Auto-FedRL: Federated Hyperparameter Optimization for Multi-institutional Medical Image Segmentation
——*Supplementary Material*——

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

1 Supplementary Introduction

In this supplementary document, we first perform a convergence analysis of federated learning under the proposed federated hyperparameter optimization framework (Sec. 2.1) and then provide details of the network architectures used for our classification (Sec. 2.2) and segmentation experiments. Finally, we analyze the learning process for the pancreas segmentation task (Sec. 3.1).

2 Supplementary Method

2.1 Convergence Analysis

In Eq. 1 of main manuscript, we define the FL optimization problem as follows:

$$\min_{x \in R^d} \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}_i(x),$$

where m is the number of clients and $\mathcal{L}_i(x) = \mathbb{E}_{z \sim \mathcal{D}_i}[f_i(x, z)]$ is the loss function of the i^{th} client. $z \in \mathcal{Z}$, and $\mathcal{D}_i$ represents the data distribution of the i^{th} client. Following the proof originally proposed in adaptive federated optimization [6], we have the unbiased stochastic gradient $g_i(x)$ and the client's true gradient $\nabla \mathcal{L}_i(x)$. Then we make the following three common assumptions:

* Work done during an internship at NVIDIA.

Assumption 1 (Lipschitz Gradient):

$$||\nabla \mathcal{L}_i(x) - \nabla \mathcal{L}_i(y)|| \leq L||x-y||, \forall x, y \in R^d$$

Assumption 2 (Bounded Global Variance):

$$(\frac{1}{m})\sum_{i=1}^{m} ||\nabla[\mathcal{L}_i(x)]_j - [\nabla f(x)]_j||^2 \leq \sigma_{g,j}^2,$$
$$\forall x \in R^d \text{ and } j \in [d]$$

Assumption 3 (Bounded Gradients):

$$\text{For any } i \in [m], x \in R^d \text{ and } z \in \mathcal{Z},$$
$$\text{We have } |[\nabla f_i(x,z)]_j| \leq G, \forall j \in [d]$$

As discussed in [6], the three assumptions are widely adopted in the non-convex optimization [7,8,5] and federated learning literature [4,10]. For the illustration purpose, we assume the server optimizer is the commonly used Adam optimizer. Let $\sigma^2 = \sigma_l^2 + 6K\sigma_g^2$, where $\sigma_l^2 = \sum_{j=1}^{d} \sigma_{l,j}^2$ and $\sigma_g^2 = \sum_{j=1}^{d} \sigma_{g,j}^2$. Suppose the client learning γ_l is bounded by the search space and satisfies $\gamma_l \leq \frac{1}{16}LK$ and

$$\gamma_l \leq \frac{1}{6K} \min\left\{ \left[\frac{\alpha}{GL}\right]^{1/2}, \left[\frac{\alpha^2}{GL^3\gamma}\right]^{1/4}, \left[\frac{\alpha}{GL^2}\right]^{1/3} \right\},$$

where α controls the algorithms' degree of adaptivity. To highlight the dependency of K (the number of clients) and Q (the number of rounds) for the convergence rate, we can assume γ_l, γ and α are specifically chosen as follows:

$$\gamma_l = \Theta(\frac{1}{KL\sqrt{Q}}),$$
$$\gamma = \Theta(\sqrt{KM}),$$
$$\alpha = \frac{G}{L}.$$

Based on the proof of FedAdam [6], when Q is sufficiently large, the proposed methods satisfies:

$$\min_{0 \leq q \leq Q-1} \mathbb{E}||\nabla f(x_q)||^2 = \mathcal{O}\Bigg(\frac{f(x_0) - f(x^\star)}{\sqrt{mKQ}} + \frac{2\sigma_l^2 L}{G^2\sqrt{mKQ}} + \frac{\sigma^2}{GKQ} + \frac{\sigma^2 L\sqrt{m}}{G^2\sqrt{K}Q^{3/2}} \Bigg).$$

Hence, when $Q \gg K$, the proposed method can achieve a convergence rate of $\mathcal{O}(\frac{1}{\sqrt{mKQ}})$ under the adaptive federated optimization framework. Readers are referred to [6] for a complete convergence analysis of the adaptive federated optimization.

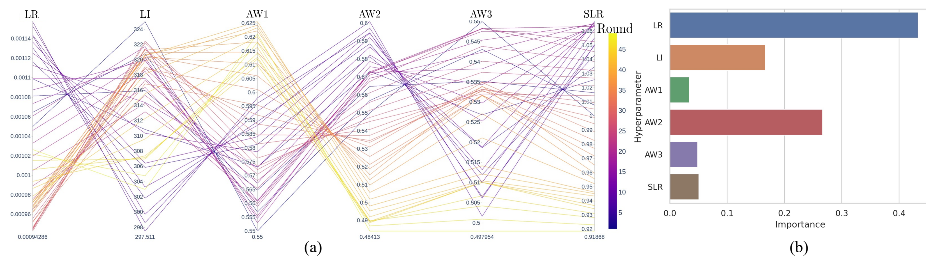

Fig. 1. Analysis of the learning process of Auto-FedRL(css) in Pancreas CT segmentation. (a) The parallel plot of the hyperparameter change during the training. LR, LI, AW, and SLR demotes the learning rate, local iterations, aggregation weights, and the server learning rate, respectively. (b) The importance analysis of different hyperparameters.

Table 1. Configuration of 3D Unet

Network	Block	In Channel	Out Channel
Encoder	ConvBlock	1	16
	ResConvBlockWD	16	32
	ResConvBlock	32	32
	ResConvBlockD	32	64
	ResConvBlock	64	64
	ResConvBlockWD	64	128
	ResConvBlock	128	128
	ResConvBlockWD	128	256
	ResConvBlock	256	256
Decoder	UpBlock	256	128
	UpBlock	128	64
	UpBlock	64	32
	UpBlock	32	16
	Conv3d	16	1

2.2 Network Architectures

We use a 3D U-Net [2] style encoder-decoder architecture for the segmentation networks. The encoder and decoder networks can be described as shown in Table 1, where ResConvBlockWD represents a 3D ResConvBlock with down-sampling layer and network modules are expressed by (in-channel, out-channel). Table 2 shows the details of each block in our segmentation network. The VGG-9 [9] architecture used for CIFAR-10 experiments is presented in Table 3. For the Auto-FedRL(MLP), due to our online setting, we have to keep the learnable parameters in networks small but effective. The MLP can be described as following: Liner(in-chanel, 256)-ReLu(256)-Liner(256, 256)-ReLu(256)-Liner(256,in-chanel), where in-chanel is decided by the size of mean vector μ and the covariance matrix Σ.

3 Supplementary Results

3.1 Learning Process for Pancreas Segmentation

Figure 1 presents the learning process of our best performing model in the pancreas segmentation task. As shown in Fig. 1(a), while in this task the number

Table 2. Configuration of Blocks in 3D Unet

Block	Layer	Kernel size	Stride	Padding
ConvBlock	Conv3D	3	2	1
	InstanceNorm	-	-	-
	ReLu	-	-	-
ResConvBlock	Conv3D	3	1	1
	InstanceNorm	-	-	-
	ReLu	-	-	-
ResConvBlockD	Conv3D	3	2	1
	InstanceNorm	-	-	-
	ReLu	-	-	-
	Conv3D	1	2	0
UpSample	Conv3D	3	1	1
	InstanceNorm	-	-	-
	ReLu	-	-	-
	Interpolate	-	-	-

Table 3. Configuration of VGG-9

Block	Layer	In Channel	Out Channel	Kernel size	Stride	Padding
ConvBlock1	Conv2D	3	32	3	1	1
	ReLu	32	32	-	-	-
	Conv2D	32	64	3	1	1
	ReLu	64	64	-	-	-
	MaxPool2d	64	64	2	2	0
ConvBlock2	Conv2D	64	128	3	1	1
	ReLu	128	128	-	-	-
	Conv2D	128	128	3	1	1
	ReLu	128	128	-	-	-
	MaxPool2d	128	128	2	2	0
	Dropout2d	-	-	-	-	-
ConvBlock3	Conv2D	128	256	3	1	1
	ReLu	256	256	-	-	-
	Conv2D	256	256	3	1	1
	ReLu	256	256	-	-	-
	MaxPool2d	256	256	2	2	0
FC	Dropout	-	-	-	-	-
	Linear	4096	512	-	-	-
	ReLU	512	512	-	-	-
	Linear	512	512	-	-	-
	ReLU	512	512	-	-	-
	Dropout	512	512	-	-	-
	Linear	512	10	-	-	-

Table 4. The additional computational details of different search strategies under the same setting on CIFAR-10.

Search Space Type	Accuracy	Memory Usage	Running Time for Search	C3-Score [1]
Discrete	90.70	42.8 GB	8.246 s	0.778
Continuous	90.85	**3.00 GB**	**0.012 s**	<u>0.799</u>
Continuous MLP	91.27	<u>3.13 GB</u>	<u>0.019 s</u>	**0.803**

of optimization steps is quite limited (*i.e.*, 50), we still can observe that the RL agent is able to naturally form the training scheduler for each hyperparameter (*e.g.*, the learning rate for clients and the server). Similar as the analysis of COVID-19 lesions segmentation, we use FANOVA [3] to assess the hyperparameter importance. As shown in Fig .1(b), LR, AW2, and LI rank as top-3 most important hyperparameters, which implies that including aggregation weights into search space is also important in our setting.

References

1. Chopra, A., et al.: Adasplit: Adaptive trade-offs for resource-constrained distributed deep learning. arXiv preprint arXiv:2112.01637 (2021)
2. 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)
3. 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)
4. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. arXiv preprint arXiv:1812.06127 (2018)
5. Reddi, S., Zaheer, M., Sachan, D., Kale, S., Kumar, S.: Adaptive methods for nonconvex optimization. In: Proceeding of 32nd Conference on Neural Information Processing Systems (NIPS 2018) (2018)
6. 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)
7. Reddi, S.J., Hefny, A., Sra, S., Poczos, B., Smola, A.: Stochastic variance reduction for nonconvex optimization. In: International conference on machine learning. pp. 314–323. PMLR (2016)
8. Reddi, S.J., Hefny, A., Sra, S., Poczos, B., Smola, A.: Stochastic variance reduction for nonconvex optimization. In: International conference on machine learning. pp. 314–323. PMLR (2016)
9. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
10. Wang, S., Tuor, T., Salonidis, T., Leung, K.K., Makaya, C., He, T., Chan, K.: Adaptive federated learning in resource constrained edge computing systems. IEEE Journal on Selected Areas in Communications **37**(6), 1205–1221 (2019)