Auto-FedRL: Federated Hyperparameter Optimization for Multi-institutional Medical Image Segmentation

Pengfei Guo[⋆1], Dong Yang[2], Ali Hatamizadeh[2], An Xu[3], Ziyue Xu[2], Wenqi Li[2], Can Zhao[2], Daguang Xu[2], Stephanie Harmon[4], Evrim Turkbey[5], Baris Turkbey[4], Bradford Wood[5], Francesca Patella[6], Elvira Stellato[7], Gianpaolo Carrafiello[7], Vishal M. Patel[1], and Holger R. Roth[2]

[1] Johns Hopkins University
[2] NVIDIA
[3] University of Pittsburgh
[4] National Cancer Institute
[5] National Institutes of Health
[6] ASST Santi Paolo e Carlo
[7] University of Milan

Abstract. Federated learning (FL) is a distributed machine learning technique that enables collaborative model training while avoiding explicit data sharing. The inherent privacy-preserving property of FL algorithms makes them especially attractive to the medical field. However, in case of heterogeneous client data distributions, standard FL methods are unstable and require intensive hyperparameter tuning to achieve optimal performance. Conventional hyperparameter optimization algorithms are impractical in real-world FL applications as they involve numerous training trials, which are often not affordable with limited compute budgets. In this work, we propose an efficient reinforcement learning (RL)-based federated hyperparameter optimization algorithm, termed Auto-FedRL, in which an online RL agent can dynamically adjust hyperparameters of each client based on the current training progress. Extensive experiments are conducted to investigate different search strategies and RL agents. The effectiveness of the proposed method is validated on a heterogeneous data split of the CIFAR-10 dataset as well as two real-world medical image segmentation datasets for COVID-19 lesion segmentation in chest CT and pancreas segmentation in abdominal CT.

Keywords: FL, Reinforcement Learning, Hyperparameter Optimization

1 Introduction

A large amount of data is needed to train robust and generalizable machine learning models. A single institution often does not have enough data to train

---

such models effectively. Meanwhile, there are emerging regulatory and privacy concerns about the data sharing and management [21,13]. Federated Learning (FL) [34] mitigates such concerns as it leverages data from different clients or institutions to collaboratively train a global model while allowing the data owners to control their private datasets. Unlike the conventional centralized training, FL algorithms open the potential for multi-institutional collaborations in a privacy-preserving manner [63]. This multi-institutional collaboration scenario often refers to *cross-silo* FL [20] and is the main focus of this paper. In this FL setting, clients are autonomous data owners, such as medical institutions storing patients' data, and collaboratively train a general model to overcome the data scarcity issue and privacy concerns [63]. This makes *cross-silo* FL applications especially attractive to the healthcare sector [43,14]. Several methods have already been proposed to leverage FL for multi-institutional collaborations in digital healthcare [59,15,50,44].

The most recently introduced FL frameworks [59,15,38,27] are variations of the Federated Averaging (FedAvg) [34] algorithm. The training process of FedAvg consists of the following steps: (i) clients perform local training and upload model parameters to the server. (ii) The server carries out the averaging aggregation over the received parameters from clients and broadcasts aggregated parameters to clients. (iii) Clients update local models and evaluate its performance. After sufficient communication rounds between clients and the server, a global model can be obtained. The design of FedAvg is based on standard Stochastic Gradient Descent (SGD) learning with the assumption that data is uniformly distributed across clients [34]. However, in real-world applications, one has to deal with underlying unknown data distributions that are likely not independent and identically distributed (non-iid). The heterogeneity of data distributions has been identified as a critical problem that causes the local models to diverge during training and consequently sub-optimal performance of the trained global model [31,38,27].

To achieve the required performance, the proper tuning of hyperparameters (*e.g.*, the learning rate, the number of local iterations, aggregation weights, *etc.*) plays a critical role for the success of FL [38,59]. [29] shows that the learning rate decay is a necessary condition of the convergence for FL on non-iid data. While the general hyperparameter optimization has been intensively studied [6,58,53], the unique setting of FL makes federated hyperparameters optimization especially difficult [24]. Reinforcement learning (RL) provides a promising solution to approach this complex optimization problem. Compared to other methods for finding the optimal hyperparameters, RL-based methods do not require the prior knowledge of the complicated underlying system dynamics [47]. Thus, federated hyperparameter optimization can be reduced to defining appropriate reward metrics, search space, and RL agents.

In this paper, we aim to make the automated hyperparameter optimization applicable in realistic FL applications. An online RL algorithm is proposed to dynamically tune hyperparameters during a single trial. Specifically, the proposed Auto-FedRL formulates hyperparameter optimization as a task of

discovering optimal policies for the RL agent. Auto-FedRL can dynamically adjust hyperparameters at each communication round based on relative loss reduction. Without the need for multiple training trails, an online RL agent is introduced to maximize the rewards in small intervals, rather than the sum of all rewards. While RL-based hyperparameter optimization method has been explored in [38], our experiments show that the prior work has several deficiencies impeding its practical use in real-world applications. (**i**) The discrete action space (*i.e.*, hyperparameter search space) not only leads to limited available actions but also suffers from scalability issues. At each optimization step, the gradient of all possible hyperparameter combinations is retained, which causes high memory consumption and com-

Fig. 1. The computational details of different search strategies under the same setting on CIFAR-10 when the number of clients equals to 2 (△), 4 (+), 6 (◇), and 8 (×). The green box shows the zoomed-in region.

putational inefficiency. Therefore, as shown in Fig. 1, the hardware limitation can be reached quickly, when one needs to collaborate with multiple institutions using a large search space. To circumvent this challenge, Auto-FedRL can leverage continuous search space. Its memory usage is practically constant as the memory consumption per hyperparameter is negligible and does not explode with increased search space and the number of involved clients. Meanwhile, its computational efficiency is significantly improved compared to discrete search space. (**ii**) The flexibility of hyperparameter search space is limited. [38] focuses on a small number of hyperparameters (*e.g.*, one or two hyperparameters) in less general settings. In contrast, our method is able to tune a wide range of hyperparameters (*e.g.*, client/server learning rates, the number of local iterations, and the aggregation weight of each client) in a realistic FL setting. It is worth noting that the averaging model aggregation is replaced by a pseudo-gradient optimization [42] in Auto-FedRL. Thus, we are able to search server-side hyperparameters. To this end, we propose a more practical federated hyperparameter optimization framework with notable computational efficiency and flexible search space.

Our main contributions in this work are summarized as follows:

- A novel federated hyperparameter optimization framework Auto-FedRL is proposed, which enables the dynamic tuning of hyperparameters via a single trial.
- Auto-FedRL makes federated hyperparameter optimization more practical in real-world applications by efficiently incorporating continuous search space and the deep RL agent to tune a wide range of hyperparameters.
- Extensive experiments on multiple datasets show the superior performance and notable computational efficiency of our methods over existing FL baselines.
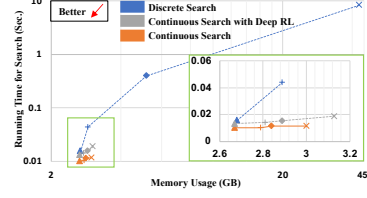
2   Related Works

Federated Learning on Heterogeneous Data.   The heterogeneous data distribution across clients impedes the real-world deployment of FL applications and draws emerging attentions. Several methods [65,48,29,23,7,61,35] have been proposed to address this issue. FedOpt [42] introduced the adaptive federated optimization, which formulated a more flexible FL optimization framework but also introduced more hyperparameters, such as the server learning rate and server-side optimizers. FedProx [27] and Agnostic Federated Learning (AFL) [37] are variants of FedAvg [34] which attempted to address the learning bias issue of the global models for local clients by imposing additional regularization terms. FedDyn [2] was proposed to address the problem that the minima of the local-device level loss are inconsistent with those of the global loss by introducing a dynamic regularizer for each device. Those works demonstrated good theoretical analysis but are evaluated only on manually created toy datasets. Recently, FL-MRCM [15] was proposed to address the domain shift issue among different clients by aligning the distribution of latent features between the source domain and the target domain. Although those methods [30,15] achieved promising results in overcoming domain shift in the multi-institutional collaboration, directly sharing latent features between clients increased privacy concerns.

Conventional Hyperparameter Optimization. Grid and random search [6] can perform automated hyperparameter tuning but require long running time due to often exploring unpromising regions of the search space. While advanced random search [5] and Bayesian optimization-based search methods [58,53] require fewer iterations, several training trails are required to evaluate the fitness of hyperparameter configurations. Repeating the training process multiple times is impractical in the FL setting, especially for deep learning models, due to the limited communication and compute resources in real-world FL setups.

Federated Hyperparameter Optimization. Auto-FedAvg [59] is a recent automated search method, which only is compatible with differentiable hyperparameters and focuses on searching client aggregation weights. The method proposed in [38] is the most relevant to our work. However, as discussed in the previous section, it suffers from limited practicability and flexibility of search space in real-world applications. Inspired by the recent hyperparameter search [3,11,12] and differentiable [32,8], evolutionary [41,60] and RL-based automated machine learning methods [66,4], we propose an efficient automated approach with flexible search space to discover a wide range of hyperparameters.

3   Methodology

In this section, we first introduce the general notations of FL and the adaptive federated optimization that provides the theoretical foundation of tuning FL server-side hyperparameters (Sec. 3.1). Then, we describe our method in detail (Sec. 3.2), including online RL-based hyperparameter optimization, the

discrete/continuous search space, and the deep RL agent. In addition, we provide theoretical analysis to guarantee the convergence of Auto-FedRL in the supplementary material.

3.1   Federated Learning

In a FL system, suppose K clients collaboratively train a global model. The goal is to solve the optimization problem as follows:

$$\min_{x \in \mathbb{R}^d} \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}_k(x), \tag{1}$$

where $\mathcal{L}_k(x) = \mathbb{E}_{z \sim \mathcal{D}_k}[\mathcal{L}_k(x, z)]$ is the loss function of the kth client. $z \in \mathcal{Z}$, and $\mathcal{D}_k$ represents the data distribution of the kth client. Commonly, for two different clients i and j, $\mathcal{D}_i$ and $\mathcal{D}_j$ can be dissimilar, so that Eq. 1 can become nonconvex. A widely used method for solving this optimization problem is FedAvg [34]. At each round, the server broadcasts the global model to each client. Then, all clients conduct local training on their own data and send back the updated model to the server. Finally, the server updates the global model by a weighted average of these local model updates. FedAvg's server update at round q can be formulated as follows:

$$\Theta^{q+1} = \sum_{k=1}^{K} \alpha_k \Theta_k^q, \tag{2}$$

where Θ_k^q denotes the local model of kth client and α_k is the corresponding aggregation weight. The update of global model Θ^{q+1} in Eq. 2 can be further rewritten as follows:

$$
\begin{aligned}
\Theta^{q+1} &= \Theta^q - \sum_{k=1}^{K} \alpha_k (\Theta^q - \Theta_k^q) \\
&= \Theta^q - \sum_{k=1}^{K} \alpha_k \Delta_k^q \\
&= \Theta^q - \Delta^q,
\end{aligned}
\tag{3}
$$

where $\Delta_k^q := \Theta^q - \Theta_k^q$ and $\Delta^q := \sum_{k=1}^{K} \alpha_k \Delta_k^q$. Therefore, the server update in FedAvg is equivalent to applying optimization to the *pseudo-gradient* $-\Delta^q$ with a learning rate $\gamma = 1$. This general FL optimization formulation refers to adaptive federated optimization [42]. Auto-FedRL utilizes this *pseudo-gradient* update formulation to enable the server-side hyperparameter optimization, such as the server learning rate γ.

3.2   Auto-FedRL

Online RL Hyperparameter Optimization. The online setting in the targeted task is very challenging since the same actions at different training stages may receive various responses. Several methods [40,1,19] have been proposed in

the literature to deal with such non-stationary problems. However, these methods require multiple training runs, which is usually not affordable in FL settings where clients often have limited computation resources. Typically, a client can run only one training procedure at the same time and the resources for parallelization as would be done in a cluster environment is not available. To circumvent the limitations of conventional hyperparameter optimization methods and inspired by previous works [4,38,66], we introduce an online RL-based approach to directly learn the proper hyperparameters from data at the clients' side during a single training trial. At round q, a set of hyperparameters h^q can be sampled from the distribution $P(\mathcal{H}|\psi^q)$. We denote the validation loss of client k at round q as $\mathcal{L}^q_{\text{val}_k}$ and the hyperparameter loss at round q as

$$\mathcal{L}^q_h = \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}^q_{\text{val}_k}. \tag{4}$$

The relative loss reduction reward function of the RL agent is defined as follows:

$$r^q = \frac{\mathcal{L}^q_h - \mathcal{L}^{q+1}_h}{\mathcal{L}^q_h}. \tag{5}$$

The goal of the RL agent at round q is to maximize the objective as follows:

$$J^q = \mathbb{E}_{P(h^q|\psi^q)}[r^q]. \tag{6}$$

By leveraging the one-sample Monte Carlo estimation technique [57], we can approximate the derivative of J^q as follows:

$$\nabla_{\psi^q} J^q = r^q \nabla_{\psi^q} \log(P(h^q|\psi^q)). \tag{7}$$

To this end, we can evaluate Eq. 6 and use it to update the condition of hyperparameter distribution ψ^q. To formulate an online algorithm, we utilize the averaged rewards in a small interval ("window") rather than counting the sum of all rewards to update ψ^q as follows:

$$\psi^{q+1} \leftarrow \psi^q - \gamma_h \sum_{\tau=q-Z}^{\tau=q} (r^\tau - \hat{\tau}^q) \nabla_{\psi^\tau} \log(P(h^\tau|\psi^\tau)), \tag{8}$$

where Z is the size of the update window and γ_h is the RL agent learning rate. The averaged rewards $\hat{\tau}^q$ in the interval $[q - Z, q]$ are defined as follows:

$$\hat{\tau}^q = \frac{1}{Z+1} \sum_{\tau=q-Z}^{\tau=q} r^\tau. \tag{9}$$

Discrete Search. Selecting the form of hyperparameter distribution $P(\mathcal{H}|\psi)$ is non-trivial, since it determines the available actions in the search space. We denote the proposed method using discrete search (DS) space as Auto-FedRL(DS). Here, $P(\mathcal{H}|\psi)$ is defined by a D-dimensional discrete Gaussian distribution, where D denotes the number of searchable hyperparameters. For each hyperparameter, there is a finite set of available selections. Therefore, $\mathcal{H}$ is a grid that consists of all possible combinations of available hyperparameters. A hyperparameter combination h^q at round q is a point on $\mathcal{H}$ as follows:

$$P(h^q|\psi^q) = \mathcal{N}(h^q|\mu^q, \Sigma^q), \tag{10}$$

where $h^q = \{h^q_1, \ldots, h^q_D\}$. ψ^q is defined by the mean vector μ^q and the covariance matrix Σ^q, which are learnable parameters that the RL agent targets to
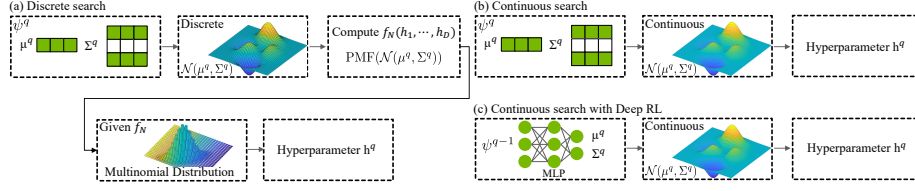
Fig. 2. The sampling workflow comparison of different search strategies in the proposed Auto-FedRL. PMF denotes the probability mass function.

optimize. To increase the stability of RL training and encourage learning in all directions, different types of predefined hyperparameter selections are normalized to the same scale with zero-mean when constructing the search space. This hyperparameter sampling procedure is presented in Fig. 2 (a).

Continuous Search. While defining a discrete action space can be more controllable for hyperparameter optimization, as discussed in Section 1, it limits the scalability of the search space. The gradients of all possible hyperparameter combinations are retained in the discrete search during the windowed update as in Eq. 8, which requires a large amount of memory. To overcome this issue, we extend Auto-FedRL(DS) to Auto-FedRL(CS), that can utilize a continuous search (CS) space for the RL agent. Instead of constructing a gigantic grid that stores all possible hyperparameter combinations, one can directly sample a choice from a continuous multivariate Gaussian distribution $\mathcal{N}(\mu^q, \Sigma^q)$. It is worth noting that with the expansion of search space, the increase of memory usage of Auto-FedRL(CS) is negligible. A comparison between the hyperparameter sampling workflows in discrete and continuous search are presented in Fig. 2. The main difference between DS and CS lies in the sampling process. In practice, one can adopt the Box–Muller transform [55] for sampling the continuous Gaussian distribution. However, as shown in Fig. 2(a), the sampling for multivariate discrete Gaussian distributions typically involves the following steps: **(i)** We compute the probabilities of all possible combinations. **(ii)** Given the probabilities, we draw a choice from the multinomial distribution or alternatively can use the "inverse CDF" method [49]. In either way, we need to compute the probabilities of all possible hyperparameter combinations for DS, which is not required for CS. Hence, our CS is much more efficient for hyperparameter optimization, as shown in Fig. 1.

Deep RL Agent. An intuitive extension of Auto-FedRL(CS) is to leverage neural networks (NN) as the agent to update the condition of hyperparameter distribution ψ^q rather than the direct optimization. A more complicated RL agent design could deal with potentially more complex search spaces [17]. To investigate the potential of NN-based agent in our setting, we further propose the Auto-FedRL(MLP), which leverages a multilayer perceptron (MLP) as the agent to update the ψ. The sampling workflow of Auto-FedRL(MLP) is presented in Fig. 2(c). The proposed MLP takes the condition of previous hyperparameter distribution ψ^{q-1} as the network's input and predicts the updated ψ^q. Mean-

8   P. Guo et al.

Algorithm 1: Auto-FedRL

Input: initial global model Θ^1; initial hyperparameter combination h^1; server optimizer `ServerOpt`; client optimizer `ClientOpt`; RL optimizer `RLOpt`; RL learning rate γ_h .

for q = 1 to Q **do**
  for k = 1 to K **in parallel do**
    ▷ deploy h^q to clients
    if q = 1 **then**
      ▷ deploy Θ^q to clients
    ▷ unpack P^q and γ_k^q, from h^q
    $\Theta_k^q \leftarrow$ `LocalTrain`$(\Theta^q, P^q, \gamma_k^q)$
    ▷ $\Delta_k^q = \Theta^q - \Theta_k^q$
    ▷ upload Δ_k^q to the server
  ▷ unpack α_k^q and γ^q, from h^q
  ▷ $\Delta^q = \sum_{k=1}^{K} \alpha_k^q \Delta_k^q$
  $\Theta^{q+1} \leftarrow$ `ServerOpt`$(\Theta^q, -\Delta^q, \gamma^q)$
  $h^{q+1} \leftarrow$ `RLUpdate`(Θ^{q+1}, γ_h)
return Θ^Q

`LocalTrain`$(\Theta^q, P^q, \gamma_k^q)$:
  $\Theta_k^{q,1} \leftarrow \Theta^q$
  for p = 1 to P **do**
    ▷ compute the gradients $g_k^{q,p}$
    ▷ $\Theta_k^{q,p+1} =$ `ClientOpt`$(\Theta_k^{q,p}, g_k^{q,p}, \gamma_k)$
  return Θ_k^q
`RLUpdate`(Θ^{q+1}, γ_h):
  ▷ deploy Θ^{q+1} to clients
  for k = 1 to K **in parallel do**
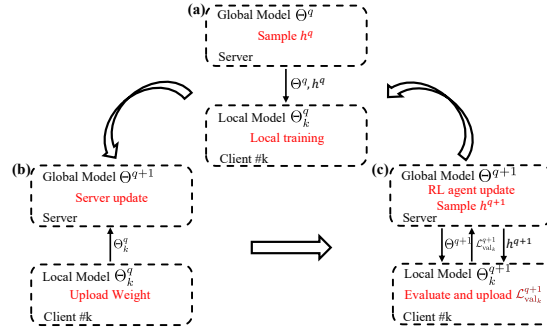   ▷ compute and upload $\mathcal{L}_{\mathrm{val}_k}^{q+1}$ to the server
  $\mathcal{L}_h^{q+1} = \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}_{\mathrm{val}_k}^{q+1}$
  ▷ Compute r^q with Eq. 5
  $\psi^{q+1} \leftarrow$ `RLOpt`(r^{q+1}, γ_h) with Eq. 8
  $h^{q+1} \sim P(\mathcal{H}|\psi^{q+1})$
  return h^{q+1}

Fig. 3. The schematics of Auto-FedRL at round q.

while, due to our online setting (*i.e.* limited optimization steps), we have to keep the learnable parameters in MLP small but effective. The detailed network configuration can be found in the supplementary material.

Full Algorithm. The overview of Auto-FedRL framework is presented in Alg. 1 and Fig. 3. At each training round q, the training of Auto-FedRL consists of following steps: **(i)** As shown in Fig. 3(a), clients receive the global model Θ^q and hyperparameters h^q. Clients perform `LocalTrain` based on the received hyperparameters. **(ii)** The updated local models are then uploaded to the server as shown in Fig. 3(b). Instead of performing the average aggregation, we use *pseudo-gradient* $-\Delta^q$ in Eq. 3 to carry out the server update with a searchable server learning rate. **(iii)** Clients evaluate the received the updated global model Θ^{q+1} and upload the validation loss $\mathcal{L}_{\mathrm{val}_k}^{q+1}$ to the server. The server performs the RL update as shown in `RLUpdate` of Alg. 1. Here, we consider the applicability in a real-world scenario, in which each client maintains its own validation data

rather than relying on validation data being available on the server. Then, the server computes the reward r^{q+1} as in Eq. 5 and updates the RL agent (RLOpt) as in Eq. 8. Finally, hyperparameters for the next training round h^{q+1} can be sampled from the updated hyperparameter distribution $P(\mathcal{H}|\psi^{q+1})$. As shown in Fig. 3(c), the proposed method requires one extra round of communication between clients and the server for $\mathcal{L}_{\text{val}_k}$. It is worth noting that the message size of $\mathcal{L}_{\text{val}_k}$ is negligible. Thus, this extra communication can still be considered practical under our targeted scenario in which all clients have a reliable connection (*i.e.*, multi-institutional collaborations in cross-silo FL).

3.3 Datasets and Implementation Details

CIFAR-10. We simulate an environment in which the number of data points and label proportions are imbalanced across clients. Specifically, we partition the standard CIFAR-10 training set [25] into 8 clients by sampling from a Dirichlet distribution ($\alpha = 0.5$) as in [56]. The original test set in CIFAR-10 is considered as the global test set used to measure performance. VGG-9 [51] is used as the classification network. All models are trained using the following settings: Adam optimizer for RL; SGD optimizer for clients and the server; γ_h of 1×10^{-2}; initial learning rate of 1×10^{-2}; maximum rounds of 100; initial local epochs of 20; batch size of 64.

Multi-national COVID-19 Lesion Segmentation. This dataset contains 3D computed tomography (CT) scans of COVID-19 infected patients collected from three medical centers[1] [16,46,59,62]. We partition this dataset into three clients based on collection locations as following: 671 scans from China (Client I), 88 scans from Japan (Client II), and 186 scans from Italy (Client III). Each voxel containing a COVID-19 lesion was annotated by two expert radiologists. The training/validation/testing data splits are as follows: 447/112/112 (Client I), 30/29/29 (Client II), and 124/31/31 (Client III). The architecture of the segmentation network is 3D U-Net [10]. All models are trained using the following settings: Adam optimizer for RL and clients; γ_h of 1×10^{-2}; SGD optimizer for the server; initial learning rate of 1×10^{-3}; initial local iterations of 300; maximum rounds of 300; batch size of 16.

Multi-institutional Pancreas Segmentation. Here, we utilize 3D CT scans from three public datasets, including 281 scans from the pancreas segmentation subset of the Medical Segmentation Decathlon [52] as Client I, 82 scans from the Cancer Image Archive (TCIA) Pancreas-CT dataset [45] as Client II, and 30 scans from Beyond the Cranial Vault (BTCV) Abdomen dataset [22] as Client III. The training/validation/testing data splits are as follows: 95/93/93 (Client I), 28/27/27 (Client II), and 10/10/10 (Client III). All models are trained using the same network architecture and settings as COVID-19 lesion segmentation except that the maximum rounds are 50.

---

[1] https://wiki.cancerimagingarchive.net/display/Public/CT+Images+in+COVID-19

Table 1. CIFAR-10 classification results. **Bold** and <u>Underline</u> indicate the best and the second best performance, respectively.

Method	Accuracy (%)
FedAvg [34]	88.43
FedProx [27]	89.45
Mostafa *et al.* [38]	89.86
Auto-FedAvg [59]	89.16
Auto-FedRL(DS)	90.70
Auto-FedRL(CS)	<u>90.85</u>
Auto-FedRL(MLP)	**91.27**
Centralized	92.56

Table 2. The computational details of different search strategies under the same setting on CIFAR-10.

Search Space Type	Memory Usage	Running Time for Search
Discrete	42.8 GB	8.246 s
Continuous	**3.00 GB**	**0.012 s**
Continuous MLP	3.13 GB	0.019 s

4   Experiments and Results

In this section, the effectiveness of our approach is first validated on a heterogeneous data split of the CIFAR-10 dataset (Sec. 4.1). Then, experiments are conducted on two multi-institutional medical image segmentation datasets (*i.e.*, COVID-19 lesion segmentation and pancreas segmentation) to investigate the real-world potential of the proposed Auto-FedRL (Sec. 4.2). Finally, detailed comparisons between discrete and continuous search space, and the exploration of deep RL agents are provided (Sec. 4.3). We evaluate the performance of our method against the following popular FL methods: FedAvg [34] and FedProx [27] as well as FL-based hyperparameter optimization methods: Auto-FedAvg [59], and Mostafa *et al.* [38].

4.1   CIFAR-10

Table 1 shows the quantitative performance of different methods in terms of the average accuracy across 8 clients. We denote the model that is directly trained with all available data as *Centralized* in Table 1. We treat it as an upper bound when data can be shared. As can be seen from this table, the proposed Auto-FedRL methods clearly outperform the other competing FL alternatives. Auto-FedRL(MLP) gains the best performance improvement by taking the advantage of a more complex RL agent design. To investigate the underlying hyperparameter change, we plot the evolution of aggregation weights in Fig. 4. We found that the proposed RL agent is able to reveal more informative clients (i.e., clients containing more unique labels) and assign larger aggregation weights to those client's model updates. In particular, in Fig. 4(a), C4 (red), C5 (purple), and C8 (gray) are gradually assigned three of the largest aggregation weights. As shown in Fig. 4(b), although those three clients do not contain the largest number of
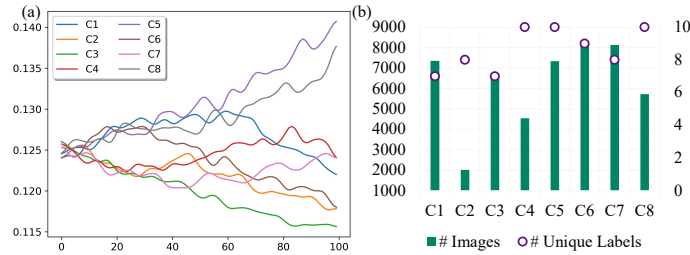
Fig. 4. Analysis of the learning process of Auto-FedRL(MLP) in CIFAR-10. (a) the evolution of aggregation weights during the training. (b) the data statistics of different clients.

Table 3. Multi-national COVID-19 lesion segmentation. † indicates significant improvement (p ≪ 0.05 in the Wilcoxon signed rank test) of the global model over the best counterpart.

Method	Client I	Client II	Client III	Global Test Avg.
Local only - I	59.8	61.8	51.8	57.8
Local only - II	41.9	59.9	50.2	50.7
Local only - III	34.5	52.5	65.9	51.0
FedAvg [34]	59.9	63.8	60.5	61.4±0.2
FedProx [27]	60.3	64.9	60.5	61.9±0.5
Mostafa *et al.* [38]	60.9	64.6	65.6	63.7±0.3
Auto-FedAvg [59]	60.3	65.3	64.8	63.5±0.2
Auto-FedRL(DS)	59.3	65.6	$68.9^{\dagger}$	64.6±0.2
Auto-FedRL(CS)	59.9	$66.1^{\dagger}$	$68.2^{\dagger}$	**64.7**±0.1
Auto-FedRL(MLP)	57.8	65.6	$68.5^{\dagger}$	64.0±0.4
Centralized	61.1	65.9	69.3	65.4

images, all have the most number of unique labels (*i.e.* 10 in CIFAR-10). This behavior further demonstrates the effectiveness of Auto-FedRL. Moreover, we provide the computational details of different search strategies to investigate their practicability under a same setting in Table 2. Without losing performance, the proposed continuous search requires only 7% memory usage but is 690× faster compared to the discrete search. While Auto-FedRL(MLP) introduces the deep RL agent, it is still 430× faster compared to the discrete version. Additional multi-dimensional comparisons [9] are provided in the supplementary material. The notable computational efficiency and low memory usage of Auto-FedRL validate our motivation of making federated hyperparameter optimization more practical in real-world applications.

4.2 Real-world FL Medical Image Segmentation

Multi-national COVID-19 Lesion Segmentation: The quantitative results are presented in Table 3. We show the segmentation results of different methods for qualitative analysis in Fig. 5(a). Dice score is used to evaluate the quality of segmentation. We repeatedly run all FL algorithms 3 times and report the mean
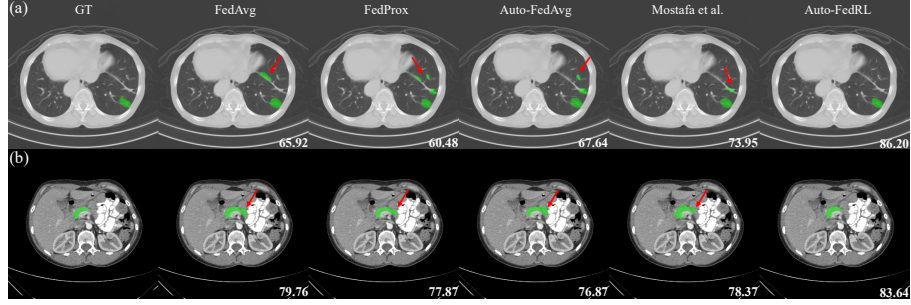
Fig. 5. Qualitative results of different methods that correspond to (a) COVID-19 lesion segmentation of Client III and (b) Pancreas segmentation of Client II. GT shows human annotations in green and others show the segmentation results from different methods. Red arrows point to erroneous segmentation. The dice score is presented in the lower-right corner of each subplot.
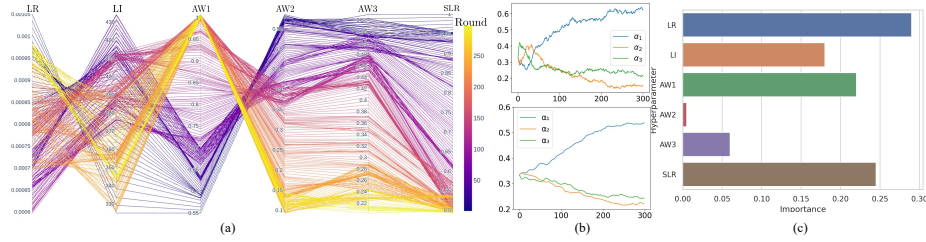

Fig. 6. Analysis of the learning process of Auto-FedRL(CS) in COVID-19 lesion segmentation. (a) The parallel plot of the hyperparameter change during the training. LR, LI, AW, and SLR denote the learning rate, local iterations, the aggregation weight of each client, and the server learning rate, respectively. (b) The aggregation weights evolution of Auto-FedAvg in the *top row* and Auto-FedRL(CS) in the *bottom row*. (c) The importance analysis of different hyperparameters.

and standard deviation. The main metric of evaluating the generalizability of the global model is *Global Test Avg.*, which is computed by the average performance of the global model across all clients. In the first three rows of Table 3, we evaluate three locally trained models as the baseline. Due to the domain shift, all locally trained models exhibit low generalizability on the other clients. By leveraging the additional regularization on weight changes, FedProx (with the empirically best $\mu=0.001$) can slightly outperform the FedAvg baseline. Mostafa *et al.* that uses the RL agent to perform discrete search can achieve slightly better performance than Auto-FedAvg. We find that with the nearly constant memory usage and notable computational efficiency, the proposed Auto-FedRL(CS) achieves the best performance, outperforming the most competitive method [38] by 1.0% in terms of the global model performance, by 1.5% and 2.6% on clients II and III, respectively. The performance gap between the FL algorithm and centralized

Table 4. Multi-institutional pancreas segmentation. † indicates significant improvement of the global model over the best counterpart.

Method	Clinet I	Clinet II	Clinet III	**Global Test Avg.**
Local only - I	69.4	71.4	63.8	68.2
Local only - II	49.7	75.5	53.0	59.3
Local only - III	42.4	61.2	51.1	51.3
FedAvg [34]	71.9	78.4	69.1	73.1±0.3
FedProx [27]	72.0	78.4	69.6	73.3±0.3
Mostafa *et al.* [38]	74.4	79.4	72.1	75.3±0.1
Auto-FedAvg [59]	71.3	79.9	71.5	74.2±0.3
Auto-FedRL(DS)	72.8	$80.8^{\dagger}$	$74.7^{\dagger}$	76.1±0.4
Auto-FedRL(CS)	73.0	$82.2^{\dagger}$	$74.5^{\dagger}$	<u>76.5</u>±0.3
Auto-FedRL(MLP)	73.2	$81.2^{\dagger}$	$75.3^{\dagger}$	**76.6**±0.3
Centralized	74.5	82.6	72.0	76.3

training is shrunk to only 0.7%. Figure 6 presents the analysis of learning process in our best performing model. As shown in Fig. 6(a), we can observe that the RL agent is able to naturally form the training scheduler for each hyperparameter (*e.g.*, the learning rate decay for the client/server), which is aligned with the theoretical analysis about the convergence on non-iid data of FL algorithms [29]. Since Auto-FedAvg specially aims to learn the optimal aggregation weights, we compare the aggregation weights learning process between our approach and Auto-FedAvg in Fig. 6(b). It can be observed that the two methods exhibit a similar trend of learning aggregation weights, which further demonstrates the effectiveness of Auto-FedRL in aggregation weights searching. Finally, we use FANOVA [18] to assess the hyperparameter importance. As shown in Fig. 6(c), LR, SLR, and AW1 rank as the top-3 most important hyperparameters, which implies the necessity of tuning server-side hyperparameters in FL setting.

Multi-institutional Pancreas Segmentation. Table 4 and Fig. 5(b) present the quantitative and qualitative results on this dataset, respectively. Similar to the results on COVID-19 segmentation, our Auto-FedRL algorithms achieves the significantly better overall performance. In particular, Auto-FedRL(MLP) outperforms the best counterpart by 1.3%. We aslo observe that our methods exhibits better generalizability on the relatively smaller clients. Specifically, on Client III, Auto-FedRL(MLP) improves the Dice score from 51.1% to 75.3%, which is even 3.28% higher than the centralized training. These results implies that by leveraging the dynamic hyperparameter tuning during the training, Auto-FedRL algorithms can achieve better generalization and are more robust towards the heterogeneous data distribution. As shown in Fig. 5(b), the proposed methods have a better capacity of handling the challenging cases, which is consistent with the quantitative results. The detailed hyperparameter evolution analysis on pancreas segmentation is provided in the supplementary material.

4.3   Ablation Study

The effectiveness of the proposed continuous search and NN-based RL agent is demonstrated by the previous sets of experiments in three datasets. Here, we

Table 5. The search space ablation study on CIFAR-10.

Search Space				Search Strategy		
LR	LE	AW	SLR	Discrete	Continuous	Continuous MLP
✓				89.83	90.02	**90.12**
✓	✓			89.86	90.10	**90.49**
✓	✓	✓		90.43	90.52	**90.87**
✓	✓	✓	✓	90.70	90.85	**91.27**

conduct a detailed ablation study to analyze the benefit of individually adding each hyperparameter into the search space. As shown in Table 5, the performance of trained global model can be further improved with the expansion of the search space, which also validates our motivation that the proper hyperparameter tuning is crucial for the success of FL algorithms. More visualizations, experimental results, and the theoretical analysis to guarantee the convergence of Auto-FedRL are provided in the supplementary material.

5   Conclusion and Discussion

In this work, we proposed an online RL-based federated hyperparameter optimization framework for realistic FL applications, which can dynamically tune the hyperparameters during a single trial, resulting in improved performance compared to several existing baselines. To make federated hyperparameter optimization more practical in real-world applications, we proposed Auto-FedRL(CS) and Auto-FedRL(MLP), which can operate on continuous search space, demand nearly constant memory and are computationally efficient. By integrating the adaptive federated optimization, Auto-FedRL supports a more flexible search space to tune a wide range of hyperparameters. The empirical results on three datasets with diverse characteristics reveal that the proposed method can train global models with better performance and generalization capabilities under heterogeneous data distributions.

While our proposed method yielded a competitive performance, there are potential areas for improvement. First, we are aware that the performance improvement brought by the proposed method is not uniform across participating clients. Since the proposed RL agent jointly optimizes the whole system, minimizing an aggregate loss can lead to potentially advantage or disadvantage a particular client's performance. We can also observe that all FL methods exhibit a relatively small improvement on the client with the largest amount of data. This is a common phenomenon of FL methods since the client itself already provides diverse and rich training and testing data. Future research could include additional fairness constraints [28,26,64,33,36] to achieve a more uniform performance distribution across clients and reduce potential biases. Second, the NN-based RL agent could be benefiting from transfer learning. The effectiveness of RL transfer learning has been demonstrated in the literature for related tasks [54]. Pre-training the NN-based agent on large-scale FL datasets and then finetuning on target tasks may further boost the performance of our approach.

References

1. Abdallah, S., Kaisers, M.: Addressing environment non-stationarity by repeating q-learning updates. The Journal of Machine Learning Research **17**(1), 1582–1612 (2016)
2. Acar, D.A.E., Zhao, Y., Matas, R., Mattina, M., Whatmough, P., Saligrama, V.: Federated learning based on dynamic regularization. In: International Conference on Learning Representations (2020)
3. Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M.W., Pfau, D., Schaul, T., Shillingford, B., De Freitas, N.: Learning to learn by gradient descent by gradient descent. In: Advances in neural information processing systems. pp. 3981–3989 (2016)
4. Baker, B., Gupta, O., Naik, N., Raskar, R.: Designing neural network architectures using reinforcement learning. arXiv preprint arXiv:1611.02167 (2016)
5. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. Advances in neural information processing systems **24** (2011)
6. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of machine learning research **13**(2) (2012)
7. Chen, X., Chen, T., Sun, H., Wu, Z.S., Hong, M.: Distributed training with heterogeneous data: Bridging median-and mean-based algorithms. arXiv preprint arXiv:1906.01736 (2019)
8. Chen, X., Xie, L., Wu, J., Tian, Q.: Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1294–1303 (2019)
9. Chopra, A., et al.: Adasplit: Adaptive trade-offs for resource-constrained distributed deep learning. arXiv preprint arXiv:2112.01637 (2021)
10. Çiçek, Ö., Abdulkadir, A., Lienkamp, S.S., Brox, T., Ronneberger, O.: 3d u-net: learning dense volumetric segmentation from sparse annotation. In: International conference on medical image computing and computer-assisted intervention. pp. 424–432. Springer (2016)
11. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation policies from data. arXiv preprint arXiv:1805.09501 (2018)
12. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 702–703 (2020)
13. Geiping, J., Bauermeister, H., Dröge, H., Moeller, M.: Inverting gradients–how easy is it to break privacy in federated learning? arXiv preprint arXiv:2003.14053 (2020)
14. Guo, P., Unberath, M., Heo, H.Y., Eberhardt, C., Lim, M., Blakeley, J., Jiang, S.: Learning-based analysis of amide proton transfer-weighted mri to identify tumor progression in patients with post-treatment malignant gliomas. Available at SSRN 4049653
15. Guo, P., Wang, P., Zhou, J., Jiang, S., Patel, V.M.: Multi-institutional collaborations for improving deep learning-based magnetic resonance image reconstruction using federated learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2423–2432 (2021)
16. Harmon, S.A., Sanford, T.H., Xu, S., Turkbey, E.B., Roth, H., Xu, Z., Yang, D., Myronenko, A., Anderson, V., Amalou, A., et al.: Artificial intelligence for the detection of covid-19 pneumonia on chest ct using multinational datasets. Nature communications **11**(1), 1–7 (2020)

17. Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D.: Deep reinforcement learning that matters. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)

18. Hutter, F., Hoos, H., Leyton-Brown, K.: An efficient approach for assessing hyperparameter importance. In: Proceedings of the 31st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 32, pp. 754–762. PMLR (2014)

19. Jaakkola, T., Singh, S.P., Jordan, M.I.: Reinforcement learning algorithm for partially observable markov decision problems. Advances in neural information processing systems pp. 345–352 (1995)

20. Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al.: Advances and open problems in federated learning. arXiv preprint arXiv:1912.04977 (2019)

21. Kaissis, G., Ziller, A., Passerat-Palmbach, J., Ryffel, T., Usynin, D., Trask, A., Lima, I., Mancuso, J., Jungmann, F., Steinborn, M.M., et al.: End-to-end privacy preserving deep learning on multi-institutional medical imaging. Nature Machine Intelligence **3**(6), 473–484 (2021)

22. Kaissis, G.A., Makowski, M.R., Rückert, D., Braren, R.F.: Secure, privacy-preserving and federated machine learning in medical imaging. Nature Machine Intelligence **2**(6), 305–311 (2020)

23. Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S., Stich, S., Suresh, A.T.: Scaffold: Stochastic controlled averaging for federated learning. In: International Conference on Machine Learning. pp. 5132–5143. PMLR (2020)

24. Khodak, M., Tu, R., Li, T., Li, L., Balcan, M.F.F., Smith, V., Talwalkar, A.: Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing. Advances in Neural Information Processing Systems **34** (2021)

25. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)

26. Li, T., Hu, S., Beirami, A., Smith, V.: Ditto: Fair and robust federated learning through personalization. In: International Conference on Machine Learning. pp. 6357–6368. PMLR (2021)

27. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. arXiv preprint arXiv:1812.06127 (2018)

28. Li, T., Sanjabi, M., Beirami, A., Smith, V.: Fair resource allocation in federated learning. arXiv preprint arXiv:1905.10497 (2019)

29. Li, X., Huang, K., Yang, W., Wang, S., Zhang, Z.: On the convergence of fedavg on non-iid data. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=HJxNAnVtDS

30. Li, X., Gu, Y., Dvornek, N., Staib, L.H., Ventola, P., Duncan, J.S.: Multi-site fmri analysis using privacy-preserving federated learning and domain adaptation: Abide results. Medical Image Analysis **65**, 101765 (2020)

31. Li, X., Jiang, M., Zhang, X., Kamp, M., Dou, Q.: Fedbn: Federated learning on non-iid features via local batch normalization. arXiv preprint arXiv:2102.07623 (2021)

32. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018)

33. Lyu, L., Xu, X., Wang, Q., Yu, H.: Collaborative fairness in federated learning. In: Federated Learning, pp. 189–204. Springer (2020)

34. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017)

35. Mei, Y., Guo, P., Patel, V.M.: Escaping data scarcity for high-resolution heterogeneous face hallucination. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18676–18686 (2022)
36. Michieli, U., Ozay, M.: Are all users treated fairly in federated learning systems? In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2318–2322 (2021)
37. Mohri, M., Sivek, G., Suresh, A.T.: Agnostic federated learning. In: International Conference on Machine Learning. pp. 4615–4625. PMLR (2019)
38. Mostafa, H.: Robust federated learning through representation matching and adaptive hyper-parameters. arXiv preprint arXiv:1912.13075 (2019)
39. Nvidia Corporation: Nvidia FLARE (6 2022). https://doi.org/10.5281/zenodo.6780567, https://github.com/NVIDIA/nvflare
40. Padakandla, S., Prabuchandran, K., Bhatnagar, S.: Reinforcement learning algorithm for non-stationary environments. Applied Intelligence 50(11), 3590–3606 (2020)
41. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: Proceedings of the aaai conference on artificial intelligence. vol. 33, pp. 4780–4789 (2019)
42. Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečnỳ, J., Kumar, S., McMahan, H.B.: Adaptive federated optimization. arXiv preprint arXiv:2003.00295 (2020)
43. Rieke, N., Hancox, J., Li, W., Milletari, F., Roth, H.R., Albarqouni, S., Bakas, S., Galtier, M.N., Landman, B.A., Maier-Hein, K., et al.: The future of digital health with federated learning. NPJ digital medicine 3(1), 1–7 (2020)
44. Roth, H.R., Chang, K., Singh, P., Neumark, N., Li, W., Gupta, V., Gupta, S., Qu, L., Ihsani, A., Bizzo, B.C., et al.: Federated learning for breast density classification: A real-world implementation. In: Domain Adaptation and Representation Transfer, and Distributed and Collaborative Learning, pp. 181–191. Springer (2020)
45. Roth, H.R., Lu, L., Farag, A., Shin, H.C., Liu, J., Turkbey, E.B., Summers, R.M.: Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation. In: International conference on medical image computing and computer-assisted intervention. pp. 556–564. Springer (2015)
46. Roth, H.R., Xu, Z., Diez, C.T., Jacob, R.S., Zember, J., Molto, J., Li, W., Xu, S., Turkbey, B., Turkbey, E., et al.: Rapid artificial intelligence solutions in a pandemic-the covid-19-20 lung ct lesion segmentation challenge. Research Square (2021)
47. Ruvolo, P., Fasel, I., Movellan, J.: Optimization on a budget: A reinforcement learning approach. Advances in Neural Information Processing Systems 21 (2008)
48. Sattler, F., Wiedemann, S., Müller, K.R., Samek, W.: Robust and communication-efficient federated learning from non-iid data. IEEE transactions on neural networks and learning systems 31(9), 3400–3413 (2019)
49. Shaw, W.T.: Sampling student's t distribution-use of the inverse cumulative distribution function. Journal of Computational Finance 9(4), 37 (2006)
50. Sheller, M.J., Edwards, B., Reina, G.A., Martin, J., Pati, S., Kotrotsou, A., Milchenko, M., Xu, W., Marcus, D., Colen, R.R., et al.: Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. Scientific reports 10(1), 1–12 (2020)
51. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

52. Simpson, A.L., Antonelli, M., Bakas, S., Bilello, M., Farahani, K., Van Ginneken, B., Kopp-Schneider, A., Landman, B.A., Litjens, G., Menze, B., et al.: A large annotated medical image dataset for the development and evaluation of segmentation algorithms. arXiv preprint arXiv:1902.09063 (2019)
53. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems **25** (2012)
54. Taylor, M.E., Stone, P.: Transfer learning for reinforcement learning domains: A survey. Journal of Machine Learning Research **10**(7) (2009)
55. Thistleton, W.J., Marsh, J.A., Nelson, K., Tsallis, C.: Generalized box–müller method for generating q-gaussian random deviates. IEEE transactions on information theory **53**(12), 4805–4810 (2007)
56. Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., Khazaeni, Y.: Federated learning with matched averaging. arXiv preprint arXiv:2002.06440 (2020)
57. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning **8**(3), 229–256 (1992)
58. Wu, J., Chen, X.Y., Zhang, H., Xiong, L.D., Lei, H., Deng, S.H.: Hyperparameter optimization for machine learning models based on bayesian optimization. Journal of Electronic Science and Technology **17**(1), 26–40 (2019)
59. Xia, Y., Yang, D., Li, W., Myronenko, A., Xu, D., Obinata, H., Mori, H., An, P., Harmon, S., Turkbey, E., et al.: Auto-fedavg: Learnable federated averaging for multi-institutional medical image segmentation. arXiv preprint arXiv:2104.10195 (2021)
60. Xie, L., Yuille, A.: Genetic cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 1379–1388 (2017)
61. Xu, A., Li, W., Guo, P., Yang, D., Roth, H.R., Hatamizadeh, A., Zhao, C., Xu, D., Huang, H., Xu, Z.: Closing the generalization gap of cross-silo federated medical image segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20866–20875 (2022)
62. Yang, D., Xu, Z., Li, W., Myronenko, A., Roth, H.R., Harmon, S., Xu, S., Turkbey, B., Turkbey, E., Wang, X., et al.: Federated semi-supervised learning for covid region segmentation in chest ct using multi-national data from china, italy, japan. Medical image analysis **70**, 101992 (2021)
63. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST) **10**(2), 1–19 (2019)
64. Yu, H., Liu, Z., Liu, Y., Chen, T., Cong, M., Weng, X., Niyato, D., Yang, Q.: A fairness-aware incentive scheme for federated learning. In: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society. pp. 393–399 (2020)
65. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-iid data. arXiv preprint arXiv:1806.00582 (2018)
66. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578 (2016)