Brendan Avent, Javier González, Tom Diethe, Andrei Paleyes, and Borja Balle

Automatic Discovery of Privacy–Utility Pareto Fronts

Abstract: Differential privacy is a mathematical framework for privacy-preserving data analysis. Changing the hyperparameters of a differentially private algorithm allows one to trade off privacy and utility in a principled way. Quantifying this trade-off in advance is essential to decision-makers tasked with deciding how much privacy can be provided in a particular application while maintaining acceptable utility. Analytical utility guarantees offer a rigorous tool to reason about this trade-off, but are generally only available for relatively simple problems. For more complex tasks, such as training neural networks under differential privacy, the utility achieved by a given algorithm can only be measured empirically. This paper presents a Bayesian optimization methodology for efficiently characterizing the privacy–utility trade-off of any differentially private algorithm using only empirical measurements of its utility. The versatility of our method is illustrated on a number of machine learning tasks involving multiple models, optimizers, and datasets.

Keywords: Differential privacy, Pareto front, Bayesian optimization

DOI 10.2478/popets-2020-0060
Received 2020-02-29; revised 2020-06-15; accepted 2020-06-16.

1 Introduction

Differential privacy (DP) [15] is the de-facto standard for privacy-preserving data analysis, including the training of machine learning models using sensitive data. The strength of DP comes from its use of randomness to hide the contribution of any individual's data from an adversary with access to *arbitrary side knowledge*. The price of DP is a loss in utility caused by the need to inject noise into computations. Quantifying the trade-off between privacy and utility is a central topic in the literature on differential privacy. Formal analysis of such trade-offs lead to algorithms achieving a pre-specified level privacy with minimal utility reduction, or, conversely, an a priori acceptable level of utility with maximal privacy. Since the choice of privacy level is generally regarded as a policy decision [41], this quantification is essential to decision-makers tasked with balancing utility and privacy in real-world deployments [3].

However, analytical analyses of the privacy–utility trade-off are only available for relatively simple problems amenable to mathematical treatment, and cannot be conducted for most problems of practical interest. Further, differentially private algorithms have more hyperparameters than their non-private counterparts, most of which affect both privacy and utility. In practice, tuning these hyperparameters to achieve an optimal privacy–utility trade-off can be an arduous task, especially when the utility guarantees are loose or unavailable. In this paper we develop a Bayesian optimization approach for *empirically* characterizing any differentially private algorithm's privacy–utility trade-off via principled, computationally efficient hyperparameter tuning.

A canonical application of our methods is differentially private deep learning. Differentially private stochastic optimization has been employed to train feed-forward [1], convolutional [10], and recurrent [38] neural networks, showing that reasonable accuracies can be achieved when selecting hyperparameters carefully. These works rely on a differentially private *gradient perturbation* technique, which clips and adds noise to gradient computations, while keeping track of the privacy loss incurred. However, these results do not provide actionable information regarding the privacy–utility trade-off of the proposed models. For example, private stochastic optimization methods can obtain the same level of privacy in different ways (e.g. by increasing the noise variance and reducing the clipping norm, or *vice-versa*), and it is not generally clear what combinations of these changes yield the best possible utility for a fixed privacy level. Furthermore, increasing the number of hyperparameters makes exhaustive hyperparameter optimization prohibitively expensive.

Brendan Avent: University of Southern California†, E-mail: bavent@usc.edu
Javier González: Now at Microsoft Research†, E-mail: Gonzalez.Javier@microsoft.com
Tom Diethe: Amazon Research Cambridge, E-mail: tdiethe@amazon.com
Andrei Paleyes: Now at University of Cambridge†, E-mail: ap2169@cam.ac.uk
Borja Balle: Now at DeepMind†, E-mail: borja.balle@gmail.com

† Work done while at Amazon Research Cambridge.

The goal of this paper is to provide a computationally efficient methodology to this problem by using Bayesian optimization to non-privately estimate the privacy–utility *Pareto front* of a given differentially private algorithm. The Pareto fronts obtained by our method can be used to select hyperparameter settings of the optimal operating points of any differentially private technique, enabling decision-makers to take informed actions when balancing the privacy–utility trade-off of an algorithm before deployment. This is in line with the approach taken by the U.S. Census Bureau to calibrate the level of DP that will be used when releasing the results of the upcoming 2020 census [2, 3, 19].

Our contributions are: (1) Characterizing the privacy–utility trade-off of a hyperparameterized algorithm as the problem of learning a Pareto front on the privacy vs. utility plane (Sec. 2). (2) Designing DPareto, an algorithm that leverages multi-objective Bayesian optimization techniques for learning the privacy–utility Pareto front of any differentially private algorithm (Sec. 3). (3) Instantiating and experimentally evaluating our framework for the case of differentially private stochastic optimization on a variety of learning tasks involving multiple models, optimizers, and datasets (Sec. 4). Finally, important and challenging extensions to this work are proposed (Sec. 5) and closely-related work is reviewed (Sec. 6).

2 The Privacy–Utility Pareto Front

This section provides an abstract formulation of the problem we want to address. We start by introducing some basic notation and recalling the definition of differential privacy. We then formalize the task of quantifying the privacy–utility trade-off using the notion of a Pareto front. Finally, we conclude by illustrating these concepts using classic examples from both machine learning as well as differential privacy.

2.1 General Setup

Let $A : \mathcal{Z}^n \to \mathcal{W}$ be a *randomized algorithm* that takes as input a tuple containing n records from $\mathcal{Z}$ and outputs a value in some set $\mathcal{W}$. Differential privacy formalizes the idea that A preserves the privacy of its inputs when the output distribution is stable under changes in one input.

Definition 1 ([14, 15])**.** *Given $\varepsilon \geq 0$ and $\delta \in [0, 1]$, we say algorithm A is (ε, δ)-DP if for any pair of inputs z, z' differing in a single coordinate we have*[1]

$$\sup_{E \subseteq \mathcal{W}} \left(\mathbb{P}[A(z) \in E] - e^{\varepsilon} \mathbb{P}[A(z') \in E] \right) \leq \delta \ .$$

To analyze the trade-off between utility and privacy for a given problem, we consider a *parametrized* family of algorithms $\mathcal{A} = \{A_\lambda : \mathcal{Z}^n \to \mathcal{W}\}$. Here, $\lambda \in \Lambda$ indexes the possible choices of hyperparameters, so $\mathcal{A}$ can be interpreted as the set of all possible algorithm configurations for solving a given task. For example, in the context of a machine learning application, the family $\mathcal{A}$ consists of a set of learning algorithms which take as input a training dataset $z = (z_1, \ldots, z_n)$ containing n example-label pairs $z_i = (x_i, y_i) \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ and produce as output the parameters $w \in \mathcal{W} \subseteq \mathbb{R}^d$ of a predictive model. It is clear that in this context different choices for the hyperparameters might yield different utilities. We further assume each configuration A_λ of the algorithm satisfies DP with potentially distinct privacy parameters.

To capture the privacy–utility trade-off across $\mathcal{A}$ we introduce two oracles to model the effect of hyperparameter changes on the privacy and utility of A_λ. A *privacy oracle* is a function $\mathsf{P}_\delta : \Lambda \to [0, +\infty]$ that given a choice of hyperparameters λ returns a value $\varepsilon = \mathsf{P}_\delta(\lambda)$ such that A_λ satisfies (ε, δ)-DP for a given δ. An instance-specific *utility oracle* is a function $\mathsf{U}_z : \Lambda \to [0, 1]$ that given a choice of hyperparameters λ returns some measure of the utility[2] of the output distribution of $A_\lambda(z)$. These oracles allow us to condense everything about our problem in the tuple $(\Lambda, \mathsf{P}_\delta, \mathsf{U}_z)$. Given these three objects, our goal is to find hyperparameter settings for A_λ that simultaneously achieve maximal privacy and utility on a given input z. Next we will formalize this goal using the concept of Pareto front, but we first provide remarks about the definition of our oracles.

Remark 1 (Privacy Oracle)**.** Parametrizing the privacy oracle P_δ in terms of a fixed δ stems from the convention that ε is considered the most important privacy parameter[3], whereas δ is chosen to be a negligibly small value ($\delta \ll 1/n$). This choice is also aligned with

1 Smaller values of ε and δ yield more private algorithms.

2 Due to the broad applicability of DP, concrete utility measures are generally defined on a per-problem basis. Here we use the conventions that U_z is bounded and that larger utility is better.

3 This choice comes without loss of generality since there is a connection between the two parameters guaranteeing the existence of a valid ε for any valid δ [6].

recent uses of DP in machine learning where the privacy analysis is conducted under the framework of Rényi DP [39] and the reported privacy is obtained a posteriori by converting the guarantees to standard (ε, δ)-DP for some fixed δ [1, 18, 22, 38, 48]. In particular, in our experiments with gradient perturbation for stochastic optimization methods (Sec. 4), we implement the privacy oracle using the moments accountant technique proposed by Abadi et al. [1] coupled with the tight bounds provided by Wang et al. [48] for Rényi DP amplification by subsampling without replacement. More generally, privacy oracles can take the form of analytic formulas or numerical optimized calculations, but future advances in empirical or black-box evaluation of DP guarantees could also play the role of privacy oracles.

Remark 2 (Utility Oracle)**.** Parametrizing the utility oracle U_z by a fixed input is a choice justified by the type of applications we tackle in our experiments (cf. Sec. 4). Other applications might require variations which our framework can easily accommodate by extending the definition of the utility oracle. We also stress that since the algorithms in $\mathcal{A}$ are randomized, the utility $\mathsf{U}_z(\lambda)$ is a property of the output *distribution* of $A_\lambda(z)$. This means that in practice we might have to implement the oracle approximately, e.g. through sampling. In particular, in our experiments we use a test set to measure the utility of a hyperparameter setting by running $A_\lambda(z)$ a fixed number of times R to obtain model parameters $w_1, \ldots, w_R$, and then let $\mathsf{U}_z(\lambda)$ be the average accuracy of the models on the test set.

The Pareto front of a collection of points $\Gamma \subset \mathbb{R}^p$ contains all the points in Γ where none of the coordinates can be decreased further without increasing some of the other coordinates (while remaining inside Γ).

Definition 2 (Pareto Front)**.** *Let $\Gamma \subset \mathbb{R}^p$ and $u, v \in \Gamma$. We say that u* dominates *v if $u_i \leq v_i$ for all $i \in [p]$, and we write $u \preceq v$. The* Pareto front *of Γ is the set of all non-dominated points $\mathcal{PF}(\Gamma) = \{u \in \Gamma \mid v \not\preceq u,\ \forall\, v \in \Gamma \setminus \{u\}\}$.*

According to this definition, given a privacy–utility trade-off problem of the form $(\Lambda, \mathsf{P}_\delta, \mathsf{U}_z)$, we are interested in finding the Pareto front $\mathcal{PF}(\Gamma)$ of the 2-dimensional set[4] $\Gamma = \{(\mathsf{P}_\delta(\lambda), 1 - \mathsf{U}_z(\lambda)) \mid \lambda \in \Lambda\}$. If able to fully characterize this Pareto front, a decision-maker looking to deploy DP would have all the necessary information to make an informed decision about how to trade-off privacy and utility in their application.

Threat Model Discussion

In the idealized setting presented above, the desired output is the Pareto front $\mathcal{PF}(\Gamma)$ which depends on z through the utility oracle; this is also the case for the Bayesian optimization algorithm for approximating the Pareto front presented in Sec. 3. This warrants a discussion about what threat model is appropriate here.

DP guarantees that an adversary observing the output $w = A_\lambda(z)$ will not be able to infer too much about any individual record in z. The (central) threat model for DP assumes that z is owned by a trusted curator that is responsible for running the algorithm and releasing its output to the world. However, the framework described above does not attempt to prevent information about z from being exposed by the Pareto front. This is because our methodology is only meant to provide a substitute for using closed-form utility guarantees when selecting hyperparameters for a given DP algorithm *before its deployment*. Accordingly, throughout this work we assume the Pareto fronts obtained with our method are only revealed to a small set of trusted individuals, which is the usual scenario in an industrial context. Privatization of the estimated Pareto front would remove the need for this assumption, and is discussed in Sec. 5 as a useful extension of this work.

An alternative approach is to assume the existence of a *public* dataset z_0 following a similar distribution to the private dataset z on which we would like to run the algorithm. Then we can use z_0 to compute the Pareto front of the algorithm, select hyperparameters λ^* achieving a desired privacy–utility trade-off, and release the output of $A_{\lambda^*}(z)$. In particular, this is the threat model used by the U.S. Census Bureau to tune the parameters for their use of DP in the context of the 2020 census (see Sec. 6 for more details).

2.2 Two Illustrative Examples

To concretely illustrate the oracles and Pareto front concept, we consider two distinct examples: private logistic regression and the sparse vector technique. Both examples are computationally light, and thus admit computation of near-exact Pareto fronts via a fine-grained grid-search on a low-dimensional hyperparameter space; for

4 The use of $1 - \mathsf{U}_z(\lambda)$ for the utility coordinate is for notational consistency, since we use the convention that the points in the Pareto front are those that minimize each individual dimension.

brevity, we subsequently refer to these as the "exact" or "true" Pareto fronts.

Private Logistic Regression

Here, we consider a simple private logistic regression model with ℓ_2 regularization trained on the Adult dataset [28]. The model is privatized by training with mini-batched projected SGD, then applying a Gaussian perturbation at the output using the method from [49, Algorithm 2] with default parameters[5]. The only hyperparameters tuned in this example are the regularization γ and the noise standard deviation σ, while the rest are fixed[6]. Note that both hyperparameters affect privacy and accuracy in this case. To implement the privacy oracle we compute the global sensitivity according to [49, Algorithm 2] and find the ε for a fixed $\delta = 10^{-6}$ using the exact analysis of the Gaussian mechanism provided in [7]. To implement the utility oracle we evaluate the accuracy of the model on the test set, averaging over 50 runs for each setting of the hyperparameters. To obtain the exact Pareto front for this problem, we perform a fine grid search over $\gamma \in [10^{-4}, 10^{0}]$ and $\sigma \in [10^{-1}, 10^{1}]$. The Pareto front and its corresponding hyperparameter settings are displayed in Fig. 1, along with the values returned by the privacy and utility oracles across the entire range of hyperparameters.

Sparse Vector Technique

The *sparse vector technique* (SVT) [16] is an algorithm to privately run m queries against a fixed sensitive database and release under DP the indices of those queries which exceed a certain threshold. The naming of the algorithm reflects the fact that it is specifically designed to have good accuracy when only a small number of queries are expected to be above the threshold. The algorithm has found applications in a number of problems, and several variants of it have been proposed [36].

Alg. 1 details our construction of a non-interactive version of the algorithm proposed in [36, Alg. 7]. Unlike the usual SVT that is parametrized by the target privacy ε, our construction takes as input a total noise level b and is tailored to answer m binary queries $q_i : \mathcal{Z}^n \to \{0, 1\}$ with sensitivity $\Delta = 1$ and fixed threshold $T = 1/2$. The privacy and utility of the algorithm are controlled by the noise level b and the bound C on the number of answers; increasing b or decreasing C yields a more private but less accurate algorithm. This noise level is split across two parameters b_1 and b_2 controlling how much noise is added to the threshold and to the query answers respectively[7]. Privacy analysis of Alg. 1 yields the following closed-form privacy oracle for our algorithm: $\mathsf{P}_0 = (1 + (2C)^{1/3})(1 + (2C)^{2/3})b^{-1}$ (refer to Appx. B for proof).

Algorithm 1: Sparse Vector Technique

Input: dataset z, queries $q_1, \dots, q_m$
Hyperparameters: noise b, bound C
$c \leftarrow 0$, $w \leftarrow (0, \dots, 0) \in \{0, 1\}^m$
$b_1 \leftarrow b/(1 + (2C)^{1/3})$, $b_2 \leftarrow b - b_1$, $\rho \leftarrow \mathsf{Lap}(b_1)$
for $i \in [m]$ **do**
  $\nu \leftarrow \mathsf{Lap}(b_2)$
  if $q_i(z) + \nu \geq \frac{1}{2} + \rho$ **then**
    $w_i \leftarrow 1$, $c \leftarrow c + 1$
    if $c \geq C$ **then return** w
return w

As a utility oracle, we use the F_1-*score* between the vector of true answers $(q_1(z), \dots, q_m(z))$ and the vector w returned by the algorithm. This measures how well the algorithm identifies the support of the queries that return 1, while penalizing both for false positives and false negatives. This is again different from the usual utility analyses of SVT algorithms, which focus on providing an interval around the threshold outside which the output is guaranteed to have no false positives or false negatives [17]. Our measure of utility is more fine-grained and relevant for practical applications, although to the best of our knowledge no theoretical analysis of the utility of the SVT in terms of F_1-score is available in the literature.

In this example, we set $m = 100$ and pick queries at random such that exactly 10 of them return a 1. Since the utility of the algorithm is sensitive to the query order, we evaluate the utility oracle by running the algorithm 50 times with a random query order and compute the average F_1-score. The Pareto front and its corresponding hyperparameter settings are displayed in Fig. 2, along with the values returned by the privacy and utility oracles across the entire range of hyperparameters.

5 These are the smoothness, Lipschitz and strong convexity parameters of the loss, and the learning rate.

6 Mini-batch size $m = 1$ and number of epochs $T = 10$.

7 The split used by the algorithm is based on the privacy budget allocation suggested in [36, Section 4.2].

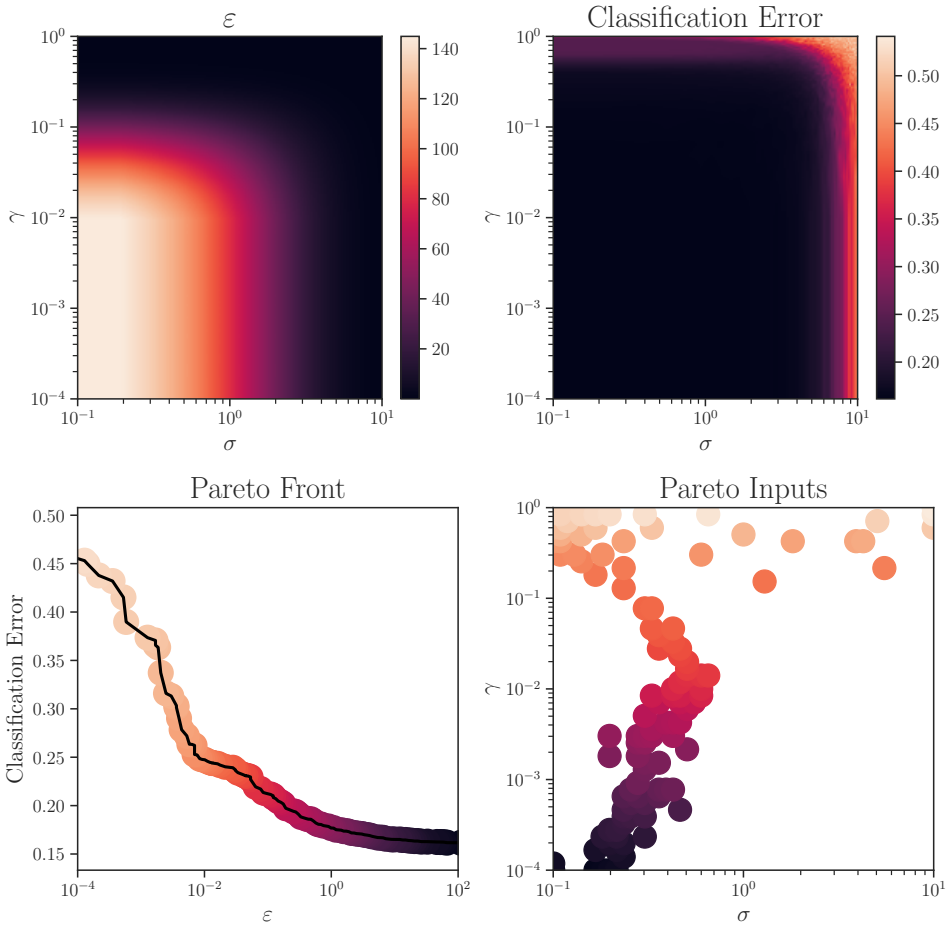

Fig. 1. *Top:* Values returned by the privacy and utility oracles across a range of hyperparameters in the private logistic regression example. *Bottom:* The Pareto front and its corresponding set of input points.

3 DPareto: Learning the Pareto Front

This section starts by recalling the basic ideas behind multi-objective Bayesian optimization. We then describe our proposed methodology to efficiently learn the privacy–utility Pareto front. Finally, we revisit the private logistic regression and SVT examples to illustrate our method.

3.1 Multi-objective Bayesian Optimization

Bayesian optimization (BO) [40] is a strategy for sequential decision making useful for optimizing expensive-to-evaluate black-box objective functions. It has become increasingly relevant in machine learning due to its success in the optimization of model hyperparameters [24, 45]. In its standard form, BO is used to find the minimum of an objective function $f(\lambda)$ on some subset $\Lambda \subseteq \mathbb{R}^p$ of a Euclidean space of moderate dimension. It works by generating a sequence of evaluations of the objective at locations $\lambda_1, \ldots, \lambda_k$, which is done by (i) building a surrogate model of the objective function using the current data and (ii) applying a pre-specified criterion to select a new location λ_{k+1} based on the model until a budget is exhausted. In the single-objective case, a common choice is to select the location that, in expectation under the model, gives the best improvement to the current estimate [40].

In this work, we use BO for learning the privacy–utility Pareto front. When used in multi-objective problems, BO aims to learn a Pareto front with a minimal number of evaluations, which makes it an appealing tool in cases where evaluating the objectives is expensive. Although in this paper we only work with two objective functions, we detail here the general case of minimizing p objectives $f_1, \ldots, f_p$ simultaneously. This generaliza-

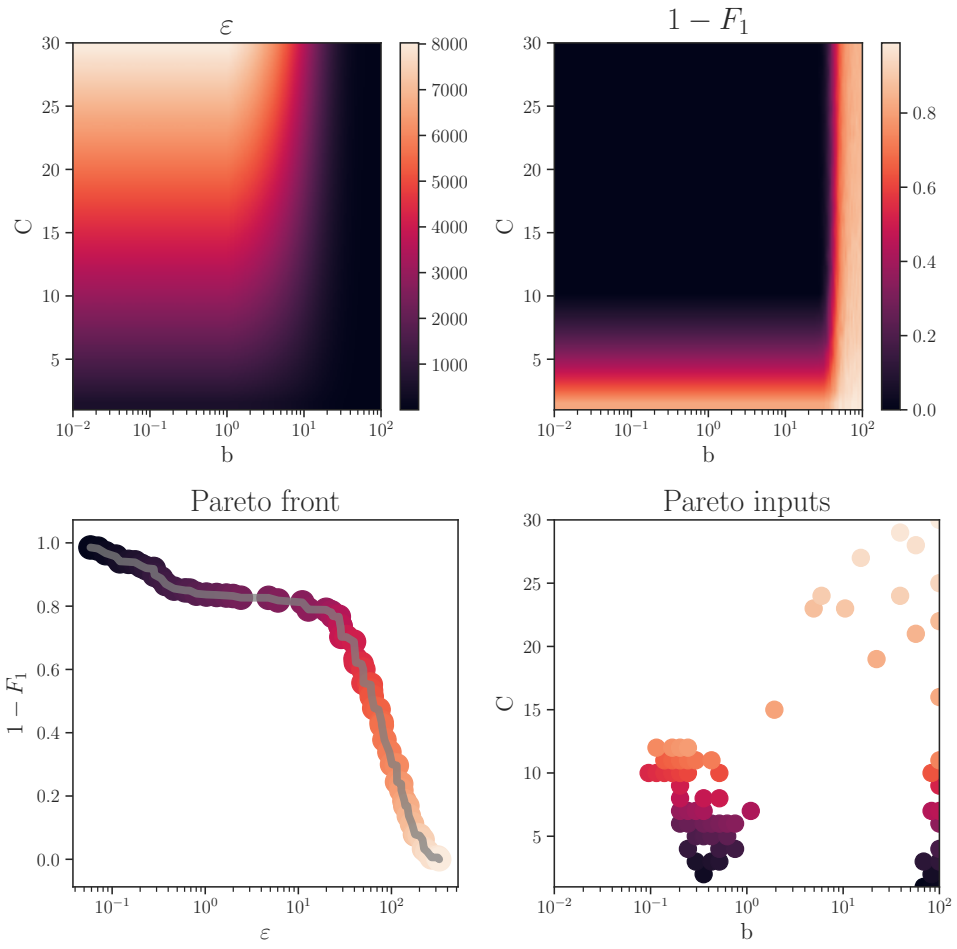

Fig. 2. *Top:* Values returned by the privacy and utility oracles across a range of hyperparameters in the SVT example. *Bottom:* The Pareto front and its corresponding set of input points.

tion could be used, for instance, to introduce the running time of the algorithm as a third objective to be traded off against privacy and utility.

Let $\lambda_1, \ldots, \lambda_k$ be a set of locations in Λ and denote by $\mathcal{V} = \{v_1, \ldots, v_k\}$ the set such that each $v_i \in \mathbb{R}^p$ is the vector $(f_1(\lambda_i), \ldots, f_p(\lambda_i))$. In a nutshell, BO works by iterating over the following:

1. Fit a surrogate model of the objectives $f_1(\lambda) \ldots, f_p(\lambda)$ using the available dataset $\mathcal{D} = \{(\lambda_i, v_i)\}_{i=1}^k$. The standard approach is to use a Gaussian process (GP) [42].
2. For each objective f_j calculate the predictive distribution over $\lambda \in \Lambda$ using the surrogate model. If GPs are used, the predictive distribution of each output can be fully characterized by their mean $m_j(\lambda)$ and variance $s_j^2(\lambda)$ functions, which can be computed in closed form.
3. Use the posterior distribution of the surrogate model to form an acquisition function $\alpha(\lambda; \mathcal{I})$, where $\mathcal{I}$ represents the dataset $\mathcal{D}$ and the GP posterior conditioned on $\mathcal{D}$.
4. Collect the next evaluation point λ_{k+1} at the (numerically estimated) global maximum of $\alpha(\lambda; \mathcal{I})$.

The process is repeated until the budget to collect new locations is over.

There are two key aspects of any BO method: the surrogate model of the objectives and the acquisition function $\alpha(\lambda; \mathcal{I})$. In this work, we use independent GPs as the surrogate models for each objective; however, generalizations with multi-output GPs [4] are possible.

Acquisition with Pareto Front Hypervolume

Next we define an acquisition criterion $\alpha(\lambda; \mathcal{I})$ useful to collect new points when learning the Pareto front. Let $\mathcal{P} = \mathcal{PF}(\mathcal{V})$ be the Pareto front computed with the objective evaluations in $\mathcal{I}$ and let $v^\dagger \in \mathbb{R}^p$ be some chosen

"anti-ideal" point[8]. To measure the relative merit of different Pareto fronts we use the *hypervolume* $\mathsf{HV}_{v^\dagger}(\mathcal{P})$ of the region dominated by the Pareto front $\mathcal{P}$ bounded by the anti-ideal point. Mathematically this can be expressed as $\mathsf{HV}_{v^\dagger}(\mathcal{P}) = \mu(\{v \in \mathbb{R}^p \mid v \preceq v^\dagger,\ \exists u \in \mathcal{P}\ u \preceq v\})$, where μ denotes the standard Lebesgue measure on $\mathbb{R}^p$. Henceforth we assume the anti-ideal point is fixed and drop it from our notation.

Larger hypervolume therefore implies that points in the Pareto front are closer to the ideal point $(0, 0)$. Thus, $\mathsf{HV}(\mathcal{PF}(\mathcal{V}))$ provides a way to measure the quality of the Pareto front obtained from the data in $\mathcal{V}$. Furthermore, hypervolume can be used to design acquisition functions for selecting hyperparameters that will improve the Pareto front. Start by defining the increment in the hypervolume given a new point $v \in \mathbb{R}^p$: $\Delta_{\mathcal{PF}}(v) = \mathsf{HV}(\mathcal{PF}(\mathcal{V} \cup \{v\})) - \mathsf{HV}(\mathcal{PF}(\mathcal{V}))$. This quantity is positive only if v lies in the set $\tilde{\Gamma}$ of points non-dominated by $\mathcal{PF}(\mathcal{V})$. Therefore, the *probability of improvement* (PoI) over the current Pareto front when selecting a new hyperparameter λ can be computed using the model trained on $\mathcal{I}$ as follows:

$$\mathsf{PoI}(\lambda) = \mathbb{P}[(f_1(\lambda), \ldots, f_p(\lambda)) \in \tilde{\Gamma} \mid \mathcal{I}] \tag{1}$$

$$= \int_{v \in \tilde{\Gamma}} \prod_{j=1}^{p} \phi_j(\lambda; v_j) dv_j \ , \tag{2}$$

where $\phi_j(\lambda; \cdot)$ is the predictive Gaussian density for f_j with mean $m_j(\lambda)$ and variance $s_j^2(\lambda)$.

The $\mathsf{PoI}(\lambda)$ function (1) accounts for the probability that a given $\lambda \in \Lambda$ has to improve the Pareto front, and it can be used as a criterion to select new points. However, in this work, we opt for the *hypervolume-based* PoI (HVPoI) due to its superior computational and practical properties [13]. The HVPoI is given by

$$\alpha(\lambda; \mathcal{I}) = \Delta_{\mathcal{PF}}(m(\lambda)) \cdot \mathsf{PoI}(\lambda) \ , \tag{3}$$

where $m(\lambda) = (m_1(\lambda), \ldots, m_p(\lambda))$. This acquisition weights the probability of improving the Pareto front with a measure of how much improvement is expected, computed using the GP means of the outputs. The HVPoI has been shown to work well in practice and efficient implementations exist [27].

8 The anti-ideal point must be dominated by all points in $\mathcal{PF}(\mathcal{V})$. In the private logistic regression example, this could correspond to the worst-case ϵ and worst-case classification error. See Couckuyt et al. [13] for further details.

3.2 The DPareto Algorithm

The main optimization loop of DPareto is shown in Alg. 2. It combines the two ingredients sketched so far: GPs for surrogate modeling of the objective oracles, and HVPoI as the acquisition function to select new hyperparameters. The basic procedure is to first seed the optimization by selecting k_0 hyperparameters from Λ at random, and then fit the GP models for the privacy and utility oracles based on these points. We then maximize of the HVPoI acquisition function to obtain the next query point, which is then added into the dataset. This is repeated k times until the optimization budget is exhausted.

Algorithm 2: DPareto

Input: hyperparameter set Λ, privacy and utility oracles $\mathsf{P}_\delta, \mathsf{U}_z$, anti-ideal point $v^\dagger$, number of initial points k_0, number of iterations k, prior GP

Initialize dataset $\mathcal{D} \leftarrow \emptyset$

for $i \in [k_0]$ **do**
- Sample random point $\lambda \in \Lambda$
- Evaluate oracles $v \leftarrow (\mathsf{P}_\delta(\lambda), 1 - \mathsf{U}_z(\lambda))$
- Augment dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\lambda, v)\}$

for $i \in [k]$ **do**
- Fit GPs to transformed privacy and utility using $\mathcal{D}$
- Obtain new query point λ by optimizing HVPoI in (3) using anti-ideal point $v^\dagger$
- Evaluate oracles $v \leftarrow (\mathsf{P}_\delta(\lambda), 1 - \mathsf{U}_z(\lambda))$
- Augment dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\lambda, v)\}$

return *Pareto front* $\mathcal{PF}(\{v \mid (\lambda, v) \in \mathcal{D}\})$

A Note on Output Domains

The output domains for the privacy and utility oracles may not be well-modeled by a GP, which models outputs on the entire real line. For instance, the output domain for the privacy oracle is $[0, +\infty]$. The output domain for the utility oracle depends on the chosen measure of utility. A common choice of utility oracle for ML tasks is accuracy, which has output domain $[0, 1]$. Thus, neither the privacy nor utility oracles are well-modeled by a GP as-is. Therefore, in both cases, we transform the outputs so that we are modeling a GP in the transformed space. For privacy, we use a simple log transform; for accuracy, we use a logit transform $\mathrm{logit}(x) = \log(x) - \log(1 - x)$. With this, both oracles

have transformed output domain $[-\infty, +\infty]$. Note that it is possible to *learn* the transformation using Warped GPs [44]. The advantage there is that the form of both the covariance matrix and the nonlinear transformation are learnt simultaneously under the same probabilistic framework. However, for simplicity and efficiency we choose to use fixed transformations.

3.3 Two Illustrative Examples: Revisited

We revisit the examples discussed in Sec. 2.2 to concretely illustrate how the components of DPareto work together to effectively learn the privacy–utility Pareto front.

Private Logistic Regression

For this example, we initialize the GP models with $k_0 = 250$ random hyperparameter pairs (γ_i, σ_i). γ_i takes values in $[10^{-4}, 10^{0}]$ and σ_i takes values in $[10^{-1}, 10^{1}]$, both sampled uniformly on a logarithmic scale. The privacy and mean utility of the trained models corresponding to each sample are computed, and GPs are fit to these values as surrogate models for each oracle. The predicted means of these surrogate models are shown in the top row of Fig. 3. Comparing directly to the oracles' true values in Fig. 1, we observe that the surrogate models have modeled them well in the high σ and γ regions, but is still learning the low regions. The bottom-left of Fig. 3 shows the exact Pareto front of the problem, along with the output values of the initial sample and the corresponding empirical Pareto front. The empirical Pareto front sits almost exactly on the true one, except in the extremely-high privacy region ($\varepsilon < 10^{-2}$) – this indicates that the selection of random points (γ_i, σ_i) was already quite good outside of this region. The goal of DPareto is to select new points in the input domain whose outputs will bring the empirical front closer to the true one. This is the purpose of the HVPoI function; the bottom-right of Fig. 3 shows the HVPoI function evaluated over all (γ_i, σ_i) pairs. The maximizer of this function, marked with a star, is used as the next location to evaluate the oracles. Note that given the current surrogate models, the HVPoI seems to be making a sensible choice: selecting a point where ε and classification error are both predicted to have relatively low values, possibly looking to improve the upper-left region of the Pareto front.

Sparse Vector Technique

For this example, we initialize the GP models with $k_0 = 250$ random hyperparameter pairs (C_i, b_i). The C_i values are sampled uniformly in the interval $[1, 30]$, and the b_i values are sampled uniformly in the interval $[10^{-2}, 10^{2}]$ on a logarithmic scale. The privacy and utility values are computed for each of the samples, and GPs are fit to these values as surrogate models for each oracle. The predicted means of these surrogate models are shown in the top row of Fig. 4. We observe that both surrogate models have modeled their oracles reasonably well, comparing directly to the oracles' true values in Fig. 2. The bottom-left of Fig. 4 shows the exact Pareto front of the problem, along with the output values of the initial sample and the corresponding empirical Pareto front. The empirical Pareto front sits close to the true one, which indicates that the selection of points (C_i, b_i) is already quite good. The HVPoI function is used by DPareto to select new points in the input domain whose outputs will bring the empirical front closer to the true one. The bottom-right of Fig. 4 shows this function evaluated over all (C_i, b_i) pairs. The maximizer of this function, marked with a star, is used as the next location to evaluate the oracles. Note that given the current surrogate models, the HVPoI appears to be making a sensible choice: selecting a point where ε is predicted to have a medium value and $1 - F_1$ is predicted to have a low value, possibly looking to improve the gap in the lower-right corner of the Pareto front.

4 Experiments

In this section, we provide experimental evaluations of DPareto on a number of ML tasks. Unlike the illustrations previously discussed in Secs. 2.2 and 3.3, it is computationally infeasible to compute exact Pareto fronts for these tasks. This highlights the advantage of using DPareto over random and grid search baselines, showcasing its versatility on a variety of models, datasets, and optimizers. See Appendix A for implementation details.

4.1 Experimental Setup

In all our experiments we used $v^{\dagger} = (10, 1)$ as the anti-ideal point in DPareto, encoding our interest in a Pareto front which captures a practical privacy range (i.e., $\varepsilon \leq 10$) across all possible utility values (since classification error can never exceed 1).

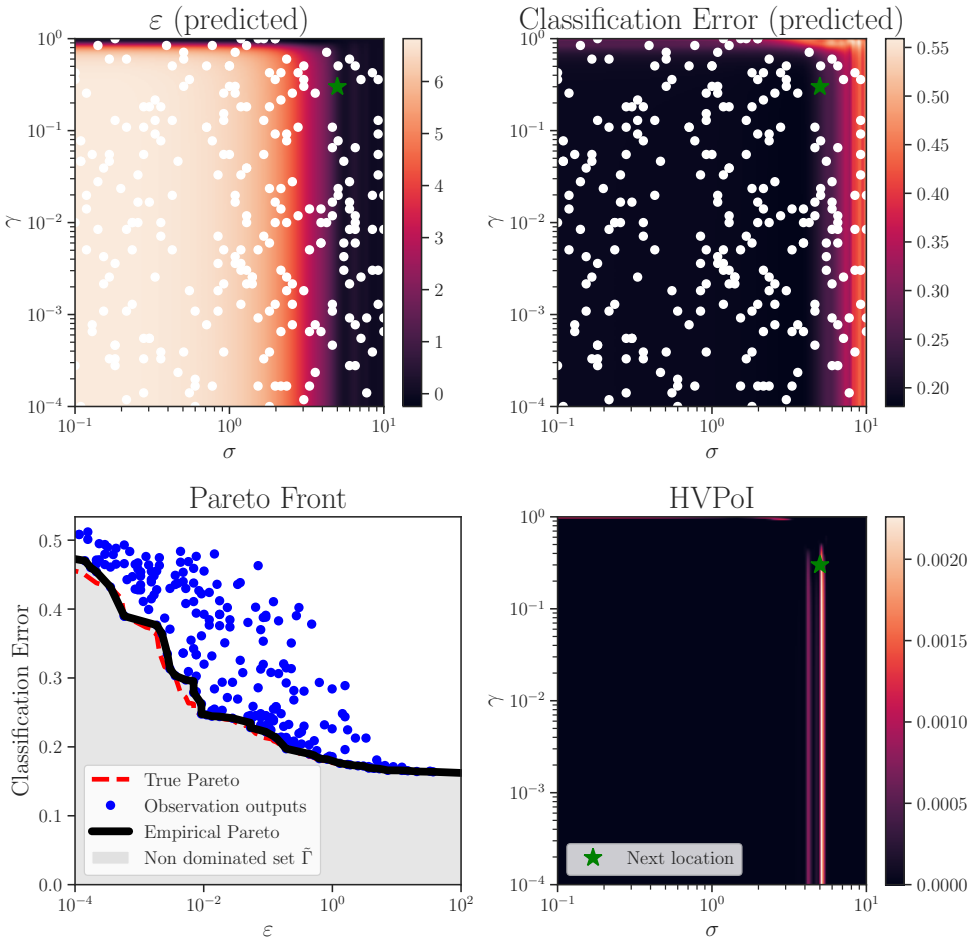

Fig. 3. *Top:* Mean predictions of the privacy (ε) and the utility (classification error) oracles using their respective GPs models in the private logistic regression example. The locations of the $k_0 = 250$ sampled points are plotted in white. *Bottom left:* Empirical and true Pareto fronts. *Bottom right:* HVPoI and the selected next location.

Optimization Domains and Sampling Distributions

Table 1 gives the optimization domain Λ for each of the different experiments, which all point-selection approaches (i.e., DPareto, random sampling, and grid search) operate within. Random sampling distributions for experiments on both MNIST and ADULT datasets were carefully constructed in order to generate as favorable results for random sampling as possible. These distributions are precisely detailed in Appx. C, and were constructed by reviewing literature (namely Abadi et al. [1] and McMahan et al. [38]) in addition to the authors' experience from training these differentially private models. The Pareto fronts generated from these constructed distributions were significantly better than those yielded by the naïve strategy of sampling from the uniform distribution, justifying the choice of these distributions.

Datasets

We tackle two classic problems: multiclass classification of handwritten digits with the MNIST dataset, and binary classification of income with the ADULT dataset. MNIST [31] is composed of 28×28 gray-scale images, each representing a single digit 0-9. It has 60k (10k) images in the training (test) set. ADULT [28] is composed of 123 binary demographic features on various people, with the task of predicting whether income $>$ \$50k. It has 40k (1.6k) points in the training (test) set.

Models

For ADULT dataset, we consider logistic regression (LogReg) and linear support vector machines (SVMs), and explore the effect of the choice of model and optimization algorithm (SGD vs. Adam), using the differentially private versions of these algorithms outlined in Sec. 4.2. For MNIST, we fix the optimization algorithm

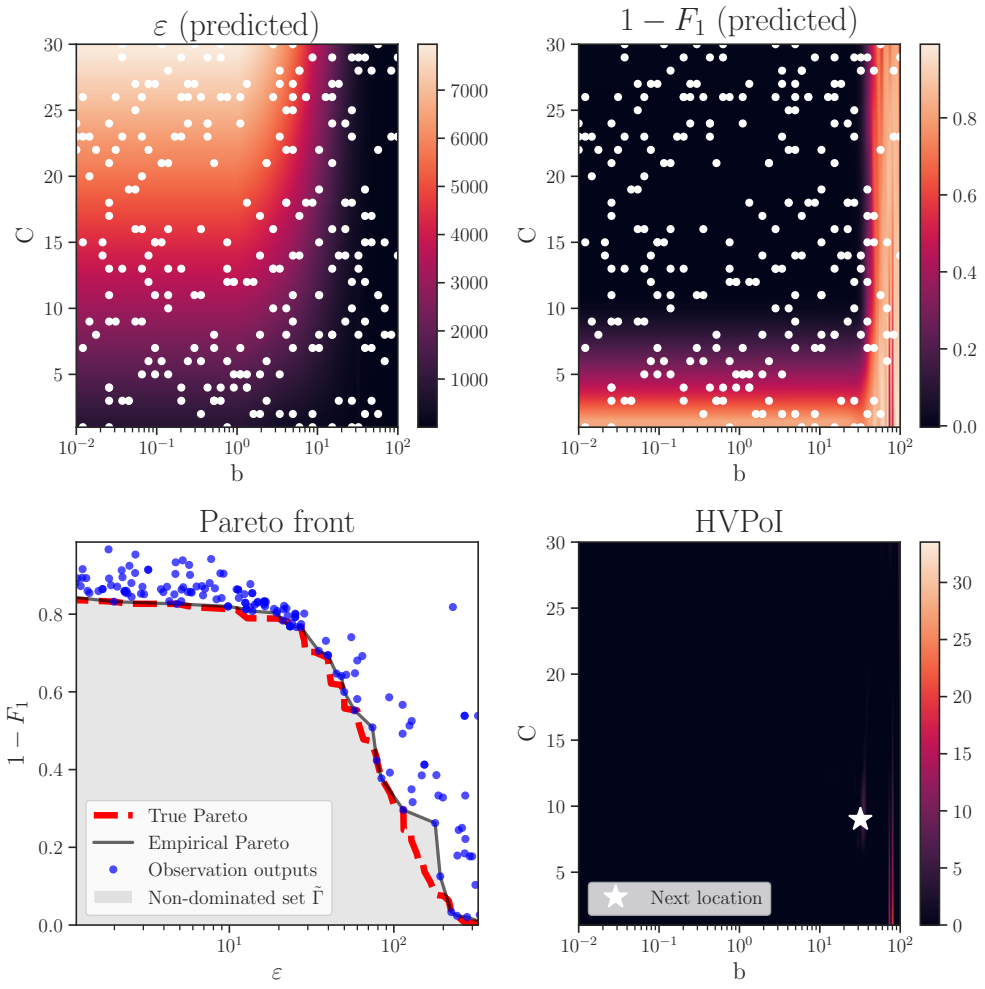

Fig. 4. *Top:* Mean predictions of the privacy (ε) and the utility ($1 - F_1$) oracles using their respective GPs models in the sparse vector technique example. The locations of the $k_0 = 250$ sampled points are plotted in white. *Bottom left:* Empirical and true Pareto fronts. *Bottom right:* HVPoI and the selected next location.

as SGD, but use a more expressive multilayer perceptron (MLP) model and explore the choice of network architectures. The first (MLP1) has a single hidden layer with 1000 neurons, which is the same as used by Abadi et al. [1] but without DP-PCA dimensionality reduction. The second (MLP2) has two hidden layers with 128 and 64 units. In both cases we use ReLU activations.

4.2 Privatized Optimization Algorithms

Experiments are performed with privatized variants of two popular optimization algorithms – stochastic gradient descent (SGD) [8] and Adam [25] – although our framework can easily accommodate other privatized algorithms when a privacy oracle is available. Stochastic gradient descent (SGD) is a simplification of gradient descent, where on each iteration instead of computing the gradient for the entire dataset, it is instead estimated on the basis of a single example (or small batch of examples) picked uniformly at random (without replacement) [8]. Adam [25] is a first-order gradient-based optimization algorithm for stochastic objective functions, based on adaptive estimates of lower-order moments.

As a privatized version of SGD, we use a mini-batched implementation with clipped gradients and Gaussian noise, detailed in Alg. 3. This algorithm is similar to that of Abadi et al.'s [1], but differs in two ways. First, it utilizes sampling without replacement to generate fixed-size mini-batches, rather than using Poisson sampling with a fixed probability which generates variable-sized mini-batches. Using fixed-size mini-matches is a is a more natural approach, which more-closely aligns with standard practice in non-private ML. Second, as the privacy oracle we use the moments ac-

Algorithm	Dataset	Epochs (T)	Lot Size (m)	Learning Rate (η)	Noise Variance (σ^2)	Clipping Norm (L)
LogReg+SGD	ADULT	$[1, 64]$	$[8, 512]$	$[5e{-}4, 5e{-}2]$	$[0.1, 16]$	$[0.1, 4]$
LogReg+Adam	ADULT	$[1, 64]$	$[8, 512]$	$[5e{-}4, 5e{-}2]$	$[0.1, 16]$	$[0.1, 4]$
SVM+SGD	ADULT	$[1, 64]$	$[8, 512]$	$[5e{-}4, 5e{-}2]$	$[0.1, 16]$	$[0.1, 4]$
MLP1+SGD	MNIST	$[1, 400]$	$[16, 4000]$	$[1e{-}3, 5e{-}1]$	$[0.1, 16]$	$[0.1, 12]$
MLP2+SGD	MNIST	$[1, 400]$	$[16, 4000]$	$[1e{-}3, 5e{-}1]$	$[0.1, 16]$	$[0.1, 12]$

Table 1. Optimization domains used in each of the experimental settings.

countant implementation of Wang et al. [48], which supports sampling without replacement. In Alg. 3, the function $\mathsf{clip}_L(v)$ acts as the identity if $\|v\|_2 \leq L$, and otherwise returns $(L/\|v\|_2)v$. This clipping operation ensures that $\|\mathsf{clip}_L(v)\|_2 \leq L$ so that the ℓ_2-sensitivity of any gradient to a change in one datapoint in z is always bounded by L/m.

Algorithm 3: Differentially Private SGD

Input: dataset $z = (z_1, \ldots, z_n)$
Hyperparameters: learning rate η,
mini-batch size m,
number of epochs T,
noise variance σ^2,
clipping norm L
Initialize $w \leftarrow 0$
for $t \in [T]$ **do**
  for $k \in [n/m]$ **do**
    Sample $S \subset [n]$ with $|S| = m$ uniformly at random
    Let $g \leftarrow \frac{1}{m}\sum_{j \in S} \mathsf{clip}_L(\nabla \ell(z_j, w)) + \frac{2L}{m}\mathcal{N}(0, \sigma^2 I)$
    Update $w \leftarrow w - \eta g$
return w

Our privatized version of Adam is given in Alg. 4, which uses the same gradient perturbation technique as stochastic gradient descent. Here the notation $g^{\odot 2}$ denotes the vector obtained by squaring each coordinate of g. Adam uses three numerical constants that are not present in SGD (κ, β_1 and β_2). To simplify our experiments, we fixed those constants to the defaults suggested in Kingma et al. [25].

Algorithm 4: Differentially Private Adam

Input: dataset $z = (z_1, \ldots, z_n)$
Hyperparameters: learning rate η,
mini-batch size m,
number of epochs T,
noise variance σ^2,
clipping norm L
Fix $\kappa \leftarrow 10^{-8}$, $\beta_1 \leftarrow 0.9$, $\beta_2 \leftarrow 0.999$
Initialize $w \leftarrow 0$, $\mu \leftarrow 0$, $\nu \leftarrow 0$, $i \leftarrow 0$
for $t \in [T]$ **do**
  for $k \in [n/m]$ **do**
    Sample $S \subset [n]$ with $|S| = m$ uniformly at random
    Let $g \leftarrow \frac{1}{m}\sum_{j \in S} \mathsf{clip}_L(\nabla \ell(z_j, w)) + \frac{2L}{m}\mathcal{N}(0, \sigma^2 I)$
    Update $\mu \leftarrow \beta_1 \mu + (1 - \beta_1) g$,
    $\nu \leftarrow \beta_2 \nu + (1 - \beta_2) g^{\odot 2}$, $i \leftarrow i + 1$
    De-bias $\hat{\mu} \leftarrow \mu/(1 - \beta_1^i)$, $\hat{\nu} \leftarrow \nu/(1 - \beta_2^i)$
    Update $w \leftarrow w - \eta \hat{\mu}/(\sqrt{\hat{\nu}} + \kappa)$
return w

4.3 Experimental Results

DPareto vs. Random Sampling

A primary purpose of these experiments is to highlight the efficacy of DPareto at estimating an algorithm's Pareto front. As discussed above, the hypervolume is a popular measure for quantifying the quality of a Pareto front. We compare DPareto to the traditional approach of random sampling by computing the hypervolumes of Pareto fronts generated by each method.

In Fig. 5 the first two plots show, for a variety of models, how the hypervolume of the Pareto front expands as new points are sampled. In nearly every experiment, the DPareto approach yields a greater hypervolume than the random sampling analog – a direct indicator that DPareto has better characterized the Pareto front. This can be seen by examining the bottom left plot of the figure, which directly shows a Pareto front of the MLP2 model with both sampling methods. Specifically, while the random sampling method only marginally improved over its initially seeded points, DPareto was able to thoroughly explore the high-privacy regime (i.e. small ε). The bottom right plot of the figure compares the DPareto approach with

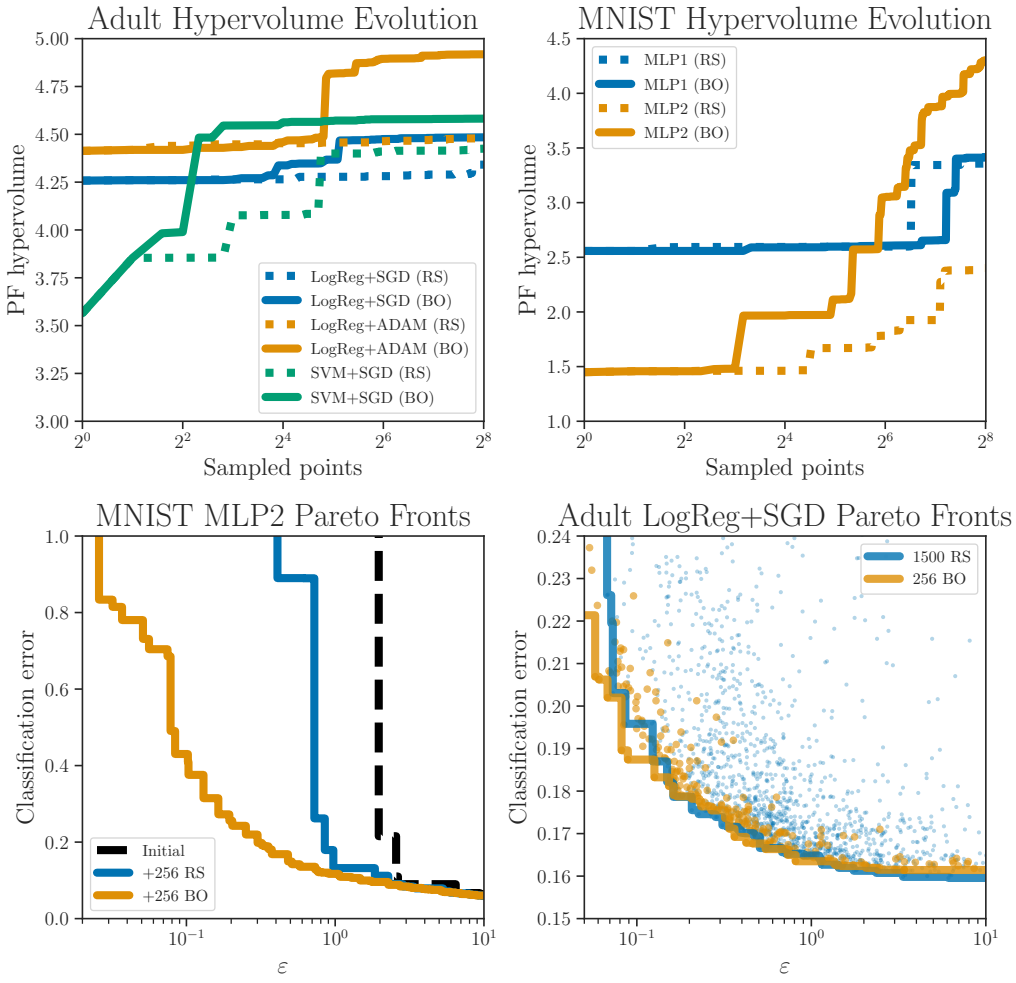

Fig. 5. *Top:* Hypervolumes of the Pareto fronts computed by the various models, optimizers, and architectures on the ADULT and MNIST datasets (respectively) by both DPARETO and random sampling. *Bottom left:* Pareto fronts learned for MLP2 architecture on the MNIST dataset with DPARETO and random sampling, including the shared points they were both initialized with. *Bottom right:* ADULT dataset DPARETO sampled points and its Pareto front compared with the larger set of random sampling points and its Pareto front.

256 sampled points against the random sampling approach with significantly more sampled points, 1500. While both approaches yield similar Pareto fronts, the efficiency of DPARETO is particularly highlighted by the points that are *not* on the front: nearly all the points chosen by DPARETO are close to the actual front, whereas many points chosen by random sampling are nowhere near it.

To quantify the differences between random sampling and DPARETO for the ADULT dataset, we split the 5000 random samples into 19 parts of size 256 to match the number of BO points, and computed hypervolume differences between the resultant Pareto fronts under the mild assumption that DPARETO is deterministic[9]. We then computed the two-sided confidence intervals for these differences, shown in Table 2. We also computed the t-statistic for these differences being zero, which were all highly significant ($p < 0.001$). This demonstrates that the observed differences between Pareto fronts are in fact statistically significant. We did not have enough random samples to run statistical tests for MNIST, however the differences are visually even clearer in this case.

9 While not strictly true, since BO is seeded with a random set of points, running repetitions would have been an extremely costly exercise with results expected to be nearly identical.

Algorithm+Optimizer	Mean Difference	95% C.I.
LogReg+SGD	0.158	$(0.053, 0.264)^*$
LogReg+ADAM	0.439	$(0.272, 0.607)^*$
SVM+SGD	0.282	$(0.161, 0.402)^*$

Table 2. Mean hypervolume differences between BO and 19 random repetitions of 256 iterations of random sampling. Two-sided 95% confidence intervals (C.I.) for these differences, as well as t-tests for the mean, are included. Asterisks indicate significance at the $p < 0.001$ level.

DPareto vs. Grid search

For completeness we also ran experiments using grid search with two different grid sizes, both of which performed significantly worse than DPareto. For these experiments, we have defined parameter ranges as the limiting parameter values from our random sampling experiment setup (see Table 4). We evaluated a grid size of 3, which corresponds to 243 total points (approximately the same amount of points as DPareto uses), and grid size 4, which corresponds to 1024 points (4 times more than were used for DPareto). As can be seen in Fig. 6, DPareto outperforms grid search even when significantly more grid points are evaluated.

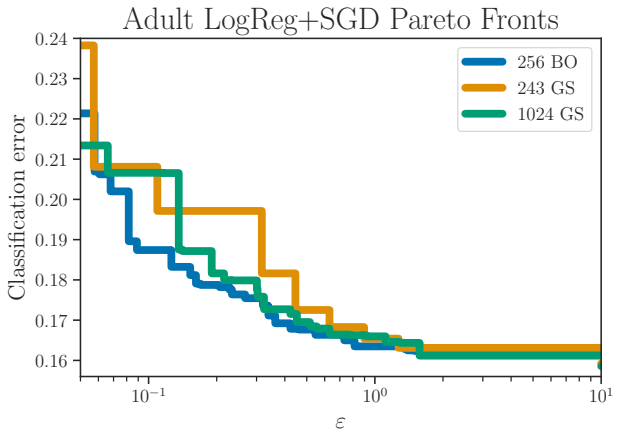

Fig. 6. Grid search experiment results compared with the Bayesian optimization approach used in DPareto.

Variability of Pareto Front

DPareto also allows us to gather information about the potential variability of the recovered Pareto front. In order to do that, recall that in our experiments we implemented the utility oracle by repeatedly running algorithm A_λ with a fixed choice of hyperparameters, and then reported the average utility across runs. Using these same runs we can also take the best and worst utilities observed for each choice of hyperparameters. Fig. 7 displays the Pareto fronts recovered from considering the best and worst runs in addition to the Pareto front obtained from the average over runs. In general we observe higher variability in utility on the high privacy regime (i.e. small ε), which is to be expected since more privacy is achieved by increasing the variance of the noise added to the computation. These type of plots can be useful to decision-makers who want to get an idea of what variability can be expected in practice from a particular choice of hyperparameters.

DPareto's Versatility

The other main purpose of these experiments is to demonstrate the versatility of DPareto by comparing multiple approaches to the same problem. In Fig. 8, the left plot shows Pareto fronts of the adult dataset for multiple optimizers (SGD and Adam) as well as multiple models (LogReg and SVM), and the right plot shows Pareto fronts of the mnist dataset for different architectures (MLP1 and MLP2). With this, we can see that on the adult dataset, the LogReg model optimized using Adam was nearly always better than the other model/optimizer combinations. We can also see that on the mnist dataset, while both architectures performed similarly in the low-privacy regime, the MLP2 architecture significantly outperformed the MLP1 architecture in the high-privacy regime. With DPareto, analysts and practitioners can efficiently create these types of Pareto fronts and use them to perform privacy–utility trade-off comparisons.

Computational Overhead of DPareto

Although it is clear that DPareto more-efficiently produces high-quality Pareto fronts relative to random sampling and grid search, we must examine the computational cost it incurs. Namely, we are interested in the running time of DPareto, excluding the model training time. Therefore, for the BO experiments on both datasets, we measured the time it took for DPareto to: 1) initialize the GP models with the 16 seed points, plus 2) iteratively propose the subsequent 256 hyperparameters and incorporate their corresponding privacy and utility results.

For both the adult and mnist datasets, despite the difference in hyperparameter domains as well as the privacy and utility values that were observed, DPareto's overhead remained fairly consistent at approximately 45 seconds of total wall-clock time. This represents a negligible fraction of the total Pareto front computation

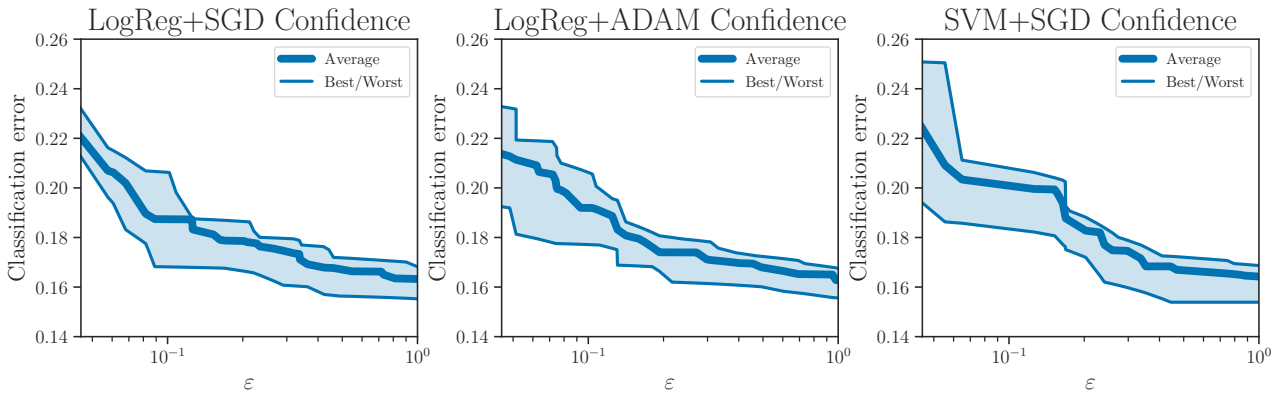

Fig. 7. Variability of estimated Pareto fronts across models and optimizers for ADULT.

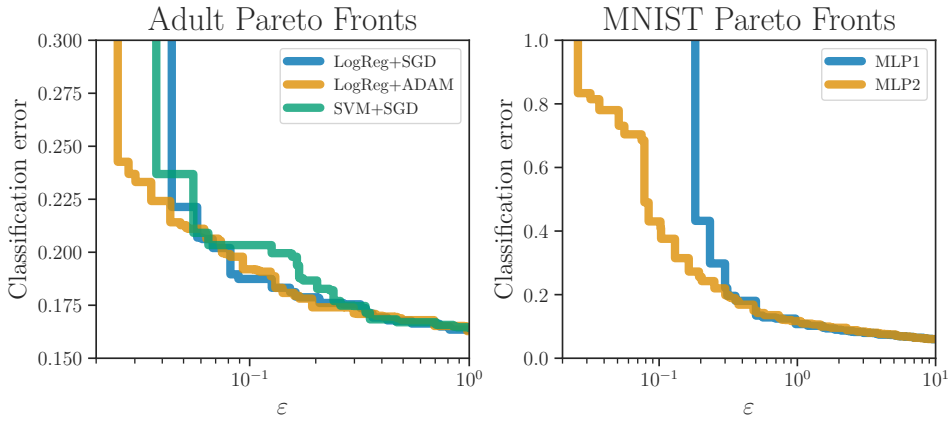

Fig. 8. *Left:* Pareto fronts for combinations of models and optimizers on the ADULT dataset. *Right:* Pareto fronts for different MLP architectures on the MNIST dataset.

time for either dataset; specifically, less than 0.1% of the total time for the ADULT Pareto fronts, and less than 0.01% for the MNIST Pareto fronts. Thus, we conclude that DPARETO's negligible overhead is more than offset by its improved Pareto fronts.

We remark here that although the overhead is negligible, DPARETO does have a shortcoming relative to traditional methods: it is an inherently sequential process which cannot be easily parallelized. On the other hand, random search and grid search can be trivially parallelized to an arbitrarily high degree bounded only by one's compute resources. Improving upon this facet of DPARETO is beyond the scope of this work, however we briefly discuss it as a possible extension in the following section.

5 Extensions

There are several extensions and open problems whose solutions would enhance DPARETO's usefulness for practical applications.

The first open problem is on the privacy side. As designed, DPARETO is a system to *non-privately* estimate the Pareto front of differentially private algorithms. One challenging open problem is how to tightly characterize the privacy guarantee of the estimated Pareto front itself. This involves analyzing the privacy guarantees for both the training and test data sets. Naïvely applying DP composition theorems immediately provides conservative bounds on the privacy for both sets (assuming a small amount of noise is added to the output of the utility oracle). This follows from observing that individual points evaluated by DPARETO enjoy the DP guarantees computed by the privacy oracle, and the rest of the algorithm only involves post-processing and adaptive composition. However, these bounds would be prohibitively large for practical use; we expect a more advanced analy-

sis to yield significantly tighter guarantees since for each point we are only releasing its utility and not the complete trained model. For a decision-maker, these would provide an end-to-end privacy guarantee for the entire process, and allow the Pareto front to be made publicly available.

The other open problem is on the BO side. Recall that the estimated Pareto front contains only the privacy–utility values of the trained models, along with their corresponding hyperparameters. In practice, a decision-maker may be interested in finding a hyperparameter setting that induces a particular point on the estimated Pareto front but which was not previously tested by DPareto. The open problem here is how to design a computationally efficient method to extract this information from DPareto's GPs.

On the theoretical side, it would also be interesting to obtain bounds on the speed of convergence of DPareto to the optimal Pareto front. Results of this kind are known for *single-objective* BO under smoothness assumptions on the objective function (see, e.g., [9]). Much less is known in the *multi-objective* case, especially in a setting like ours that involves continuous hyper-parameters and noisy observations (from the utility oracle). Nonetheless, the smoothness we observe on the privacy and utility oracles as a function of the hyperparameters, and our empirical evaluation against grid and random sampling, suggests similar guarantees could hold for DPareto.

For applications of DPareto to concrete problems, there are several interesting extensions. We focused on supervised learning, but the method could also be applied to e.g. stochastic variational inference on probabilistic models, as long as a utility function (e.g. held-out perplexity) is available. DPareto currently uses independent GPs with fixed transformations, but an interesting extension would be to use warped multi-output GPs. It may be of interest to optimize over additional criteria, such as model size, training time, or a fairness measure. If the sequential nature of BO is prohibitively slow for the problem at hand, then adapting the recent advances in batch multi-objective BO to DPareto would enable parallel evaluation of several candidate models [20, 34, 47]. Finally, while we explored the effect of changing the model (logistic regression vs. SVM) and the optimizer (SGD vs. Adam) on the privacy–utility trade-off, it would be interesting to optimize over these choices as well.

6 Related Work

While this work is the first to examine the privacy–utility trade-off of differentially private algorithms using multi-objective optimization and Pareto fronts, efficiently computing Pareto fronts without regards to privacy is an active area of research in fields relating to multi-objective optimization. DPareto's point-selection process aligns with Couckuyt et al. [13], but other approaches may provide promising alternatives for improving DPareto. For example, Zuluaga et al. [50] propose an acquisition criteria that focuses on uniform approximation of the Pareto front instead of a hypervolume based criteria. Note that their technique does not apply out-of-the-box to the problems we consider in our experiments since it assumes a discrete hyperparameter space.

The threat model and outputs of the DPareto algorithm are closely aligned with the methodology used by the U.S. Census Bureau to choose the privacy parameter ε for their deployment of DP to release data from the upcoming 2020 census. In particular, the bureau is combining a graphical approach to represent the privacy–utility trade-off for their application [19] together with economic theory to pick a particular point to balance the trade-off [3]. Their graphical approach works with Pareto fronts identical to the ones computed by our algorithm, which they construct using data from previous censuses [2]. We are not aware of the specifics of their hyperparameter tuning, but, from what has transpired, it would seem that the gross of hyperparameters in their algorithms is related to the post-processing step and therefore only affects utility[10].

Several aspects of this paper are related to recent work in single-objective optimization. For non-private single-objective optimization, there is an abundance of recent work in machine learning on hyperparameter selection, using BO [23, 26] or other methods [32] to maximize a model's utility. Recently, several related questions at the intersection of machine learning and differential privacy have emerged regarding hyperparameter selection and utility.

One such question explicitly asks how to perform the hyperparameter-tuning process in a privacy-preserving way. Kusner et al. [30] and subsequently Smith et al. [43] use BO to find near-optimal hyperpa-

10 Or, in the case of invariant forcing, privacy effects not quantifiable within standard DP theory.

rameter settings for a given model while preserving the privacy of the data during the utility evaluation stage. Aside from the single-objective focus of this setting, our case is significantly different in that we are primarily interested in *training* the models with differential privacy, not in protecting the privacy of the data used to evaluate an already-trained model.

Another question asks how to choose utility-maximizing hyperparameters when privately training models. When privacy is independent of the hyperparameters, this reduces to the non-private hyperparameter optimization task. However, two variants of this question do not have this trivial reduction. The first variant inverts the stated objective to study the problem of maximizing privacy given constraints on the final utility [21, 33]. The second variant, closely aligning with this paper's setting, studies the problem of choosing utility-maximizing, but privacy-dependent, hyperparameters. This is particularly challenging, since the privacy's dependence on the hyperparameters may be non-analytical and computationally expensive to determine. Approaches to this variant have been studied [37, 46], however the proposed strategies are 1) based on heuristics, 2) only applicable to the differentially private SGD problem, and 3) do not provide a computationally efficient way to find the Pareto optimal points for the privacy–utility trade-off of a given model. Wu et al. [49] provide a practical analysis-backed approach to privately training utility-maximizing models (again, for the case of SGD with a fixed privacy constraint), but hyperparameter optimization is naïvely performed using grid-search. By contrast, this paper provides a computationally efficient way to *directly* search for Pareto optimal points for the privacy–utility trade-off of arbitrary hyperparameterized algorithms.

The final related question revolves around the differentially private "selection" or "maximization" problem [11], which asks: how can an item be chosen (from a predefined universe) to maximize a data-dependent function while still protecting the privacy of that data? Here, Liu et al. [35] recently provided a way to choose hyperparameters that approximately maximize the utility of a given differentially private model in a way that protects the privacy of both the training and test data sets. However, this only optimizes utility with fixed privacy – it does not address our problem of directly optimizing for the selection of hyperparameters that generate privacy–utility points which fall on the Pareto front.

Recent work on data-driven algorithm configuration has considered the problem of tuning the hyperparameters of combinatorial optimization algorithms while maintaining DP [5]. The setting considered in [5] assumes there is an underlying distribution of problem instances, and a sample from this distribution is used to select hyperparameters that will have good computational performance on future problem instances sampled from the same distribution. In this case, the authors consider a threat model where the whole sample of problem instances used to tune the algorithm needs to be protected. A similar problem of data-driven algorithm selection has been considered, where the problem is to choose the best algorithm to accomplish a given task while maintaining the privacy of the data used [29]. For both, only the utility objective is being optimized, assuming a fixed constraint on the privacy.

7 Conclusion

In this paper, we characterized the privacy–utility trade-off of a hyperparameterized algorithm as a Pareto front learning problem.We then introduced DPareto, a method to empirically learn a differentially private algorithm's Pareto front. DPareto uses Bayesian optimization (BO), a state-of-the-art method for hyperparameter optimization, to efficiently form the Pareto front by simultaneously optimizing for both privacy and utility. We evaluated DPareto across various datasets, models, and differentially private optimization algorithms, demonstrating its efficiency and versatility. Further, we showed that BO allows us to construct useful visualizations to aid the decision making process.

Acknowledgments

This work was funded by Amazon Research Cambridge.

References

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

[2] John M. Abowd. Disclosure avoidance for block level data and protection of confidentiality in public tabulations. Census Scientific Advisory Committee (Fall Meeting), 2018.

[3] John M Abowd and Ian M Schmutte. An economic analysis of privacy protection and statistical accuracy as social choices. *American Economic Review*, pages 171–202, 2019.

[4] Mauricio A. Álvarez, Lorenzo Rosasco, and Neil D. Lawrence. Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning*, 4(3):195–266, March 2012.

[5] Maria-Florina Balcan, Travis Dick, and Ellen Vitercik. Dispersion for data-driven algorithm design, online learning, and private optimization. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 603–614. IEEE, 2018.

[6] Borja Balle, Gilles Barthe, and Marco Gaboardi. Privacy amplification by subsampling: Tight analyses via couplings and divergences. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, 2018.

[7] Borja Balle and Yu-Xiang Wang. Improving the Gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 403–412, 2018.

[8] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[9] Adam D Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12(Oct):2879–2904, 2011.

[10] Nicholas Carlini, Chang Liu, Jernej Kos, Úlfar Erlingsson, and Dawn Song. The secret sharer: Measuring unintended neural network memorization & extracting secrets. In *Proceedings of the 27th USENIX Security Symposium*, 2018.

[11] Kamalika Chaudhuri, Daniel J Hsu, and Shuang Song. The large margin mechanism for differentially private maximization. In *Advances in Neural Information Processing Systems*, pages 1287–1295, 2014.

[12] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

[13] Ivo Couckuyt, Dirk Deschrijver, and Tom Dhaene. Fast calculation of multiobjective probability of improvement and expected improvement criteria for Pareto optimization. *Journal of Global Optimization*, 60(3):575–594, 2014.

[14] Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 109, pages 1–12, 2006.

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

[16] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, and Salil Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 381–390, 2009.

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

[18] Vitaly Feldman, Ilya Mironov, Kunal Talwar, and Abhradeep Thakurta. Privacy amplification by iteration. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, 2018.

[19] Simson L Garfinkel, John M Abowd, and Sarah Powazek. Issues encountered deploying differential privacy. In *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*, 2018.

[20] David Gaudrie, Rodolphe Le Riche, Victor Picheny, Benoit Enaux, and Vincent Herbert. Targeting solutions in bayesian multi-objective optimization: sequential and batch versions. *Annals of Mathematics and Artificial Intelligence*, 88(1):187–212, 2020.

[21] Chang Ge, Xi He, Ihab F Ilyas, and Ashwin Machanavajjhala. Apex: Accuracy-aware differentially private data exploration. In *Proceedings of the 2019 International Conference on Management of Data*. ACM, 2019.

[22] Joseph Geumlek, Shuang Song, and Kamalika Chaudhuri. Rényi differential privacy mechanisms for posterior sampling. In *Advances in Neural Information Processing Systems*, pages 5289–5298, 2017.

[23] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1487–1495. ACM, 2017.

[24] Rodolphe Jenatton, Cedric Archambeau, Javier González, and Matthias Seeger. Bayesian optimization with tree-structured dependencies. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1655–1664, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015.

[26] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 528–536, Fort Lauderdale, FL, USA, 20–22 Apr 2017. PMLR.

[27] Nicolas Knudde, Joachim van der Herten, Tom Dhaene, and Ivo Couckuyt. GPflowOpt: A Bayesian Optimization Library using TensorFlow. *arXiv preprint – arXiv:1711.03845*, 2017.

[28] Ron Kohavi. Scaling up the accuracy of Naïve-Bayes classifiers: a decision-tree hybrid. In *KDD*, volume 96, pages 202–207. Citeseer, 1996.

[29] Ios Kotsogiannis, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. Pythia: Data dependent differentially private algorithm selection. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1323–1337. ACM, 2017.

[30] Matt Kusner, Jacob Gardner, Roman Garnett, and Kilian Weinberger. Differentially private Bayesian optimization. In *International Conference on Machine Learning*, pages 918–927, 2015.

[31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[32] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.*, 18(1):6765–6816, January 2017.

[33] Katrina Ligett, Seth Neel, Aaron Roth, Bo Waggoner, and Steven Z Wu. Accuracy first: Selecting a differential privacy level for accuracy constrained ERM. In *Advances in Neural Information Processing Systems*, pages 2566–2576, 2017.

[34] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qingfu Zhang, and Sam Kwong. A batched scalable multi-objective bayesian optimization algorithm. *arXiv preprint arXiv:1811.01323*, 2018.

[35] Jingcheng Liu and Kunal Talwar. Private selection from private candidates. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 298–309, New York, NY, USA, 2019. ACM.

[36] Min Lyu, Dong Su, and Ninghui Li. Understanding the sparse vector technique for differential privacy. *Proceedings of the VLDB Endowment*, 2017.

[37] H. Brendan McMahan and Galen Andrew. A general approach to adding differential privacy to iterative training procedures. In *NeurIPS 2018 workshop on Privacy Preserving Machine Learning*, 2018.

[38] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *International Conference on Learning Representations*, 2018.

[39] Ilya Mironov. Rényi differential privacy. In *Computer Security Foundations Symposium (CSF), 2017 IEEE 30th*, pages 263–275. IEEE, 2017.

[40] J Močkus. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404. Springer, 1975.

[41] Kobbi Nissim, Thomas Steinke, Alexandra Wood, Micah Altman, Aaron Bembenek, Mark Bun, Marco Gaboardi, David O'Brien, and Salil Vadhan. Differential privacy: A primer for a non-technical audience (preliminary version). *Vanderbilt Journal of Entertainment and Technology Law*, 2018.

[42] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[43] Michael Smith, Mauricio Álvarez, Max Zwiessele, and Neil Lawrence. Differentially private regression with Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 1195–1203, 2018.

[44] Edward Snelson, Zoubin Ghahramani, and Carl E Rasmussen. Warped Gaussian processes. In *Advances in neural information processing systems*, pages 337–344, 2004.

[45] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[46] Koen Lennart van der Veen. A practical approach to differential private learning. Master's thesis, University of Amsterdam, The Netherlands, 2018.

[47] Hongyan Wang, Hua Xu, Yuan Yuan, Xiaomin Sun, and Junhui Deng. Balancing exploration and exploitation in multiobjective batch bayesian optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 237–238, 2019.

[48] Yu-Xiang Wang, Borja Balle, and Shiva Kasiviswanathan. Subsampled Rényi differential privacy and analytical moments accountant. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

[49] Xi Wu, Fengan Li, Arun Kumar, Kamalika Chaudhuri, Somesh Jha, and Jeffrey Naughton. Bolt-on differential privacy for scalable stochastic gradient descent-based analytics. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1307–1322. ACM, 2017.

[50] Marcela Zuluaga, Andreas Krause, and Markus Püschel. ε-pal: an active learning approach to the multi-objective optimization problem. *The Journal of Machine Learning Research*, 17(1):3619–3650, 2016.

A Implementation Details

Hyperparameter optimization was done using the GPFlowOpt library [27] which offers GP-based Bayesian optimization. In particular, we used the provided GP regression functionality with a Matérn 5/2 kernel, as well as the provided HVPoI acquisition function. Machine learning models used in the paper are implemented with Apache MXNet [12]. We have made use of the high-level Gluon API whenever possible. However, the privacy accountant implementation that we used (see [48]) required low-level changes to the definitions of the models. In order to keep the continuous MXNet execution graph to ensure a fast evaluation of the model, we reverted to the pure MXNet model definitions. Even though this approach requires much more effort to implement the models themselves, it allows for more fine-grained control of how the model is executed, as well as provides a natural way of implementing privacy accounting.

B Privacy Proof for Sparse Vector Technique

Here we provide a proof of the privacy bound for Alg. 1 used to implement the privacy oracle P_0. The proof is based on observing that our Alg. 1 is just a simple reparametrization of [36, Alg. 7] where some of the parameters have been fixed up-front. For concreteness, we reproduce [36, Alg. 7] as Alg. 5 below. The result then

follows from a direct application of [36, Thm. 4], which shows that Alg. 5 is $(\varepsilon_1 + \varepsilon_2, 0)$-DP.

Algorithm 5: Sparse Vector Technique ([36, Alg. 7] with $\varepsilon_3 = 0$)

Input: dataset z, queries $q_1, \ldots, q_m$, sensitivity Δ

Hyperparameters: bound C, thresholds $T_1, \ldots, T_m$, privacy parameters ε_1, ε_2

$c \leftarrow 0$, $w \leftarrow (\bot, \ldots, \bot) \in \{\bot, \top\}^m$

$\rho \leftarrow \mathsf{Lap}(\Delta/\varepsilon_1)$

for $i \in [m]$ **do**

  $\nu \leftarrow \mathsf{Lap}(2C\Delta/\varepsilon_2)$

  if $q_i(z) + \nu \geq T_i + \rho$ **then**

    $w_i \leftarrow \top$, $c \leftarrow c + 1$

    if $c \geq C$ **then return** w

return w

Comparing Alg. 5 with the sparse vector technique in Alg. 1, we see that they are virtually the same algorithms, where we have fixed $\Delta = 1$, $T_i = 1/2$, $\varepsilon_1 = 1/b_1$ and $\varepsilon_2 = 2C/b_2$. Thus, by expanding the definitions of b_1 and b_2 as a function of b and C, we can verify that Alg. 1 is $(\varepsilon, 0)$-DP with

$$\begin{aligned}\varepsilon &= \varepsilon_1 + \varepsilon_2 \\ &= \frac{1}{b_1} + \frac{2C}{b_2} \\ &= \frac{1 + (2C)^{1/3}}{b} + \frac{(2C)^{2/3}(1 + 2C)^{1/3}}{b} \\ &= \frac{(1 + (2C)^{1/3})(1 + (2C)^{2/3})}{b}.\end{aligned}$$

This concludes the proof.

C Random Sampling Distributions

Tables 3 and 4 list the distributions for the hyperparameters used in the MNIST and ADULT experiments respectively.

Hyperparameter	Distribution	Parameters	Int-Valued	Accept Range
Epochs	Uniform	$a = 1, b = 400$	True	$[1, 400]$
Lot Size	Normal	$\mu = 800, \sigma = 800$	True	$[16, 4000]$
Learning Rate	Shifted Exp.	$\lambda = 10, \text{shift} = 1\text{e}{-3}$	False	$[1\text{e}{-3}, 5\text{e}{-1}]$
Noise Variance	Shifted Exp.	$\lambda = 5\text{e}{-1}, \text{shift} = 1\text{e}{-1}$	False	$[1\text{e}{-1}, 16]$
Clipping Norm	Shifted Exp.	$\lambda = 5\text{e}{-1}, \text{shift} = 1\text{e}{-1}$	False	$[1\text{e}{-1}, 12]$

Table 3. MNIST random sampling distributions.

Hyperparameter	Distribution	Parameters	Int-Valued	Accept Range
Epochs	Uniform	$a = 1, b = 64$	True	$[1, 64]$
Lot Size	Normal	$\mu = 128, \sigma = 64$	True	$[8, 512]$
Learning Rate	Shifted Exp.	$\lambda = 10, \text{shift} = 1\text{e}{-3}$	False	$[1\text{e}{-3}, 1\text{e}{-1}]$
Noise Variance	Shifted Exp.	$\lambda = 1\text{e}{-1}, \text{shift} = 1\text{e}{-1}$	False	$[1\text{e}{-1}, 16]$
Clipping Norm	Shifted Exp.	$\lambda = 1\text{e}{-1}, \text{shift} = 1\text{e}{-1}$	False	$[1\text{e}{-1}, 4]$

Table 4. ADULT random sampling distributions.