QUALITY-ADAPTIVE SPLIT-FEDERATED LEARNING FOR SEGMENTING MEDICAL IMAGES WITH INACCURATE ANNOTATIONS

Zahra Hafezi Kafshgari, Chamani Shiranthika, Parvaneh Saeedi, and Ivan V. Bajić

School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada

2023 IEEE 20th International Symposium on Biomedical Imaging (ISBI) | 978-1-6654-7358-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISBI53787.2023.10230484

ABSTRACT

SplitFed Learning, a combination of Federated and Split Learning (FL and SL), is one of the most recent developments in the decentralized machine learning domain. In SplitFed learning, a model is trained by clients and a server collaboratively. For image segmentation, labels are created at each client independently and, therefore, are subject to clients' bias, inaccuracies, and inconsistencies. In this paper, we propose a data quality-based adaptive averaging strategy for SplitFed learning, called QA-SplitFed, to cope with the variation of annotated ground truth (GT) quality over multiple clients. The proposed method is compared against five state-of-the-art model averaging methods on the task of learning human embryo image segmentation. Our experiments show that all five baseline methods fail to maintain accuracy as the number of corrupted clients increases. QA-SplitFed, however, copes effectively with corruption as long as there is at least one uncorrupted client.

Index Terms— SplitFed learning, privacy, poor annotation, ground truth corruption, data heterogeneity

1. INTRODUCTION

Decentralized learning decouples data from the computational infrastructure, and removes the need for sharing data, in both training and inference. Instead, features derived from the data at various levels of processing are shared among clients and servers. Besides benefits to privacy, decentralized learning could scale to the available resources, both human and computational, available at different locations. This makes it attractive for medical applications, where clients may be hospitals, clinics, and medical centers. SplitFed Learning (SFL) is a combination of two decentralized learning methods: Split Learning [1], and Federated Learning [2]. In FL, clients keep their own data, but must train their local full models and send their model weights to the server for averaging [3]. Hence, in FL, each client must have sufficiently powerful hardware to train its own model. SL methods shift the computational burden from the client to the server. In SL, the model is split into two or more parts, and training is done collaboratively by clients and a server communicating the activations and gradient updates of the split layers [4]. SFL takes advantage of both FL and SL, and addresses their shortcomings [5]. Specifically, in SFL, each client's data stays on the client and most of the computational burden is offloaded to the server.

In this paper, we study SFL in the case where data annotation quality differs across clients. Specifically, we focus on embryo image segmentation, where ground truth segmentation accuracy may vary from client to client, depending on the expertise and human resources available at each client. We propose a model averaging procedure that is significantly more robust in maintaining the global accuracy to such variation in annotation quality compared to existing methods. Related work is reviewed in Section 2. Proposed method is presented in Section 3, followed by experiments in Section 4 and conclusions in Section 5.

2. RELATED WORKS

Different types of data heterogeneity could occur in the context of collaborative learning, e.g. unbalanced data [2], different feature map distributions [6], or different levels of annotation accuracies over clients [7]. In the literature, these are commonly referred to as non-IID (Independent and Identically Distributed) scenarios. McMahan *et al.* [2] first introduced the concept of FL by averaging all local model weights to update a global model. Their model averaging procedure, called FedAVG, assigned a weight proportional to the number of data samples at each client. Several extensions to FedAVG exist in the literature. FedAVG-M is an optimized version of FedAVG that applies a momentum on the server optimizer to deal with Non-IID clients [8]. Shi *et al.* [9] introduced Fedensemble to average *k* randomly trained models in the federation step to reduce the variance of the global model. Fang *et al.* [10] studied heterogeneous and noisy labels in FL. Noisy clients were created by symmetric and pair flip noise, and a public dataset was used to adjust the global model. For medical applications, accessing patients' public data is difficult due to privacy concerns. Yang *et al.* [11] proposed a method in which noisy labels were created in a similar way to those in [10]. The average of the local feature vectors per client, called centriods, were shared with the server for model aggregation and label correction. In [12], noisy labels were created by shuffling labels randomly and adding Gaussian noise to the input data. The weighted sum of empirical risks was then minimized for clients with regularization terms. Considering statistical data heterogeneity in FL, FedProx was proposed in [13] to stabilize the final model by a proximal term added to the optimization function.

The idea of combining FL and SL into SplitFed (or SFL) was first introduced by Chandra *et al.* [5]. They proposed two versions of SplitFed: vanilla SplitFed (SplitFedv1) and U-shaped SplitFed (SplitFedv2). In vanilla SplitFed, labels are shared with the server, while in the U-shaped version, all data is kept private on the clients. Simple arithmetic averaging of model weights was used in both cases. In the healthcare domain, Poirot *et al.* [14] first utilized U-shaped SplitFed in experiments on fundus photos and chest X-rays. Gawali *et al.* [15] reviewed FL, SL, and SplitFed approaches in the medical domain, and introduced SpltFedv3, which was a label sharing approach (similar to SpliFedv1). SpltFedv3 did not change the client-side model weights, and only applied the federation on the server-side model weights, while still using simple averaging.

SplitAVG, a very recent work, was proposed to handle heterogeneous clients whose feature maps differ [6]. Feature maps from different clients are concatenated and fed to the server simultaneously. No averaging is performed on client-side models. Chen *et al.* [16] studied another type of heterogeneity in terms of energy harvesting capability per client. Their proposed SplitFed algorithm was tested on CIFAR-10 and CIFAR-100 datasets. A weighted averaging strategy was proposed in [7] that took advantage of labeled data for creating pseudo labels for unlabeled data. It used clients with better training quality when updating the global model. This method was applied on medical image segmentation task.

In this paper, a novel method for SplitFed model averaging, called QA-SplitFed, is proposed to address scenarios where not all clients' labels are annotated correctly. Model weights are updated twice in each global epoch based on the local and global behavior of all clients. QA-SplitFed is applied to training an embryo image segmentation model. To assess its robustness, a comparison with five baseline averaging methods is presented.

3. PROPOSED METHOD

3.1. Human embryo components segmentation model

We study training of a SplitFed U-Net, as in Fig. 1, for segmentation of human embryo components in microscopic images. The overall U-Net consists of five downsampling blocks between the input and the bottleneck, and five upsampling blocks between the bottleneck and the output. Each block contains two convolutional layers with the kernel size of 3×3, followed by batch normalization and `ReLU` activation. Each downsampling block ends with a 2×2 max-pooling layer; the number of filters in downsampling blocks is 32, 64, 128, 256, and 512, respectively. Each upsampling block starts with a 2×2 upsampling layer, followed by tanspose convolution layers; the number of filters in upsampling blocks is 512, 256, 128, 64, and 32, respectively. The last convolutional layer is followed by an output `argmax` layer. The model is split such that the first convolutional layer (front-end, FE) and the last two convolutional layers together with the output `argmax` layer (back-end, BE) are on the client side, while the rest of the model is on the server.

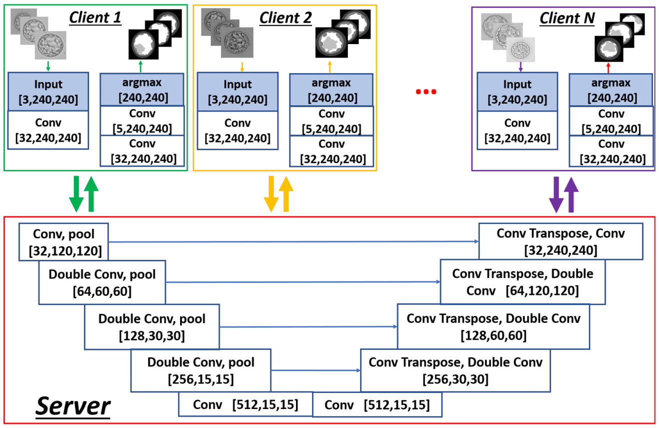

Fig. 1: Our SplitFed architecture.

3.2. Training procedure and averaging

The system adopts sequential training, where the server trains with each of the N clients in turn. At the end of this training process, both client and server store their weights at the server side. Let's W_i^{S} be the server weights while training with client i, and $W_i^{\text{C}} = \{W_i^{\text{FE}}, W_i^{\text{BE}}\}$ be the weights of client i, which include both FE and BE weights, from the same epoch. One pass across all clients is called a *global epoch*. After all clients get trained locally, W_i^{S} and W_i^{C} are averaged into $\overline{W}^{\text{S}}$ and $\overline{W}^{\text{C}}$, respectively, to create a global model of $W = \{\overline{W}^{\text{S}}, \overline{W}^{\text{C}}\}$. $\overline{W}^{\text{C}}$ is sent back to all clients and local validation is performed on the global model. Considering the validation loss statistics, global model is updated one more time before starting the next global epoch.

Within each local epoch, in the forward pass, features produced by FE are sent to the server. The server then processes them using its part of the model. The server features are sent to the client-side's BE. In the back-propagation pass, gradient updates from the BE are sent to the server, and back-propagated through the server-side part of the model to FE.

Algorithm 1 describes the QA-SplitFed averaging method. Its inputs are: N (the number of clients), G (the number of global epochs), E (the number of local epochs), $\mathbf{d}^t = [m_1, ..., m_N]^\top / (\sum_i m_i)$ (the vector of training data scores), and $\mathbf{d}^v = [n_1, ..., n_N]^\top / (\sum_i n_i)$ (the vector of validation data scores). Its outputs are $\overline{W}^{\text{C}}$ (global client model) and $\overline{W}^{\text{S}}$ (global server model).

At each global epoch, after each client i is trained locally, the mean training loss value μ_i^t at the best local epoch is calculated. To have a better understanding of the loss values

Algorithm 1: QA-SplitFed

```
Inputs:
    $N$: Number of clients
    $G$: Number of global epochs
    $E$: Number of local epochs
    $\mathbf{d}^t$: Vector of training data scores
    $\mathbf{d}^v$: Vector of validation data scores
Outputs:
    $\overline{W}^{\mathrm{C}} = \{\overline{W}^{\mathrm{FE}}, \overline{W}^{\mathrm{BE}}\}$: Global client model
    $\overline{W}^{\mathrm{S}}$: Global server model
for $g = 1$ to $G$ do
    for $i = 1$ to $N$ do
        Local training and validation over $E$ epochs
        $W_i = \{W_i^{\mathrm{C}}, W_i^{\mathrm{S}}\} \leftarrow$ model's weights from
          epoch with the lowest validation loss
        $\mathcal{L}_i^j \leftarrow$ loss of $W_i$ at training sample $j$
        $\mu_i^t \leftarrow \texttt{mean}\{\mathcal{L}_i^1, ..., \mathcal{L}_i^{m_i}\}$
        $\sigma_i^t \leftarrow \texttt{std}\{\mathcal{L}_i^1, ..., \mathcal{L}_i^{m_i}\}$
        $b_i^t \leftarrow \mu_i^t + 2\sigma_i^t$
    $\mathbf{b}^t \leftarrow [b_1^t, ..., b_N^t]^\top$
    $\mathbf{W} \leftarrow [W_1, ..., W_N]^\top$
    $\{\overline{W}^{\mathrm{C}}, \overline{W}^{\mathrm{S}}\} \leftarrow \texttt{ModelUpdates}(\mathbf{W}, \mathbf{b}^t, \mathbf{d}^t)$
    for $i = 1$ to $N$ do
        $\mathcal{L}_i^j \leftarrow$ loss of $\{\overline{W}^{\mathrm{C}}, \overline{W}^{\mathrm{S}}\}$ at valid. sample $j$
        $\mu_i^v \leftarrow \texttt{mean}\{\mathcal{L}_i^1, ..., \mathcal{L}_i^{n_i}\}$
        $\sigma_i^v \leftarrow \texttt{std}\{\mathcal{L}_i^1, ..., \mathcal{L}_i^{n_i}\}$
        $b_i^v \leftarrow \mu_i^v + 2\sigma_i^v$
    $\mathbf{b}^v \leftarrow [b_1^v, ..., b_N^v]^\top$
    $\{\overline{W}^{\mathrm{C}}, \overline{W}^{\mathrm{S}}\} \leftarrow \texttt{ModelUpdates}(\mathbf{W}, \mathbf{b}^v, \mathbf{d}^v)$
function ModelUpdates $(\mathbf{W}, \mathbf{b}, \mathbf{d})$
    $\mathbf{q} \leftarrow \texttt{softmax}[1/\mathbf{b}] = [q_1, ..., q_N]^\top$
    $\mathbf{r} \leftarrow \frac{\mathbf{q} \odot \mathbf{d}}{\mathbf{q}^\top \mathbf{d}} = [r_1, ..., r_N]^\top$
    $\overline{W}^{\mathrm{C}} \leftarrow \sum_{i=1}^N r_i W_i^{\mathrm{C}}, \quad \overline{W}^{\mathrm{S}} \leftarrow \sum_{i=1}^N r_i W_i^{\mathrm{S}}$
```

and their distributions, the standard deviation σ_i^t of the loss values at the best epoch are also calculated. The mean loss value based on a relatively few data samples is unreliable, so we take an upper bound of a 95% confidence interval for the mean. If the loss values were normally distributed, this upper bout would be $b_i^t = \mu_i^t + 2\sigma_i^t$. Hence, b_i^t is an indicator of the (un)reliability of client i.

After one round of local training of all clients, two vectors, $\mathbf{b}^t = [b_1^t, ..., b_N^t]^\top$ and $\mathbf{q}^t = \texttt{softmax}[1/\mathbf{b}^t]$, are created. A smaller b_i^t, and thus a larger q_i^t, represents a more reliable client. In addition to $\mathbf{b}^t$ and $\mathbf{q}^t$, we use the training data score vector $\mathbf{d}^t = [m_1, ..., m_N]^\top / \sum_i m_i$, similar to [2, 8], which specifies the training data distribution across clients. By combining clients' training quality scores $\mathbf{q}^t$ and training data scores $\mathbf{d}^t$, averaging weights are calculated as $\mathbf{r} = \frac{\mathbf{q}^t \odot \mathbf{d}^t}{\mathbf{q}^{t\top} \mathbf{d}^t}$, where $\odot$ is the Hadamard (element-wise) product. Note that $\sum_i r_i = 1$. Then, averaging is performed as: $\overline{W}^{\mathrm{S}} = \sum_{i=1}^N r_i W_i^{\mathrm{S}}, \quad \overline{W}^{\mathrm{C}} = \sum_{i=1}^N r_i W_i^{\mathrm{C}}$.

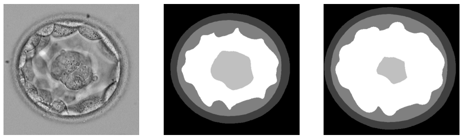

Fig. 2: Left to right: input image, GT mask, corrupted mask.

$\overline{W}^{\mathrm{C}}$ is broadcast to all clients and the server model is updated to $\overline{W}^{\mathrm{S}}$. Local validation is performed using the new local and global models, to assess the performance on each client's validation data. Then the above averaging process is repeated, but with validation vectors $\mathbf{b}^v$ and $\mathbf{d}^v$ taking the place of $\mathbf{b}^t$ and $\mathbf{d}^t$. This process is repeated G times. The best global model is chosen according to the global epoch with the smallest global validation loss.

4. EXPERIMENTS

4.1. Dataset description and clients configuration setup

We train the SplitFed U-Net model (Fig. 1) on multi-label segmentation of human blastocyst components [17, 18]. Our training dataset includes 815 embryo images, each with a segmentation mask image that includes four components: Zona Pellucida (ZP), Trophectoderm (TE), Inner Cell Mass (ICM), and Blastocoel (BL). A test set of 100 images with uncorrupted GTs is saved for testing the global model. Our system includes 5 clients. Input images are distributed non-uniformly across clients as follows: client 1: 210, client 2: 120, client 3: 85, client 4:180 and client 5: 120, with each client using 85% of it's images for training and 15% for validation.

To corrupt a ground truth mask, mask segments were dilated by a circular structuring element of radius 20 pixels. The effect of this is that boundaries between segments are shifted, emulating inaccurate segment boundary annotation. An example is given in Fig. 2. For each corrupted client, corruption is applied to all its training and validation masks.

4.2. Results and discussion

We compare our QA-SplitFed against five baseline methods: Naive SplitFed ($\overline{W} = \frac{1}{N} \sum_{i=1}^N W_i$), FedAVG [2] ($\overline{W} = \frac{1}{\sum_i m_i} \sum_{i=1}^N m_i W_i$), FedAVG-M [8] (SGD optimizer and momentum of 0.9), SplitAVG [6], and FedMix [7] ($\beta = 1.5$, $\lambda = 10$). All models are trained and validated over $E = 12$ local epochs and $G = 10$ global epochs.

The test set average results are shown in Table 1. In this table, accuracy (Acc.) is defined as the percentage of correctly classified pixels, and loss is the normalized global model loss. The reported values in ZP, TE, ICM and BL are *Jaccard Indices*. Experiments were conducted for different numbers of corrupted clients as shown in first column of this table. Best results are in bold. These results show that the performance of all baseline methods drops dramatically as the number of cor-

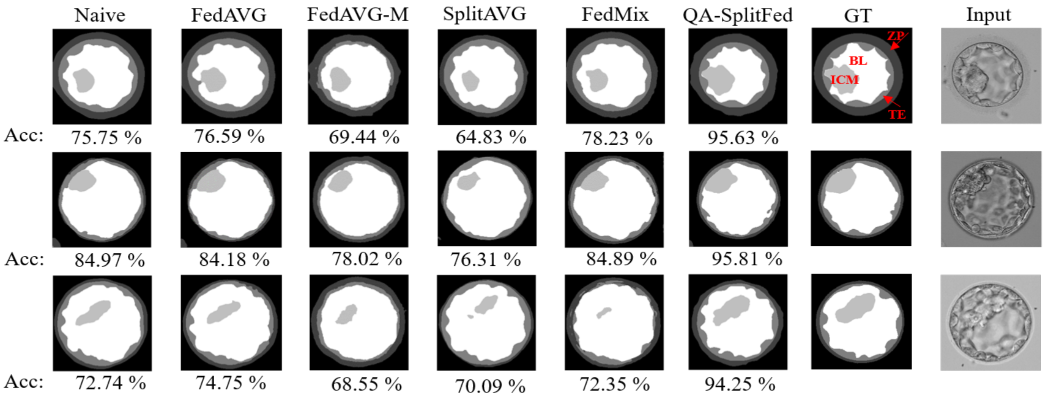

Fig. 3: Visual comparison of the predicted segmentation masks produced by various methods with 3 out of 5 corrupted clients.

Table 1: Comparison of baseline averaging methods with QA-SplitFed on human embryo test set. Acc. is the accuracy; the last four columns show *Jaccard Indices* for ZP, TE, ICM, and BL.

No. of corrupted clients	**Method**	**Loss**	**Acc.** [%]	**ZP** [%]	**TE** [%]	**ICM** [%]	**BL** [%]
0	Naive	0.10	91.87	74.21	73.54	82.35	86.19
	FedAVG	0.11	91.46	72.22	73.18	84.56	87.10
	FedAVG-M	0.15	89.18	69.13	64.16	72.25	80.61
	SplitAVG	0.10	92.25	76.21	72.32	82.10	85.11
	FedMix	0.14	89.00	73.45	74.21	76.12	79.12
	QA-SplitFed	**0.10**	**93.28**	**77.21**	**74.65**	**83.47**	**87.23**
1	Naive	0.13	90.31	68.04	68.62	80.43	85.22
	FedAVG	0.11	91.25	73.21	71.26	81.17	86.09
	FedAVG-M	0.18	87.67	65.22	62.55	62.69	78.37
	SplitAVG	49	66.37	16.25	13.76	58.07	67.01
	FedMix	0.15	88.34	68.03	71.11	75.18	81.15
	QA-SplitFed	**0.10**	**92.52**	**76.43**	**74.66**	**83.02**	**87.17**
2	Naive	0.26	79.79	40.22	68.35	73.11	77.13
	FedAVG	0.24	82.93	45.28	39.21	71.32	78.00
	FedAVG-M	0.25	83.24	59.65	37.87	57.18	71.09
	SplitAVG	0.42	69.52	18.21	17.76	52.02	68.18
	FedMix	0.23	79.24	51.50	63.21	69.36	74.12
	QA-SplitFed	**0.10**	**91.98**	**74.55**	**73.87**	**83.65**	**87.18**
3	Naive	0.34	75.19	29.80	21.25	63.35	71.29
	FedAVG	0.34	75.27	29.02	23.00	63.04	72.16
	FedAVG-M	0.33	77.45	47.26	19.65	58.33	67.72
	SplitAVG	0.40	73.44	21.00	15.19	51.05	68.61
	FedMix	0.28	78.20	43.01	37.05	68.07	71.91
	QA-SplitFed	**0.10**	**92.40**	**75.05**	**71.18**	**81.27**	**86.98**
4	Naive	0.45	65.50	16.76	11.35	53.87	67.46
	FedAVG	0.43	68.79	19.71	13.00	54.14	67.42
	FedAVG-M	0.56	63.95	28.63	14.08	0.00	58.08
	SplitAVG	0.59	62.74	17.09	13.23	0.00	60.87
	FedMix	0.50	65.44	17.85	12.84	49.02	63.00
	QA-SplitFed	**0.10**	**92.00**	**75.08**	**73.09**	**81.25**	**86.34**
5	Naive	0.50	61.50	12.22	7.08	42.21	62.45
	FedAVG	0.50	60.75	12.01	8.12	45.22	63.42
	FedAVG-M	0.64	57.78	15.16	3.08	0.00	54.39
	SplitAVG	**0.48**	**62.35**	**14.14**	**11.81**	**46.95**	**64.10**
	FedMix	0.51	61.15	14.36	7.49	44.01	58.00
	QA-SplitFed	0.51	60.71	13.52	0.00	45.65	61.36

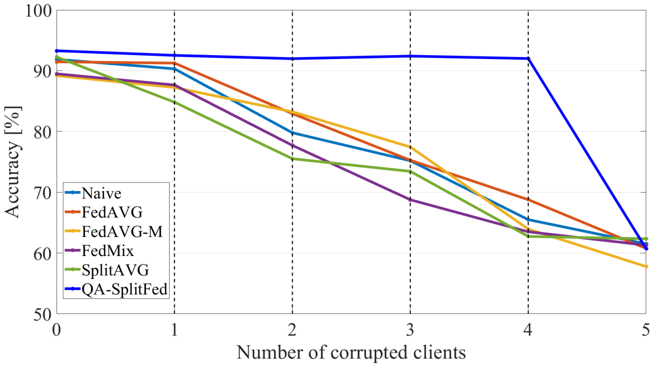

Fig. 4: Global model Accuracy vs no. of corrupted clients.

rupted clients increases. Our proposed method, QA-SplitFed, maintains its accuracy even with 4 out of 5 clients corrupted, so it is able to build an accurate global model with only one accurate client. When all clients are corrupted, all averaging methods lead to a sub-par global models. In this extreme case, SplitAVG had slightly better performance than other methods, but it still produced an inaccurate model. Fig. 4 shows the accuracy of the various methods for different numbers of corrupted clients. Fig. 3 displays three test images with their corresponding correct GTs and their predicted masks with 3 out of 5 corrupted clients. As seen in the figure, the masks predicted by our method are closest to GTs.

5. CONCLUSION

In this paper we proposed a new model averaging method for split-federated learning in the presence of corrupted or poorly annotated clients. We showed that existing averaging methods are fairly fragile in that context. Our proposed method overcomes the challenge of corrupted clients by monitoring the training and validation loss at each client and adjusting the averaging weights adaptively, leading to a more accurate global model even when all but one clients are corrupted. Moreover, QA-SplitFed is more scalable than FL since it can scale to less capable clients (i.e., clients only need to train/run FE and BE, not the full model). Meanwhile, aggregation complexity is similar to that of federated learning.

6. REFERENCES

[1] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *arXiv preprint arXiv:1812.00564*, 2018.

[2] 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*. PMLR, 2017, pp. 1273–1282.

[3] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, vol. 149, pp. 106854, 2020.

[4] Otkrist Gupta and Ramesh Raskar, "Distributed learning of deep neural network over multiple agents," *Journal of Network and Computer Applications*, vol. 116, pp. 1–8, 2018.

[5] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun, "Splitfed: When federated learning meets split learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022, vol. 36, pp. 8485–8493.

[6] Miao Zhang, Liangqiong Qu, Praveer Singh, Jayashree Kalpathy-Cramer, and Daniel L Rubin, "Splitavg: A heterogeneity-aware federated deep learning method for medical imaging," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 9, pp. 4635–4644, 2022.

[7] Jeffry Wicaksana, Zengqiang Yan, Dong Zhang, Xijie Huang, Huimin Wu, Xin Yang, and Kwang-Ting Cheng, "Fedmix: Mixed supervised federated learning for medical image segmentation," *arXiv preprint arXiv:2205.01840*, 2022.

[8] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown, "Measuring the effects of non-identical data distribution for federated visual classification," *arXiv preprint arXiv:1909.06335*, 2019.

[9] Naichen Shi, Fan Lai, Raed Al Kontar, and Mosharaf Chowdhury, "Fed-ensemble: Improving generalization through model ensembling in federated learning," *arXiv preprint arXiv:2107.10663*, 2021.

[10] Xiuwen Fang and Mang Ye, "Robust federated learning with noisy and heterogeneous clients," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10072–10081.

[11] Seunghan Yang, Hyoungseob Park, Junyoung Byun, and Changick Kim, "Robust federated learning with noisy labels," *IEEE Intelligent Systems*, vol. 37, no. 2, pp. 35–43, 2022.

[12] Shenghui Li, Edith Ngai, Fanghua Ye, and Thiemo Voigt, "Auto-weighted robust federated learning with corrupted data sources," *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2022.

[13] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 429–450, 2020.

[14] Maarten G Poirot, Praneeth Vepakomma, Ken Chang, Jayashree Kalpathy-Cramer, Rajiv Gupta, and Ramesh Raskar, "Split learning for collaborative deep learning in healthcare," *arXiv preprint arXiv:1912.12115*, 2019.

[15] Manish Gawali, CS Arvind, Shriya Suryavanshi, Harshit Madaan, Ashrika Gaikwad, KN Bhanu Prakash, Viraj Kulkarni, and Aniruddha Pant, "Comparison of privacy-preserving distributed deep learning methods in healthcare," in *Annual Conference on Medical Image Understanding and Analysis*. Springer, 2021, pp. 457–471.

[16] Xing Chen, Jingtao Li, and Chaitali Chakrabarti, "Energy and loss-aware selective updating for splitfed learning with energy harvesting-powered devices," *Journal of Signal Processing Systems*, pp. 1–15, 2022.

[17] Reza Moradi Rad, Parvaneh Saeedi, Jason Au, and Jon Havelock, "Blast-net: Semantic segmentation of human blastocyst components via cascaded atrous pyramid and dense progressive upsampling," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 1865–1869.

[18] Parvaneh Saeedi, Dianna Yee, Jason Au, and Jon Havelock, "Automatic identification of human blastocyst components via texture," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 12, pp. 2968–2978, 2017.