them from coordinating such an attack.

Beyond privacy, malicious clients may also attempt to sabotage utility by voting randomly or refusing to support good hyperparameters. Fortunately, voting mechanisms naturally provide a degree of Byzantine robustness: The impact of any single malicious client is bounded by the sensitivity of their local voting vector $\mathbf{v}_i$, preventing them from arbitrarily manipulating the result. Still, the use of secure summation means that individual votes are hidden, and thus protocol compliance cannot be verified directly. Consequently, the impact of a malicious client's local voting vector is potentially unbounded. Fortunately, we can make use of the modularity of DP-Hype and its independence of specific secure aggregation protocols. By using GOPA [35], a fully federated secure aggregation protocol, input poisoning can be mitigated as it supports input validation mechanisms in a scalable way.

Finally, if the server itself is malicious, it could attempt to influence the protocol outcome by misreporting the result of the secure summation. This type of manipulation affects only utility, not privacy, as noise is applied locally on the client side. The only way for a malicious server to compromise privacy would be by tampering with the secure summation protocol to reveal individual votes. However, many secure aggregation schemes are designed to tolerate a fraction of malicious clients and even adversarial servers, ensuring that privacy guarantees remain intact under realistic threat models.

B Model Architectures

To perform our evaluation, we use the following NN architectures:

MNIST. It begins with two convolutional layers featuring 32 and 64 filters, respectively, each using a 3×3 kernel. This is followed by a 2d max-pooling layer of size 2. The classifier head consists of two dense layers with a hidden size of 128. The first dense layer utilizes ReLU activation and is preceded by a dropout layer with a rate of 0.25. An additional dropout layer with a rate of 0.5 is applied before the final output, which serves as input for the cross entropy loss.

Cifar-10. The architecture consists of two sequential convolutional blocks followed by a global average pooling layer and a final dense layer. The first block contains two convolutional layers with 32 filters, each followed by batch normalization and ReLU activation, ending with a max-pooling layer and a dropout rate of 0.25. The second block follows a similar structure but increases the filter count to 64 and 128, respectively, also concluding with max-pooling and 0.25 dropout. A fully connected layer produces the output, which serves as input for the cross entropy loss.

Adult. For this data set, an MLP is employed, consisting of two hidden layers with 64 units each. Both layers utilize ReLU activation functions. To mitigate overfitting, a dropout layer with a rate of 0.2 is applied after the second hidden layer. The resulting raw logits are passed directly to a binary cross-entropy loss function.

C Visualization of Non-IID Clients

To better understand how tough the non-iid learning scenarios are and how much the scenarios we evaluate differ from one another, we visualize the label distribution and size of clients' local data sets. In the non-iid scenarios, the MNIST, Cifar-10 and Adult data sets are divided among clients using a Dirichlet partitioning approach. More specifically, for each label l in the data set, we draw proportions from an n-dimensional Dirichlet distribution with concentration parameter β, which determines how samples with label l are split across clients. Once these proportions are fixed, the data set is shuffled and federated accordingly. When setting the parameter β of the Dirichlet distribution to 30.0, the data set is split more iid than non-iid. For smaller values of β however, the label distribution is quite skewed over the clients and the size of their local data sets can vary a lot, see Figure 6 and Figure 7.

D Non-IID Accuracy Distributions

To provide further insights into the underlying structure of the hyperparameter search, we present the accuracy distributions for $\beta \in \{30.0, 5.0, 0.5\}$ and $n \in \{50, 100, 250\}$ in Figure 8. For each hyperparameter, a federated model was trained to obtain 100 accuracy values. Interestingly, the accuracy distributions do not differ much from the global evaluation reported in Figure 3. There are mainly two side effects that can be observed. First, for an increasing β (top to bottom) the distributions shift slightly to the left. In other words, the maximum possible accuracy decreases for harder non-iid scenarios, which can also be observed in the reported results. Second, for a growing number of clients (right to left) the accuracy again decreases but this time only a bit, which is also visible in the results. Both side effects intuitively make sense, because both make the learning task harder, either by amplifying the non-iid degree or by reducing the number of data points per client.

E Additional Non-IID Results

In this section, we present the results for the non-iid scenario that were moved here due to space limitations. The experimental setup is the same as in Section 6.3 with $n = 250$ in Figure 12 and $n = 50$ in Figure 13. As a small side note, $\beta = 2$ on Adult for $n = 250$ in the scenario of $\beta = 0.5$, because this hard non-iid setting in combination with binary classification messes up the label and data distribution so much that no reasonable results could be obtained for Adult.

The additional results uncover some interesting effects in the non-iid scenario. The performance of Opt decreases further for $n = 250$, due to smaller individual data sets of clients, and the opposite for $n = 50$. As expected, DP-Hype benefits from a larger number of clients as this naturally increases the number of votes per hyperparameter, which makes it harder for the introduced noise to let a bad one win the voting.

F Raw Losses Instead of Binary Votes

In our current design, each client participates in the binary vote by assigning a value of one to the k best-performing hyperparameter candidates and zero to all remaining candidates. A natural alternative is to replace these binary indicators with real-valued scores derived from local performance, enabling a more fine-grained vote in which the globally best hyperparameter can, in principle, attain the true maximum. Using raw losses directly is inconvenient because lower loss corresponds to higher utility. Moreover, loss functions used in DP-Hype must be bounded to avoid unbounded sensitivity. One simple normalization is to subtract each local loss from the maximum possible loss so that higher values correspond to better utility. In our setting, we can instead use accuracy as the score, since it is naturally bounded in the interval $[0, 1]$ and therefore does not alter the sensitivity bound of the voting vector. We evaluate this strategy for $n = 100$ and $\beta \in \{0.5, 5.0\}$ on the Cifar-10 data set, using the same experimental setup as in Section 6.3.

The results are shown in Figure 9, where *Binary* denotes the original DP-Hype and *Raw Loss* denotes the proposed real-valued variant. Overall, the accuracy-based voting still selects reasonable hyperparameters, as its resulting accuracy remains relatively close to the binary baseline across settings. However, in both non-iid scenarios, a clear degradation is visible for $\varepsilon \leq 3$. We attribute this to the effective value range of accuracy: while it is bounded between zero and one, realistic accuracies rarely approach one, so the per-client signal injected into the vote is substantially smaller than the worst-case change assumed by the sensitivity analysis (which permits k entries to shift by one). Consequently, the signal-to-noise ratio deteriorates more quickly under privacy noise than in the binary scheme, amplifying the impact of noise and worsening the privacy-utility trade-off. Although a real-valued vote could still be beneficial in edge cases due to its finer granularity, our experiments indicate that it typically increases noise sensitivity and can significantly reduce utility at stricter privacy levels.

G Comparison with Feathers

We compare DP-Hype with the privacy-preserving federated hyperparameter tuning that comes closest to our work, Feathers [36]. Feathers keeps a distribution over the hyperparameter candidates and updates this distribution to favor the configurations that attain the greatest decrease in clients' losses. Hyperparameter optimization is alternated with training phases that use the configuration that is rated the highest in the distribution.

However, a closer inspection of Feathers reveals that it does not preserve differential privacy; therefore we correct their privacy accounting before our evaluation. First, we clip the client loss values at 1.0 to ensure bounded sensitivity of the losses which is necessary for guaranteeing DP. To calibrate the additive noise for the clipped loss values, we employ a Rényi DP accountant. We fix the number of DP approximated loss releases to 100 to have an a priori bound on the privacy loss, which is necessary for satisfying formal DP guarantees. The vanilla version of Feathers uses the entropy of the hyperparameter distribution to select how many DP approximated losses are released, which only yields an a posteriori bound on the privacy loss, after a run completes.

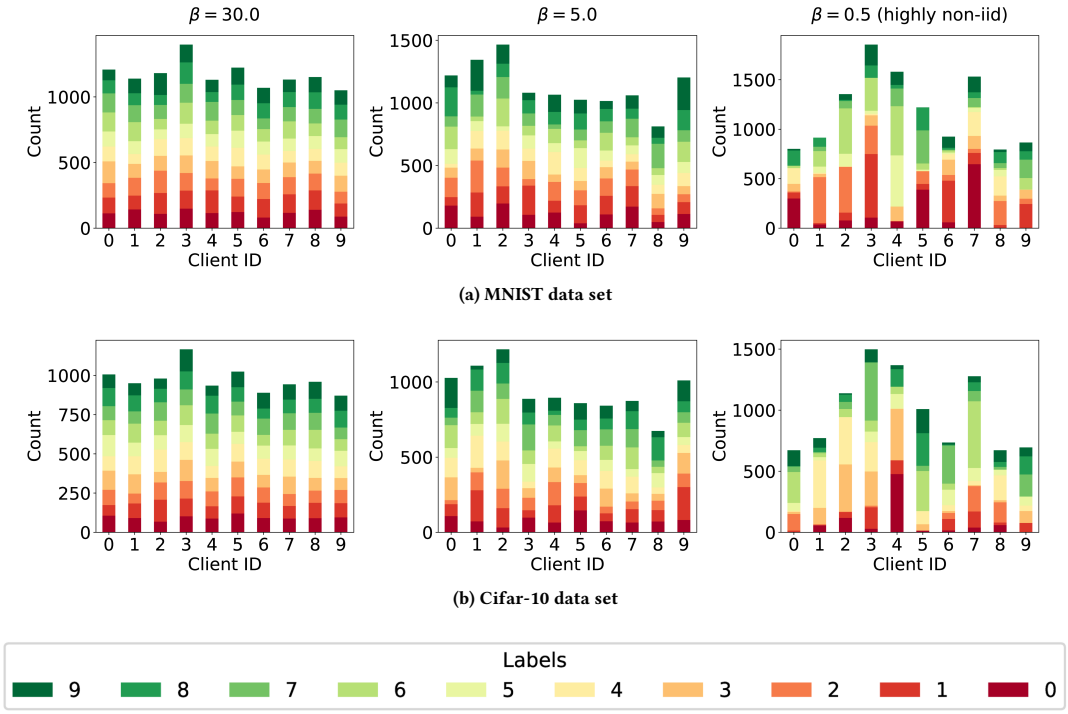


Figure 6: Visualization of label distribution and size of local data sets of 10 clients on the data sets MNIST and Cifar-10 for varying non-iid scenarios. The data sets are split among 50 clients using a Dirichlet partitioning approach with Dirichlet concentration parameter $\beta \in \{30.0, 5.0, 0.5\}$, only 10 clients are shown. The degree of non-iid β increases from left to right, which means that the data heterogeneity increases as well.

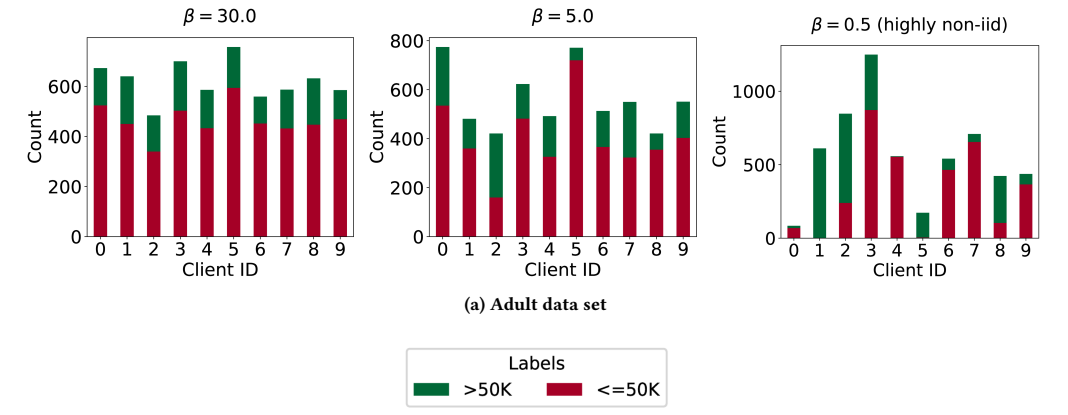


Figure 7: Visualization of label distribution and size of local data sets of 10 clients on the Adult data set for varying non-iid scenarios. The data set is split among 50 clients using a Dirichlet partitioning approach with Dirichlet concentration parameter $\beta \in \{30.0, 5.0, 0.5\}$, only 10 clients are shown. The degree of non-iid β increases from left to right, which means that the data heterogeneity increases as well.

In our evaluation, we compare DP-Hype against Feathers on MNIST and Cifar-10 in the iid scenario with 250 clients. For Feathers, we configure all additional parameters as proposed in the paper [36] and average the accuracy for 10 runs. The results are summarized in Figure 11. For MNIST and $\varepsilon \geq 1.0$, Feathers can still attain its high accuracy. However, for smaller privacy budgets Feathers' accuracy decreases (44.1% accuracy for $\varepsilon = 0.5$), while DP-Hype still attains the same high accuracy. This is as expected because

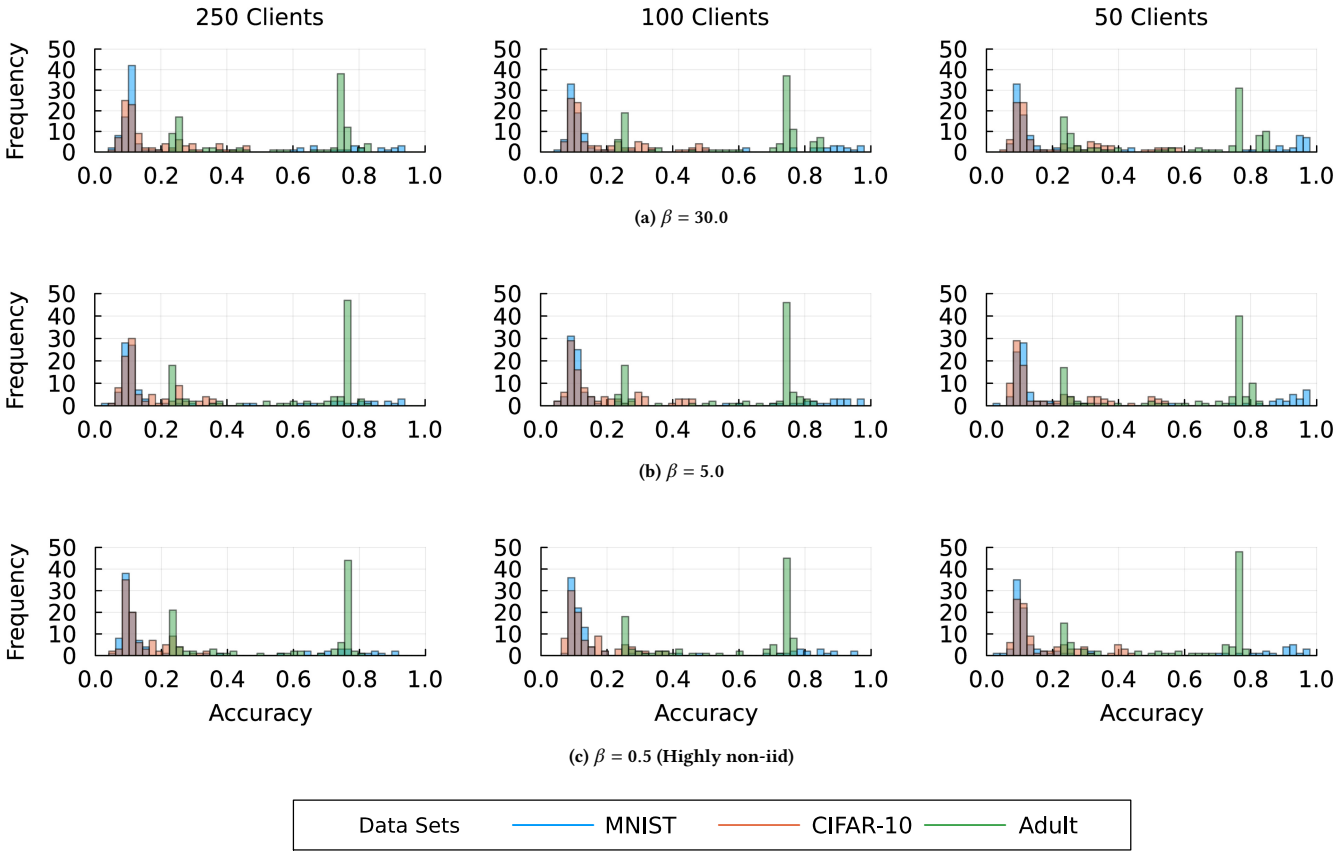


Figure 8: A histogram over individual accuracy values when using federated learning with $n \in \{50, 100, 250\}$. We train a model on each data set using SGD in different non-iid settings with Dirichlet concentration parameter $\beta \in \{30.0, 5.0, 0.5\}$ and $\beta = 2.0$ for $n = 250$ on Adult on the set of hyperparameters shown in Table 3.

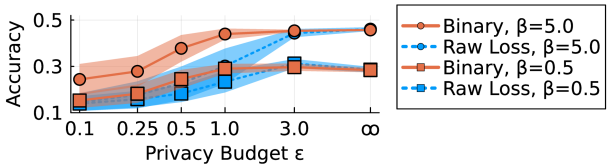


Figure 9: Comparison between default binary voting of DP-Hype and an alternative approach where the accuracy for k hyperparameters is reported directly instead of ones. The experiments are performed in Cifar-10 with $n = 100$. The utility of the alternative clearly degrades much faster for decreasing privacy budgets, which can be mainly attributed to a reduced signal-to-noise ratio as accuracy seldom reaches a value close to one.

Feathers uses a local DP approach which adds unnecessarily large amounts of noise, quickly yielding an almost random draw from the hyperparameter candidates. This is also apparent for Cifar-10, where the accuracy of Feathers begins to decrease at $\varepsilon = 3.0$ (38.9%). This matches our expectation: Figure 3 shows that for Cifar-10, fewer hyperparameter candidates perform well. Thus, Feathers' suboptimal signal-to-noise ratio likely leads it to select configurations that perform suboptimally.

H Parallelization Baseline

DP-Hype currently requires every client to train and evaluate a model for each hyperparameter candidate locally, which can be computationally expensive. To reduce this resource overhead, one can divide clients into random subsets and assign a single hyperparameter to each subset. The server then aggregates the reports by averaging the noisy losses per subset and selects the hyperparameter with the best average performance. This reduces the per-client workload to training just one model, but it also substantially lowers the signal-to-noise ratio for each hyperparameter, since each candidate is evaluated by only a fraction of the clients. From a privacy perspective, this strategy no longer involves a parameter k, and the sensitivity of the overall voting vector becomes $\sqrt{2}$ because

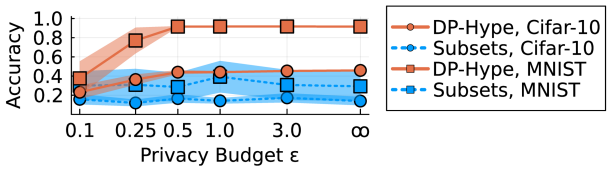


Figure 10: Comparison between DP-Hype and a simple parallelization strategy where clients are randomly assigned into subsets each evaluating a single hyperparameter. Per subset noisy losses are aggregated and the server selects the hyperparameter with optimal loss. Experiments are performed on MNIST and Cifar-10 for $n = 250, \beta = 30$ and $n = 100$, iid, respectively. DP-Hype clearly outperforms the alternative strategy. The main reason for this is that the per-hyperparameter signal is just too small to overcome even small noise, and that even for no privacy the variance per hyperparameter evaluation is too large to let the best candidates win reliably.

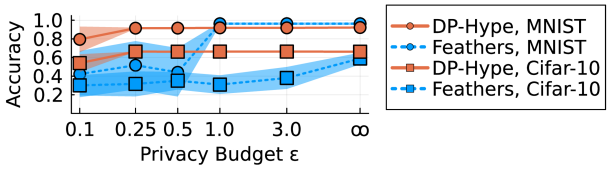


Figure 11: Empirical results in terms of Accuracy for evaluating DP-Hype and Feathers [36] on the data sets MNIST and Cifar-10 with $n = 250$ clients and different privacy budgets. DP-Hype clearly attains a superior privacy-utility trade-off compared to Feathers. Feathers often selects hyperparameters almost randomly, yielding suboptimal accuracy.

replacing the data of a single client can affect at most two entries by 1.

Results for DP-Hype and this alternative, which we denote by *Subsets*, are shown in Figure 10. Across both scenarios, DP-Hype clearly outperforms *Subsets*. Even in the MNIST iid setting, which we included to make the task as favorable as possible for *Subsets*, the method does not exceed performance comparable to guessing. We attribute this to two factors. First, the poor performance even in the non-private regime indicates that evaluating each hyperparameter on only a small number of clients (e.g., two- or three-fold for $n = 250$) does not yield a reliable signal: even strong hyperparameters tend to perform well for a majority of clients rather than uniformly for every client, and this majority effect is suppressed when evaluations are highly fragmented. Second, the signal-to-noise ratio is inherently smaller than in DP-Hype, and this disadvantage compounds under privacy noise. Indeed, as observed in Appendix F, using plain accuracy already provides a weak signal, and reducing the number of clients per hyperparameter further amplifies the impact of noise.

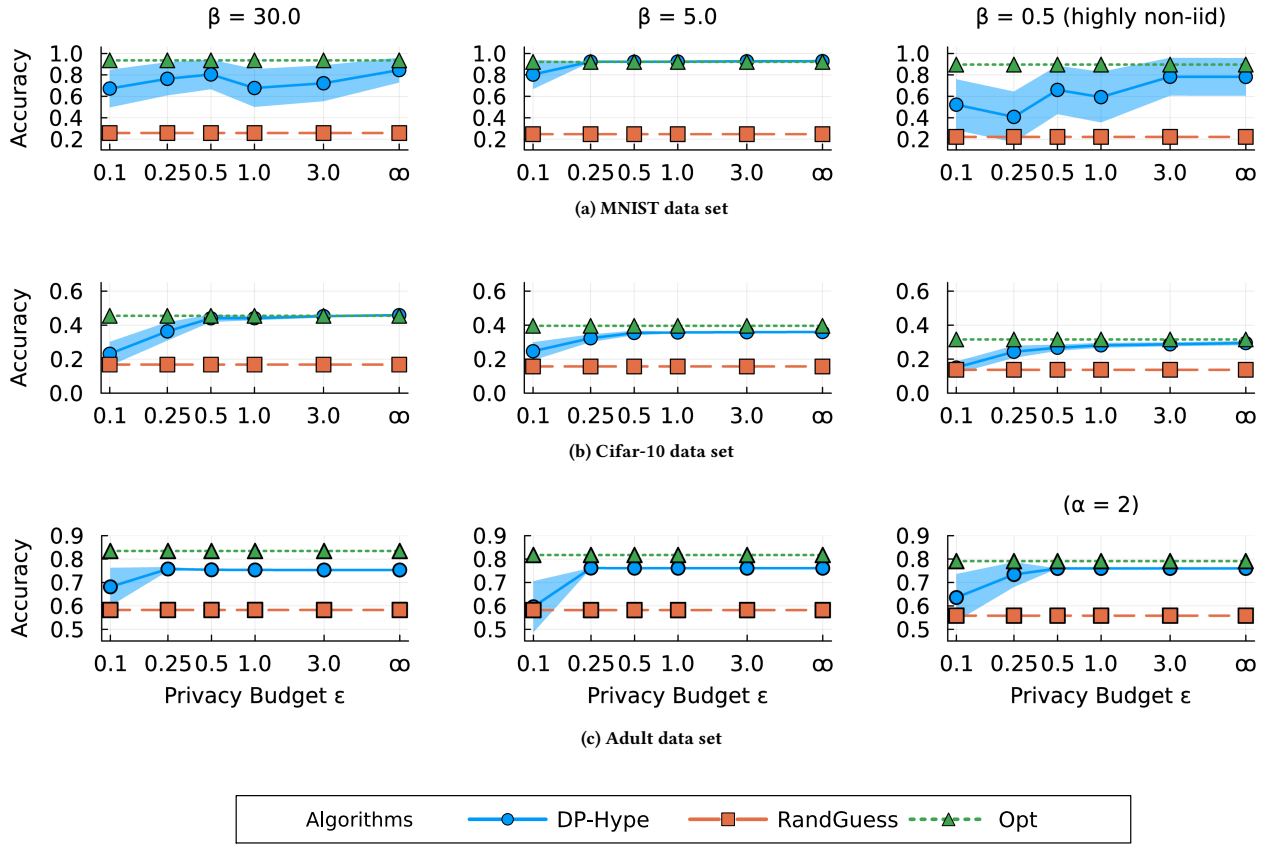


Figure 12: Empirical results in terms of Accuracy for evaluating DP-Hype, RandGuess and Opt on the data sets MNIST, Cifar-10 and Adult for $n = 250$ and different privacy budgets. We train a model in different non-iid scenario with Dirichlet concentration parameter $\beta \in \{30.0, 5.0, 0.5\}$ and $\beta = 2.0$ for $n = 250$ on Adult. The degree of non-iid β increases from left to right, which means that the data heterogeneity increases as well.

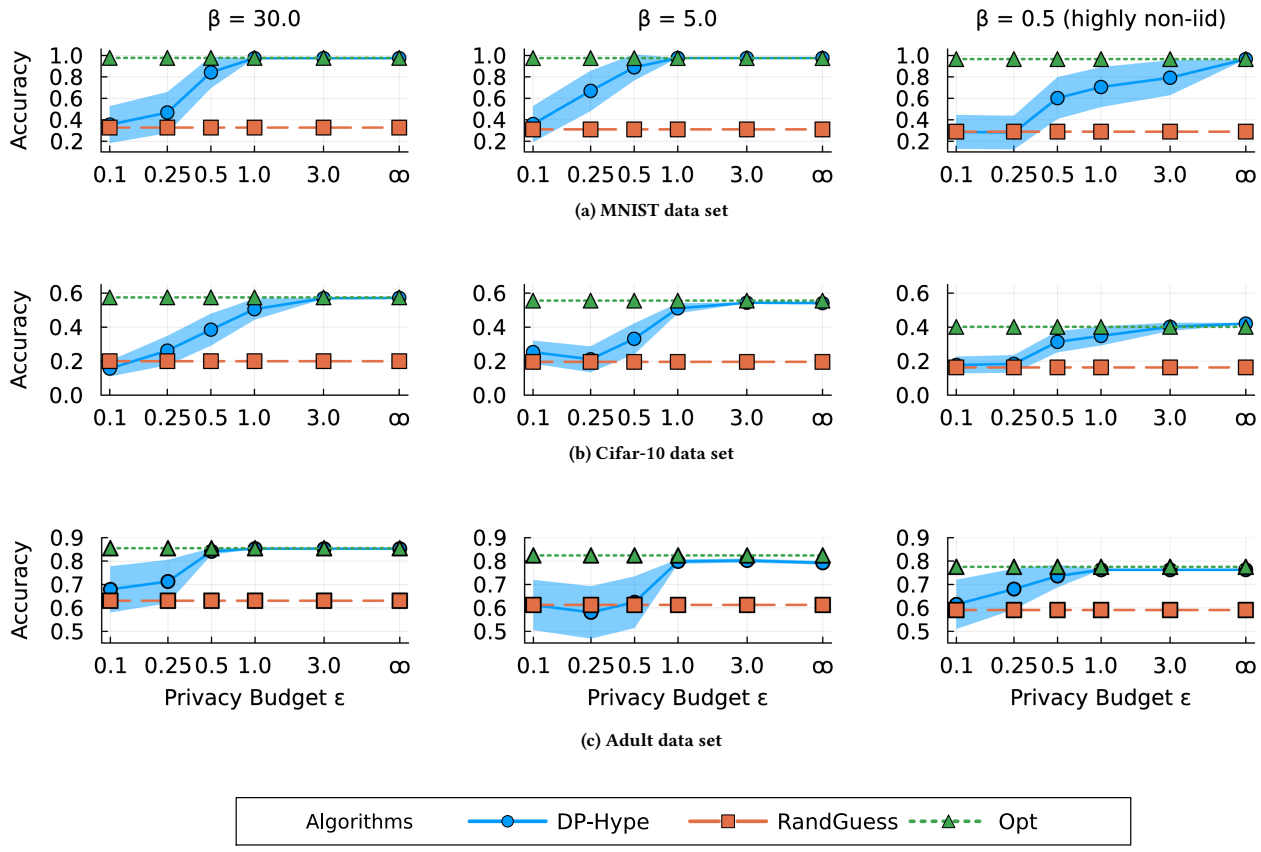


Figure 13: Empirical results in terms of Accuracy for evaluating DP-Hype, RandGuess and Opt on the data sets MNIST, Cifar-10 and Adult for $n = 50$ and different privacy budgets. We train a model in different non-iid scenario with Dirichlet concentration parameter $\beta \in \{30.0, 5.0, 0.5\}$. The degree of non-iid β increases from left to right, which means that the data heterogeneity increases as well.