FLoRA: Single-shot Hyper-parameter Optimization for Federated Learning

Yi Zhou **Parikshit Ram** **Theodoros Salonidis**

Nathalie Baracaldo **Horst Samulowitz** **Heiko Ludwig**
IBM Research
{yi.zhou, Parikshit.Ram}@ibm.com
{baracald, tsaloni, samulowitz, hludwig}@us.ibm.com

Abstract

We address the relatively unexplored problem of hyper-parameter optimization (HPO) for federated learning (FL-HPO). We introduce **F**ederated **Lo**ss Su**R**face **A**ggregation (FLoRA), the first FL-HPO solution framework that can address use cases of tabular data and gradient boosting training algorithms in addition to stochastic gradient descent/neural networks commonly addressed in the FL literature. The framework enables single-shot FL-HPO, by first identifying a good set of hyper-parameters that are used in a *single* FL training. Thus, it enables FL-HPO solutions with minimal additional communication overhead compared to FL training without HPO. Our empirical evaluation of FLoRA for Gradient Boosted Decision Trees on seven OpenML data sets demonstrates significant model accuracy improvements over the considered baseline, and robustness to increasing number of parties involved in FL-HPO training.

1 Introduction

Traditional machine learning (ML) approaches require training data to be gathered at a central location where the learning algorithm runs. In real world scenarios, however, training data is often subject to privacy or regulatory constraints restricting the way data can be shared, used and transmitted. Examples of such regulations include the European General Data Protection Regulation (GDPR), California Consumer Privacy Act (CCPA), Cybersecurity Law of China (CLA) and HIPAA, among others. Federated learning (FL), first proposed in [1], has recently become a popular approach to address privacy concerns by allowing collaborative training of ML models among multiple parties where each party can keep its data private.

FL-HPO problem. Despite the privacy protection FL brings along, there are many open problems in FL domain [2, 3], one of which is hyper-parameter optimization for FL. Existing FL systems require a user (or all participating parties) to pre-set (agree on) multiple hyper-parameters (HPs) (i) for the model being trained (such as number of layers and batch size for neural networks or tree depth and number of trees in tree ensembles), and (ii) for the the aggregator (if such hyper-parameters exist). Hyper-parameter optimization (HPO) for FL is important because the choice of HPs can have dramatic impact on performance. This is particularly important for tabular data (where datasets can be radically different from each other) as well as image data and neural nets.

While HPO has been widely studied in the centralized ML setting, it comes with unique challenges in the FL setting. First, existing HPO techniques for centralized training often make use of the entire data set, which is not available in FL. Secondly, they train a vast variety of models for a large number of HP configurations which would be prohibitively expensive in terms of communication and training time in FL settings. Thirdly, one important challenge that has not been adequately explored in FL literature is support for tabular data, which are widely used in enterprise settings. One of the best models for this setting are based on gradient boosting tree algorithms which are

1st NeurIPS Workshop on New Frontiers in Federated Learning (NFFL 2021), Virtual Meeting.

not based on the stochastic gradient descent training algorithm used for neural networks. Recently, a few approaches have been proposed for FL-HPO, however they focus on handling HPO using personalization techniques [3] and neural networks [4]. To the best of our knowledge, there is no HPO approach for FL systems to train non-neural network models, such as XGBoost that are particularly common in the enterprise setting.

Scope. In this paper, we address the aforementioned challenges of FL-HPO. We focus on the problem where the model HPs are shared across all parties and we seek a set of HPs and train a single model that is eventually used by all parties for testing/deployment. Moreover, we impose three further requirements that make the problem more challenging: (**C1**) we *do not make any assumption* that two models with different HPs can perform some form of "weight-sharing" (which is a common technique used in various HPO and neural architecture search (NAS) schemes for neural networks to reduce the computational overhead of HPO and NAS), allowing our solution to be applied beyond neural networks [4]. (**C2**) we seek to **perform "single-shot" FL-HPO**, where we have *limited* resources (in the form of computation and communication overhead) which allow training only a single model via federated learning (that is, a single HP configuration), and (**C3**) we *do not assume that parties have independent and identically distributed (IID) data distributions*.

Contributions. Given the above FL-HPO problem setting, we make the following contributions:

- (§3) We present a novel framework **F***ederated* **Lo***ss Su***R***face* **A***ggregation* (FLoRA) that leverages meta-learning techniques to utilize **local and asynchronous** HPO on each party to perform single-shot HPO for the global FL-HPO problem.
- (§4) We evaluate FLoRA on the FL-HPO of Gradient Boosted Decision Trees (GBDTs) [5] on seven classification datasets from OpenML [6], highlighting (i) its performance relative to the baseline, (ii) the effect of various choices in this scheme, (iii) the effect of the number of parties on the performance, and (iv) finally, we empirically demonstrate that FLoRA works under high heterogeneous (Non-IIDness) parties distributions.

2 Related work

Despite fruitful research results on FL regarding advanced learning algorithms [7, 8], data heterogeneity [9, 10], personalization [11–14], fairness [15, 16], system design [17–19] and privacy-preserving frameworks[20–22] etc., there are only a few focusing on hyper-parameters tuning for FL [23, 4, 24–26, 3]. In [23], Dai et.al. address Federated Bayesian Optimization. The problem setup is quite different than FL-HPO in that they focus on a single party using information from other parties to accelerate its own Bayesian Optimization. The works in [24–26] consider adaptation of learning rate in SGD-based FL training. Inspired from the neural architecture search technique of weight-sharing, [4, 3] proposed FedEx, a FL-HPO framework to accelerate a general hyper-parameter tuning procedure, i.e., successive halving algorithm (SHA), for many SGD-based FL algorithms. In contrast to these approaches, our framework is also applicable to non SGD-training settings and minimizes HPO overhead by being a one-shot approach.

While HPO has been widely studied in the centralized ML setting, it comes with unique challenges in the FL setting. First, existing HPO techniques for centralized training often make use of the entire data set, which is not available in FL. Secondly, they train a vast variety of models for a large number of HP configurations which would be prohibitively expensive in terms of communication and tra Beyond grid-search for HPO, random search is a very competitive baseline because of its simplicity and parallelizability [27]. Sequential model-based optimization (SMBO) [28] is a common technique with different 'surrogate models' such as Gaussian processes [29], random forests [30], radial basis functions [31], and tree-parzen estimators [32]. However, black-box optimization is a time consuming process because the expensive black-box function evaluation involves model training and scoring (on a held-out set). Efficient *multi-fidelity* approximations of the black-box function based on some budget (training samples/epochs) combined with bandit learning can skip unpromising candidates early via successive halving [33, 34] and HyperBand [35]. However, these schemes essentially perform an efficient random search and are well suited for search over discrete spaces or discretized continuous spaces. BOHB [36] combines SMBO (with TPE) and HyperBand for improved optimization. The problem of HPO has been extended from ML model configurations to the configuration of complete ML pipelines – the Combined Algorithm Selection and HPO (or CASH) problem – with many increasingly efficient algorithms [37–41]. All these techniques rely on multiple (partial or full) trainings of models with different HP configurations, and hence are not

practical for the single-shot FL-HPO problem.
Instead of starting the HPO from scratch for every dataset (which is usually a very expensive process), meta-learning can be used to refine the search space and warm start the search [42]. The usual techniques involve "meta-features" for data sets that are used to develop a notion of similarity between data sets. This is then used for any new data set (on which HPO needs to be performed) to identify similar previously processed data sets [38, 43]. The HPO on previously processed data sets are also utilized to (i) generate promising initial HPs for the HPO on any new data set [44, 45], and/or (ii) prune the HPO search space, removing unpromising areas, allowing the HPO solver to focus its attention on useful parts of the search space [46, 47]. For the purposes of FL-HPO, we can view the per-party data sets as "similar" data sets, even though we do not assume them to be IID, and explore how the meta-learning technique of learning good HPO initialization as a potential way of tackling single-shot FL-HPO.

3 Methodology

In the centralized ML setting, we would consider a model class $\mathcal{M}$ and its corresponding learning algorithm $\mathcal{A}$ parameterized collectively with HPs $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, and given a training set D, we can learn a single model $\mathcal{A}(\mathcal{M}, \boldsymbol{\theta}, D) \rightarrow m \in \mathcal{M}$. Given some predictive loss $\mathcal{L}(m, D')$ of any model m scored on some holdout set D', the centralized HPO problem can be stated as

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \mathcal{L}(\mathcal{A}(\mathcal{M}, \boldsymbol{\theta}, D), D'). \tag{1}$$

In the most general FL setting, we have p parties $P_1, \ldots, P_p$ each with their private local training data set $D_i, i \in [p]$. Let $D = \cup_{i=1}^{p} D_i$ denote the aggregated training data set and $\overline{D} = \{D_i\}_{i \in [p]}$ denote the set of per-party data sets. Each model class (and corresponding learning algorithm) is parameterized by global HPs $\boldsymbol{\theta}_G \in \boldsymbol{\Theta}_G$ shared by all parties and per-party local HPs $\boldsymbol{\theta}_L^{(i)} \in \boldsymbol{\Theta}_L, i \in [p]$ with $\boldsymbol{\Theta} = \boldsymbol{\Theta}_G \times \boldsymbol{\Theta}_L$. FL systems usually include an aggregator with its own set of HPs $\boldsymbol{\phi} \in \boldsymbol{\Phi}$. Finally, we would have a FL algorithm $\mathcal{F}\left(\mathcal{M}, \boldsymbol{\phi}, \boldsymbol{\theta}_G, \{\boldsymbol{\theta}_L^{(i)}\}_{i \in [p]}, \mathcal{A}, \overline{D}\right) \rightarrow m \in \mathcal{M}$ that takes as input all the relevant HPs and per-party data sets and generates a model. In this case, the FL-HPO problem can be stated in the two following ways depending on the desired goals: (i) For a global holdout data set D' (a.k.a validation set, possibly from the same distribution as the aggregated data set D), we solve the following problem:

$$\min_{\boldsymbol{\phi} \in \boldsymbol{\Phi}, \boldsymbol{\theta}_G \in \boldsymbol{\Theta}_G, \boldsymbol{\theta}_L^{(i)} \in \boldsymbol{\Theta}_L, i \in [p]} \mathcal{L}\left(\mathcal{F}\left(\mathcal{M}, \boldsymbol{\phi}, \boldsymbol{\theta}_G, \{\boldsymbol{\theta}_L^{(i)}\}_{i \in [p]}, \mathcal{A}, \overline{D}\right), D'\right). \tag{2}$$

(ii) An alternative problem would involve per-party holdout data sets $D_i', i \in [p]$ and we solve the following problem:

$$\min_{\boldsymbol{\phi} \in \boldsymbol{\Phi}, \boldsymbol{\theta}_G \in \boldsymbol{\Theta}_G, \boldsymbol{\theta}_L^{(i)} \in \boldsymbol{\Theta}_L, i \in [p]} \mathsf{Agg}\left(\left\{\mathcal{L}\left(\mathcal{F}\left(\mathcal{M}, \boldsymbol{\phi}, \boldsymbol{\theta}_G, \{\boldsymbol{\theta}_L^{(i)}\}_{i \in [p]}, \mathcal{A}, \overline{D}\right), D_i'\right), i \in [p]\right\}\right), \tag{3}$$

where $\mathsf{Agg} : \mathbb{R}^p \rightarrow \mathbb{R}$ is some aggregation function (such as average or maximum) that scalarizes the p per-party predictive losses.

Contrasting problem (1) to problems (2) & (3), we can see that the FL-HPO is significantly more complicated than the centralized HPO problem. In the ensuing presentation, we focus on problem (2) although our proposed single-shot FL-HPO scheme can be applied and evaluated for problem (3). We simplify the FL-HPO problem in the following ways: (i) we assume that there is no personalization so there are no per-party local HPs $\boldsymbol{\theta}_L^{(i)}, i \in [p]$, and (ii) we only focus on the model class HPs $\boldsymbol{\theta}_G$, deferring HPO for aggregator HPs $\boldsymbol{\phi}$ for future work. Hence the problem we will study is stated as for a fixed aggregator HP $\boldsymbol{\phi}$:

$$\min_{\boldsymbol{\theta}_G \in \boldsymbol{\Theta}_G} \mathcal{L}\left(\mathcal{F}\left(\mathcal{M}, \boldsymbol{\phi}, \boldsymbol{\theta}_G, \mathcal{A}, \overline{D}\right), D'\right). \tag{4}$$

This problem appears similar to the centralized HPO problem (1). However, note that the main challenge in (4) is the need for a federated training for each set of HPs $\boldsymbol{\theta}_G$, and hence it is not practical (from a communication overhead perspective) to apply existing off-the-shelf HPO schemes to problem (4). In the subsequent discussion, for simplicity purposes, we will use $\boldsymbol{\theta}$ to denote the global HPs, dropping the "G" subscript.

3.1 Leveraging local HPOs

While it is impractical to apply off-the-shelf HPO solvers (such as Bayesian Optimization (BO) [28], Hyperopt [32], SMAC [30], and such), we wish to understand how we can leverage local and asynchronous HPOs in each of the parties. We begin with a simple but intuitive hypothesis underlying various meta-learning schemes for HPO [42, 48]: *if a HP configuration $\boldsymbol{\theta}$ has good performance for all parties independently, then $\boldsymbol{\theta}$ is a strong candidate for federated training*.

Algorithm 1: Single-shot FL-HPO with Federated Loss Surface Aggregation

1 **FLoRA**$(\boldsymbol{\Theta}, \mathcal{M}, \mathcal{A}, \{(D_i, D'_i)\}_{i\in[p]}, T) \to m$
2 **for** *each party* $P_i, i \in [p]$ **do**
3 Run HPO to generate T (HP, loss) pairs

$$E^{(i)} = \left\{(\boldsymbol{\theta}_t^{(i)}, \mathcal{L}_t^{(i)}), t \in [T], \boldsymbol{\theta}_t^{(i)} \in \boldsymbol{\Theta}, \mathcal{L}_t^{(i)} := \mathcal{L}(\mathcal{A}(\mathcal{M}, \boldsymbol{\theta}_t^{(i)}, D_i), D'_i)\right\} \tag{5}$$

4 **end**
5 Collect all $E^{(i)}, i \in [p]$ in aggregator
6 Generate a unified loss surface $\ell : \boldsymbol{\Theta} \to \mathbb{R}$ using $\left\{E^{(i)}, i \in [p]\right\}$
7 Select best HP candidate $\boldsymbol{\theta}^\star \leftarrow \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \ell(\boldsymbol{\theta})$
8 Learn final model with federated training: $m \leftarrow \mathcal{F}(\mathcal{M}, \boldsymbol{\phi}, \boldsymbol{\theta}^\star, \mathcal{A}, \overline{D})$
9 **return** m
10 **end**

With this hypothesis, we present our proposed algorithm **FLoRA** in Algorithm 1. In this scheme, we allow each party to perform HPO locally and asynchronously with some adaptive HPO scheme such as BO (line 3). Then, at each party $i \in [p]$, we collect all the attempted T HPs $\boldsymbol{\theta}_t^{(i)}, t \in [T]$ and their corresponding predictive loss $\mathcal{L}_t^{(i)}$ into a set $E^{(i)}$ (line 3, equation (5)). Then these per-party sets of (HP, loss) pairs $E^{(i)}$ are collected at the aggregator (line 5). This operation has at most $O(pT)$ communication overhead (note that the number of HPs are usually much smaller than the number of columns or number of rows in the per-party data sets). These sets are then used to generate an aggregated loss surface $\ell : \boldsymbol{\Theta} \to \mathbb{R}$ (line 6) which will then be used to make the final single-shot HP recommendation $\boldsymbol{\theta}^\star \in \boldsymbol{\Theta}$ (line 7) for the federated training to create the final model $m \in \mathcal{M}$ (line 8). We will discuss the generation of the aggregated loss surface in detail in §3.2. Before that, we briefly want to discuss the motivation behind some of our choices in Algorithm 1.

Why adaptive HPO? The reason to use adaptive HPO schemes instead of non-adaptive schemes such as random search or grid search is that this allows us to efficiently approximate the local loss surface more accurately (and with more certainty) in regions of the HP space where the local performance is favorable instead of trying to approximate the loss surface well over the complete HP space. This has advantages both in terms of computational efficiency and loss surface approximation.

Why asynchronous HPO? Each party executes HPO asynchronously, without coordination with HPO results from other parties or with the aggregator. This is in line with our objective to minimize communication overhead. Although there could be strategies that involve coordination between parties, they could involve many rounds of communication. Our experimental results show that this approach is effective for the datasets we evaluated for.

3.2 Loss surface aggregation

Given the sets $E^{(i)}, i \in [p]$ of (HP, loss) pairs $(\boldsymbol{\theta}_t^{(i)}, \mathcal{L}_t^{(i)}), i \in [p], t \in [T]$ at the aggregator, we wish to construct a loss surface $\ell : \boldsymbol{\Theta} \to \mathbb{R}$ that best emulates the (relative) performance loss $\ell(\boldsymbol{\theta})$ we would observe when training the model on $\overline{D}$. Based on our hypothesis, we want the loss surface to be such that it would have a relatively low $\ell(\boldsymbol{\theta})$ if $\boldsymbol{\theta}$ has a low loss for all parties simultaneously. However, because of the asynchronous and adaptive nature of the local HPOs, for any HP $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, we would not have the corresponding losses from all the parties. For that reason, we will model the loss surfaces using regressors that try to map any HP to their corresponding loss. In the following, we present four ways of constructing such loss surfaces:

Single global model (SGM). We merge all the sets $E^{(i)}, i \in [p]$ into E and use it as a training set for a regressor $f : \boldsymbol{\Theta} \to \mathbb{R}$, which considers the HPs $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ as the covariates and the corresponding loss as the dependent variable. For example, we can train a random forest regressor on this training set E. Then we can define the loss surface $\ell(\boldsymbol{\theta}) := f(\boldsymbol{\theta})$. However, this loss surface does not have our

Table 1: Comparison of different loss surfaces (the 4 rightmost columns) for FLoRA relative to the baseline for single-shot 3-party FL-HPO in terms of the *relative regret* (lower is better). See text in §4.1 for the detailed description of "Party max/min".

Data	Party max/min	SGM	SGM+U	MPLM	APLM
EEG eye state	1.005	0.1507	0.1347	**0.1233**	0.1279
Electricity	1.009	0.1848	0.1518	**0.1089**	0.1381
Heart statlog	1.109	0.6904	0.5543	0.8930	**0.5008**
Oil spill	1.205	0.7086	**0.4032**	0.5678	0.5282
PC3	1.044	0.6639	0.7220	0.3921	**0.3797**
Pollen	1.016	0.4328	0.5403	**0.4269**	0.6896
Sonar	1.055	1.3298	**0.4058**	0.9215	0.7094
Aggregate	-	0.5944 ± 0.3997	$\mathbf{0.416 \pm 0.21}$	0.4905 ± 0.3286	0.4391 ± 0.2375

desirable properties: it is actually overly optimistic – under the assumption that every party generates unique HPs during the local HPO, this single global loss surface would assign a low loss to any HP $\boldsymbol{\theta}$ which has a low loss at any one of the parties. This implies that this loss surface would end up recommending HPs that have low loss in just one of the parties.

Single global model with uncertainty (SGM+U). Given the merged set E of the per-party sets of (HP, loss) pairs, we can train a regressor that provides uncertainty quantification around its predictions (such as Gaussian Process Regressor) as $f : \boldsymbol{\Theta} \to \mathbb{R}, u : \boldsymbol{\Theta} \to \mathbb{R}_+$, where $f(\boldsymbol{\theta})$ is the mean prediction of the model at $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ while $u(\boldsymbol{\theta})$ quantifies the uncertainty around this prediction $f(\boldsymbol{\theta})$. We define the loss surface as $\ell(\boldsymbol{\theta}) := f(\boldsymbol{\theta}) + \alpha \cdot u(\boldsymbol{\theta})$ for some $\alpha > 0$. This loss surface does prefer HPs that have a low loss even in just one of the parties, but it penalizes a HP if the model estimates high uncertainty around this HP. Usually, a high uncertainty around a HP would be either because the training set E does not have many samples around this HP (implying that not many parties thought that the region where this HP lies is a region for low loss), or because there are multiple samples in the region around this HP but parties do not collectively agree that this is a promising region for HPs. Hence this makes SGM+U more desirable than SGM, giving us a loss surface that estimates low loss for HPs that are simultaneously thought to be promising to multiple parties.

Maximum of per-party local models (MPLM). Instead of a single global model on the merged set E, we can instead train a regressor $f^{(i)} : \boldsymbol{\Theta} \to \mathbb{R}, i \in [p]$ with each of the per-party set $E^{(i)}, i \in [p]$ of (HP, loss) pairs. Given this, we can construct the loss surface as $\ell(\boldsymbol{\theta}) := \max_{i \in [p]} f^{(i)}(\boldsymbol{\theta})$. This can be seen as a much more pessimistic loss surface, assigning a low loss to a HP only if it has a low loss estimate across all parties.

Average of per-party local models (APLM). A less pessimistic version of MPLM would be to construct the loss surface as the average of the per-party regressors $f^{(i)}, i \in [p]$ instead of the maximum, defined as $\ell(\boldsymbol{\theta}) := {}^1\!/_p \sum_{i=1}^{p} f^{(i)}(\boldsymbol{\theta})$. This is also less optimistic than SGM since it will assign a low loss for a HP only if its average across all per-party regressors is low, which implies that all parties observed a relatively low loss around this HP.

Intuitively, we believe that loss surfaces such as SGM+U or APLM would be the most promising while the extremely optimistic and pessimistic SGM and MPLM respectively would be relatively less promising, with MPLM being superior to SGM. In the following section, we evaluate all these loss surface empirically in the single-shot FL-HPO setting.

4 Empirical evaluation

In this section, we evaluate our proposed scheme and different loss surfaces for the FL-HPO of gradient boosted decision trees [5] on OpenML [6] classification problems. Specifically, we focus on the histogram based gradient boosting, available as `HistGradientBoostingClassifier` in the `sklearn.ensemble` module of the `scikit-learn` library [49]. The precise HP search space is described in Appendix A.2. First, we fix the number of parties $p = 3$ and compare our proposed scheme to a baseline on 7 data sets. Then we study the effect of increasing the number of parties from $p = 3$ to $p = 10$ on the performance of our proposed scheme on 3 data sets. The data is randomly split across parties. Finally, we evaluate our proposed **FLoRA** in a real FL testbed IBM FL [50] using its default HP setting as a baseline.

Single-shot baseline. To appropriately evaluate our proposed single-shot FL-HPO scheme, we need to select a meaningful single-shot baseline. For this, we choose the default HP configuration of `HistGradientBoostingClassifier` in `scikit-learn` as the single-shot baseline. We choose this baseline for two main reasons: (i) this default HP configuration in `scikit-learn` is set manually

based on expert prior knowledge and extensive empirical evaluation, and (ii) this HP configuration is also used as the default for gradient boosting decision trees in the Auto-Sklearn package [38, 51], one of the leading open-source AutoML python packages, which maintains a carefully selected portfolio of default configurations.

Data set selection. For our evaluation of single-shot HPO, we consider 7 binary classification data sets of varying sizes and characteristics from OpenML [6] such that there is at least a significant room for improvement over the performance single-shot baseline. We consider data sets which have at least $> 3\%$ potential improvement in balanced accuracy. See Appendix A.1 for details on data.

Implementation. We consider two implementations for our empirical evaluation. In our first set of experiments in §4.1 and §4.2, we emulate the final FL (Algorithm 1, line 8) with a centralized training using the pooled data. We chose this implementation because we want to evaluate the final performance of any HP configuration (baseline or recommended by FLoRA) in a statistically robust manner with multiple train/validation splits (for example, via 10-fold cross-validation) instead of evaluating the performance on a single train/validation. This form of evaluation is extremely expensive to perform in a real FL system. This form of evaluation also allows us to evaluate how the performance of our single-shot HP recommendation fairs against that of the best-possible HP found via a full-scale centralized HPO. This is again not feasible in a real FL system. To highlight that our proposed scheme do translate to improved performance in a real FL testbed, we utilize the IBM FL library [50] on 3 of the data sets in §4.3. In that case, we report the metrics on a single train/test split.

Evaluation metric. In all data sets, we consider the balanced accuracy as the metric we wish to maximize. For the local per-party HPOs (as well as the centralized HPO we execute to compute the regret), we maximize the 10-fold cross-validated balanced accuracy. For the experiments in §4.1 and §4.2, we report the relative regret, computed as $(a^\star - a)/(a^\star - b)$, where $a^\star$ is the best possible accuracy obtained via the centralized HPO, b is the accuracy of the baseline, and a is the accuracy of the HP recommended by any scheme (baseline or FLoRA). The baseline has a relative regret of 1 and smaller values imply better performance. A value larger than 1 implies that the recommended HP performs worse than the baseline. For the experiments in §4.3 with a real FL system, we report the balanced accuracy of any HP (baseline or recommended by FLoRA) on a single train/test split. Given balanced accuracy as the evaluation metric, we utilize (1 - balanced accuracy) as the loss $\mathcal{L}_t^{(i)}$ in Algorithm 1.

4.1 Comparison to single-shot baseline

In our first set of experiments for 3-party FL-HPO ($p = 3$), we compare our proposed scheme with the baseline across different data sets and report the relative regret for different choices of the loss surfaces in Table 1. In this table, we also report the following ratio in the second column as the "Party max/min": $(1-\min_{i\in[p]} \mathcal{L}_\star^{(i)})/(1-\max_{i\in[p]} \mathcal{L}_\star^{(i)})$, where $\mathcal{L}_\star^{(i)} = \min_{t\in[T]} \mathcal{L}_t^{(i)}$ is the minimum loss observed during the local asynchronous HPO at party i. This ratio is always greater than 1, and highlights the difference in the observed performances across the parties. A ratio closer to 1 indicates that all the parties have relatively similar performances on their training data, while a ratio much higher than 1 indicating significant discrepancy between the per-party performances, implicitly indicating the difference in the per-party data distributions. Table 1 indicates that there are significant differences for Oil spill and Heart statlog data sets and very small differences for the Electricity and EEG eye state data sets.

The results indicate that, in almost all cases, with all loss functions, our proposed scheme is able to improve upon the baseline to varying degrees (there is only one case where SGM performs worse than the baseline on Sonar). On average (across the data sets), SGM+U and APLM perform the best as we expected, with both of them also having significantly smaller standard deviations for the relative regret compared to SGM and MPLM. MPLM performs better than SGM both in terms of average and standard deviation. Looking at the individual data sets, we see that, for data sets with low "Party max/min" (EEG eye state, Electricity), all the proposed loss surface have low relative regret, indicating that the problem is easier as expected. For data sets with high "Party max/min" (Heart statlog, Oil spill), the relative regret of all loss surfaces are higher (but still much smaller than 1), indicating that our proposed single-shot scheme can show improvement even in cases where there is significant difference in the per-party losses (and hence data sets).

4.2 Effect of increasing number of parties

In the second set of experiments, we study the effect of increasing the number of parties in the FL-HPO problem on 3 data sets. We present the relative regrets (along with

Table 2: Effect of increasing the number of parties on FLoRA with different loss surfaces. The experimental setup is described in §4.2

Data	# parties	Party max/min	SGM	SGM+U	MPLM	APLM
EEG eye state	3	1.005	0.1507	0.1347	**0.1233**	0.1279
14980 rows	6	1.011	0.0685	**0.0023**	0.0753	0.0890
	10	1.033	0.0822	**0.0000**	0.1644	0.0137
Electricity	3	1.009	0.1848	0.1518	**0.1089**	0.1381
45312 rows	6	1.007	0.2626	0.2198	0.1907	**0.1420**
	10	1.005	**0.0447**	0.0700	0.3385	0.1518
Pollen	3	1.016	0.4328	0.5403	**0.4269**	0.6896
3848 rows	6	1.101	1.0239	0.9164	**0.5403**	0.5644
	10	1.159	1.0478	**0.7313**	0.7522	1.1254

the "Party max/min") in Table 2. We notice that increasing the number of parties does not have a significant effect on the "Party max/min" for the Electricity data set, but significantly increases for the Pollen data set (making the problem harder). For the EEG eye state, the increase in the "Party max/min" with increasing number of parties is moderate. The results indicate that, with low or moderate increase in "Party max/min" (EEG eye state, Electricity), the proposed scheme is able to achieve low relative regret – the increase in the number of parties does not directly imply degradation in performance. However, with significant increase in "Party max/min" (Pollen), we see a significant increase in the relative regret (eventually going over 1 in a few cases). The difference in data size with Pollen having far fewer data points than the others is likely to have an impact on the performance as well. It is important to note that in this challenging case, MPLM (the most pessimistic loss function) has the most graceful degradation in relative regret compared to the remaining loss surfaces.

4.3 Federated Learning testbed evaluation

We now conduct experiments in a FL testbed, utilizing IBM FL library [50], which contains an implementation of `HistGradientBoostingClassifier` for FL [8]. More specifically, we reserved 40% of oil spill and electricity and 20% of EEG eye state as hold-out test set to evaluate the final FL model performance while each party randomly sampled from the rest of the original dataset to obtain their own training dataset. We use the same HP search space as in Appendix A.2. Our target metric for all experiments is balanced accuracy. Each party will run HPO to generate $T = 500$ (HP, loss) pairs and use those pairs to generate loss surface either collaboratively or by their own according to different aggregation procedures described in §3.2. Once the loss surface is generated, the aggregator uses Hyperopt [32] to select the best HP candidate and train a federated XGBoost model via the IBM FL library using the selected HPs. Table 3 summarizes the experimental results for 3 datasets, indicating that FLoRA can significantly improve over the baseline in IBM FL testbed.

Table 3: Performance of FLoRA with the IBM-FL system in terms of the *balanced accuracy* on a holdout test set (higher is better). The baseline is still the default HP configuration of `HistGradientBoostingClassifier` in `scikit-learn`.

Data	# parties	# training data per party	Baseline	SGM	SGM+U	MPLM	APLM
Oil spill	3	200	0.5895	**0.7374**	0.5909	0.7061	0.7332
EEG eye state	3	3,000	0.8864	0.9153	0.9211	**0.9251**	0.9245
Electricity	6	4,000	0.8448	0.8562	**0.8627**	0.8621	0.8624

5 Conclusions and next steps

How to effectively select HP in FL settings is an unsolved problem. In this paper, we introduced FLoRA, a single-shot FL-HPO algorithm that can be applied to a variety of ML models. Our experimental evaluation shows that FLoRA can produce HPO configurations that outperform the baseline and deal with highly heterogeneous distributions among parties. We plan to evaluate FLoRA on more data sets and FL-HPO of more machine learning methods (such as random decision forests, nearest neighbor models, kernel machines, neural network) to further quantify its performance. Moreover, we plan to extend our proposed algorithm to go from single-shot (one HP recommendation for one FL training) to a few-shot setup (where we would be allowed to perform a very small number of FL trainings). Finally, we plan to extend this approach to allow for personalization, using local party-specific HPs.

References

[1] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[2] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[3] Mikhail Khodak, Renbo Tu, Tian Li, Liam Li, Maria-Florina Balcan, Virginia Smith, and Ameet Talwalkar. Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing. *arXiv preprint arXiv:2106.04502*, 2021.

[4] Mikhail Khodak, Tian Li, Liam Li, M Balcan, Virginia Smith, and Ameet Talwalkar. Weight sharing for hyperparameter optimization in federated learning. In *Int. Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML 2020*, 2020.

[5] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[6] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013. doi: 10.1145/2641190.2641198. URL http://doi.acm.org/10.1145/2641190.2641198

[7] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for on-device federated learning. 2019.

[8] Yuya Jeremy Ong, Yi Zhou, Nathalie Baracaldo, and Heiko Ludwig. Adaptive histogram-based gradient boosted trees for federated learning. *arXiv preprint arXiv:2012.06670*, 2020.

[9] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[10] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

[11] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International Conference on Machine Learning*, pages 6357–6368. PMLR, 2021.

[12] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. *arXiv preprint arXiv:1705.10467*, 2017.

[13] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[14] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems*, 33:3557–3568, 2020.

[15] Annie Abay, Yi Zhou, Nathalie Baracaldo, Shashank Rajamoni, Ebube Chuba, and Heiko Ludwig. Mitigating bias in federated learning. *arXiv preprint arXiv:2012.02447*, 2020.

[16] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. In *International Conference on Machine Learning*, pages 4615–4625. PMLR, 2019.

[17] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[18] Zheng Chai, Ahsan Ali, Syed Zawad, Stacey Truex, Ali Anwar, Nathalie Baracaldo, Yi Zhou, Heiko Ludwig, Feng Yan, and Yue Cheng. Tifl: A tier-based federated learning system. In *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, pages 125–136, 2020.

[19] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

[20] Naman Agarwal, Ananda Theertha Suresh, Felix Yu, Sanjiv Kumar, and H Brendan Mcmahan. cpsgd: Communication-efficient and differentially-private distributed sgd. *arXiv preprint arXiv:1805.10559*, 2018.

[21] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 1–11, 2019.

[22] Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, and Heiko Ludwig. Hybridalpha: An efficient approach for privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 13–23, 2019.

[23] Z. Dai, B.K.H. Low, and P. Jaillet. Federated bayesian optimization via thompson sampling. *Advances in Neural Information Processing Systems*, 33, 2020.

[24] A. Koskela and A. Honkela. Learning rate adaptation for federated and differentially private learning. *arXiv preprint arXiv:1809.03832*, 2019.

[25] H. Mostafa. Robust federated learning through representation matching and adaptive hyper-parameters. *arXiv preprint arXiv:1912.13075*, 2019.

[26] S.J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konecny, S. Kumar, and H.B. McMahan. Adaptive federated optimization. In *International Conference on Learning Representations*, 2020.

[27] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[28] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[29] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, 2012.

[30] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.

[31] A Costa and G Nannicini. Rbfopt: an open-source library for black-box optimization with costly function evaluations. *Math. Prog. Comp. 10*, 2018.

[32] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

[33] Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, pages 240–248, 2016.

[34] Ashish Sabharwal, Horst Samulowitz, and Gerald Tesauro. Selecting near-optimal learners via incremental data allocation. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[35] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18 (185):1–52, 2018.

[36] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1437–1446, 2018.

[37] Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-weka: Automated selection and hyper-parameter optimization of classification algorithms. *arXiv*, 2012. URL `http://arxiv.org/abs/1208.3719`.

[38] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, pages 2962–2970, 2015.

[39] Lars Kotthoff, Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *J. Mach. Learn. Res.*, 18(1):826–830, January 2017. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=3122009.3122034`.

[40] Herilalaina Rakotoarison, Marc Schoenauer, and Michele Sebag. Automated machine learning with monte-carlo tree search. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3296–3303, 2019.

[41] Sijia Liu, Parikshit Ram, Deepak Vijaykeerthy, Djallel Bouneffouf, Gregory Bramble, Horst Samulowitz, Dakuo Wang, Andrew Conn, and Alexander Gray. An ADMM based framework for automl pipeline configuration. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020. URL https://arxiv.org/abs/1905.00424v5.

[42] Joaquin Vanschoren. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*, 2018.

[43] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Learning data set similarities for hyperparameter optimization initializations. In *Metasel@ pkdd/ecml*, pages 15–26, 2015.

[44] Matthias Feurer, Jost Springenberg, and Frank Hutter. Initializing bayesian hyperparameter optimization via meta-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[45] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Learning hyperparameter optimization initializations. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pages 1–10. IEEE, 2015.

[46] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Hyperparameter search space pruning–a new component for sequential model-based hyperparameter optimization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 104–119. Springer, 2015.

[47] Valerio Perrone, Huibin Shen, Matthias W Seeger, Cedric Archambeau, and Rodolphe Jenatton. Learning search spaces for bayesian optimization: Another view of hyperparameter transfer learning. *Advances in Neural Information Processing Systems*, 32:12771–12781, 2019.

[48] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. Scalable gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning*, 107(1):43–78, 2018.

[49] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[50] Heiko Ludwig, Nathalie Baracaldo, Gegi Thomas, Yi Zhou, Ali Anwar, Shashank Rajamoni, Yuya Ong, Jayaram Radhakrishnan, Ashish Verma, Mathieu Sinn, et al. IBM Federated Learning: an enterprise framework white paper v0. 1. *arXiv preprint arXiv:2007.10987*, 2020. URL https://github.com/IBM/federated-learning-lib

[51] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Auto-sklearn 2.0: The next generation. In *arXiv:2007.04074 [cs.LG]*, 2020.