FEDHCA^2: Towards Hetero-Client Federated Multi-Task Learning

Supplementary Material

A. Proof of Theorem 1

Theorem 1 *Given a multi-task model with a shared encoder and task-specific decoders, and a federated learning system consisting of clients with independent encoders and decoders, gradient descent in the shared encoder of MTL is equivalent to averaging parameter aggregation in FL, adding an extra term $\nabla_{\theta^{(0)}}\langle \hat{g}_i^{(p)}, \hat{g}_j^{(q)}\rangle$ that maximizes the inner product of gradients $\hat{g}_i^{(p)}$ and $\hat{g}_j^{(q)}$ between all pairs of tasks i and j in each iteration p and q.*

Overview The goal is to establish the relationship between the parameter updates in MTL with a shared encoder and FL with independent encoders. We aim to show that the gradient descent updates in MTL is similar to the parameter aggregation in FL, but MTL inherently reduces gradient conflicts among tasks, an effect not directly achieved by FL.

Analysis Consider the following scenario, a multi-task model handles N tasks with a standard multi-decoder architecture consisting of a shared encoder and N task-specific decoders, trained on a multi-task dataset $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{|\mathcal{D}|}$, where $\mathbf{x}_n$ is the input sample and $\mathbf{y}_n = \bigcup_{t=1}^{N} \mathbf{y}_{n,t}$ contains the ground-truth labels for all N tasks. Meanwhile, a Federated Learning system also handles N tasks with N clients, where client C_i addressing the i-th task with independent encoder and decoder. The local dataset for client C_i is separated from the multi-task dataset $\mathcal{D}_i = \{(\mathbf{x}_n, \mathbf{y}_{n,i})\}_{n=1}^{|\mathcal{D}|}$. Assume all models are initialized by $\theta^{(0)}$ and trained for M iterations before aggregation in FL. Here M equals the number of iterations in a single epoch multiplies the number of local epochs set by the FL system.

Firstly, we could define following annotations to facilitate analysis. In the following proof, we mainly focus on the optimization of encoders in MTL and FL, thus abbreviating the parameters of encoder θ^E as θ for simplicity. In MTL, the encoder is always shared by all tasks in each mini-batch, we define the loss function for the i-th task at the m-th iteration and its corresponding gradient as follows:

$$\mathcal{L}_i^{(m)}(\theta^{(m-1)}) = \mathcal{L}_i(\theta^{(m-1)}; \mathcal{B}_i^{(m)}), \tag{1}$$
$$g_i^{(m)} = \nabla_{\theta^{(m-1)}} \mathcal{L}_i^{(m)}, \tag{2}$$

where $\theta^{(m-1)}$ is the encoder parameters in the $(m-1)$-th iteration, $\mathcal{B}_i^{(m)}$ is the mini-batch sampled from $\mathcal{D}$ for task i.

While in FL, each client possesses its specific model, for the same mini-batch $\mathcal{B}_i^{(m)}$ we have the loss function and gradient:

$$\mathcal{L}_i^{(m)}(\theta_i^{(m-1)}) = \mathcal{L}_i(\theta_i^{(m-1)}; \mathcal{B}_i^{(m)}), \tag{3}$$
$$g_i^{(m)} = \nabla_{\theta_i^{(m-1)}} \mathcal{L}_i^{(m)}, \tag{4}$$

where $\theta_i^{(m-1)}$ is the encoder parameters of client C_i in the $(m-1)$-th iteration.

Since the MTL model θ and all FL clients' models θ_i are initialized from the same point $\theta^{(0)}$, we define

$$\hat{g}_i^{(m)} = \nabla_{\theta^{(0)}} \mathcal{L}_i^{(m)}(\theta^{(0)}; \mathcal{B}_i^{(m)}), \tag{5}$$
$$\hat{H}_i^{(m)} = \nabla^2_{\theta^{(0)}} \mathcal{L}_i^{(m)}(\theta^{(0)}; \mathcal{B}_i^{(m)}), \tag{6}$$

to represent the derivative and Hessian Matrix of $\mathcal{L}_i^{(m)}$ of the initial parameters respectively, and are exactly the same for MTL and FL.

Then we delve deeper into the training procedure of MTL. Since the encoder is shared by all tasks, it is updated by the gradient descents computed from the losses of all tasks, which is formulated as:

$$\theta^{(m)} = \theta^{(m-1)} - \eta \sum_{i=1}^{N} g_i^{(m)}, \tag{7}$$

where η denotes the learning rate.

To have an insightful view of gradient descent in MTL, we can perform a Taylor expansion on $g_i^{(m)}$ assuming η is sufficiently small, yielding:

$$\begin{aligned} g_i^{(m)} &= \nabla_{\theta^{(m-1)}} \mathcal{L}_i^{(m)} && (8) \\ &= \nabla_{\theta^{(0)}} \mathcal{L}_i^{(m)} + \nabla^2_{\theta^{(0)}} \mathcal{L}_i^{(m)} (\theta^{(m-1)} - \theta^{(0)}) + O(\eta^2) && (9) \\ &= \hat{g}_i^{(m)} + \hat{H}_i^{(m)} (\theta^{(m-1)} - \theta^{(0)}) + O(\eta^2) && (10) \\ &= \hat{g}_i^{(m)} + \hat{H}_i^{(m)} \sum_{k=1}^{m-1} (\theta^{(k)} - \theta^{(k-1)}) + O(\eta^2) && (11) \\ &= \hat{g}_i^{(m)} - \eta \hat{H}_i^{(m)} \sum_{k=1}^{m-1} \sum_{j=1}^{N} g_j^{(k)} + O(\eta^2). \quad \text{(Use Eq. (7))} && (12) \end{aligned}$$

After M iterations, we can calculate the overall change of encoder parameters by combining Eq. (7) and Eq. (12):

$$\begin{aligned} \Delta\theta_{MTL} &= \theta^{(M)} - \theta^{(0)} && (13) \\ &= -\eta \sum_{m=1}^{M} \sum_{i=1}^{N} g_i^{(m)} && (14) \\ &= -\eta \sum_{m=1}^{M} \sum_{i=1}^{N} \left(\hat{g}_i^{(m)} - \eta \hat{H}_i^{(m)} \sum_{k=1}^{m-1} \sum_{j=1}^{N} g_j^{(k)} \right) + O(\eta^3) && (15) \\ &= -\eta \sum_{m=1}^{M} \sum_{i=1}^{N} \hat{g}_i^{(m)} + \eta^2 \sum_{m=1}^{M} \sum_{i=1}^{N} \left(\hat{H}_i^{(m)} \sum_{k=1}^{m-1} \sum_{j=1}^{N} g_j^{(k)} \right) + O(\eta^3). && (16) \end{aligned}$$

Similarly, we consider the training procedure of client C_i in FL, where the encoder θ_i is updated independently by the gradients computed from the loss of it own task, which is formulated as:

$$\theta_i^{(m)} = \theta_i^{(m-1)} - \eta g_i^{(m)}. \quad (17)$$

Conducting Taylor expansion on $g_i^{(m)}$ yields:

$$\begin{aligned} g_i^{(m)} &= \nabla_{\theta_i^{(m-1)}} \mathcal{L}_i^{(m)} && (18) \\ &= \nabla_{\theta^{(0)}} \mathcal{L}_i^{(m)} + \nabla^2_{\theta^{(0)}} \mathcal{L}_i^{(m)} (\theta_i^{(m-1)} - \theta^{(0)}) + O(\eta^2) && (19) \\ &= \hat{g}_i^{(m)} + \hat{H}_i^{(m)} (\theta_i^{(m-1)} - \theta^{(0)}) + O(\eta^2) && (20) \\ &= \hat{g}_i^{(m)} + \hat{H}_i^{(m)} \sum_{k=1}^{m-1} (\theta_i^{(k)} - \theta_i^{(k-1)}) + O(\eta^2) && (21) \\ &= \hat{g}_i^{(m)} - \eta \hat{H}_i^{(m)} \sum_{k=1}^{m-1} g_i^{(k)} + O(\eta^2). \quad \text{(Use Eq. (17))} && (22) \end{aligned}$$

After M iterations, we can calculate the overall change of the client encoder parameters by combining Eq. (17) and Eq. (22):

$$\begin{aligned} \Delta\theta_i &= \theta_i^{(M)} - \theta^{(0)} && (23) \\ &= -\eta \sum_{m=1}^{M} g_i^{(m)} && (24) \end{aligned}$$

$$= -\eta \sum_{m=1}^{M} \left(\hat{g}_i^{(m)} - \eta \hat{H}_i^{(m)} \sum_{k=1}^{m-1} g_i^{(k)} \right) + O(\eta^3) \tag{25}$$

$$= -\eta \sum_{m=1}^{M} \hat{g}_i^{(m)} + \eta^2 \sum_{m=1}^{M} \left(\hat{H}_i^{(m)} \sum_{k=1}^{m-1} g_i^{(k)} \right) + O(\eta^3). \tag{26}$$

FL server typically aggregates client models by performing a weighted sum of client model parameters, such as FedAvg [14]. The aggregation is formulated as:

$$\tilde{\theta} = \frac{1}{N} \sum_{i=1}^{N} \theta_i, \tag{27}$$

here we simplify it with identical weights for all clients.

Consider its change from the initial weights:

$$\Delta\tilde{\theta}_{FL} = \frac{1}{N} \sum_{i=1}^{N} \Delta\theta_i \tag{28}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \left(-\eta \sum_{m=1}^{M} \hat{g}_i^{(m)} + \eta^2 \sum_{m=1}^{M} \left(\hat{H}_i^{(m)} \sum_{k=1}^{m-1} g_i^{(k)} \right) \right) + O(\eta^3) \quad \text{(Use Eq. (26))} \tag{29}$$

$$= \frac{1}{N} \left(-\eta \sum_{m=1}^{M} \sum_{i=1}^{N} \hat{g}_i^{(m)} + \eta^2 \sum_{m=1}^{M} \sum_{i=1}^{N} \left(\hat{H}_i^{(m)} \sum_{k=1}^{m-1} g_i^{(k)} \right) \right) + O(\eta^3). \tag{30}$$

If we regard the optimizer as capable of automatically scaling the learning rate η, we can neglect the coefficient $1/N$ in Eq. (30). Hence, we could find that the first term $-\eta \sum_{m=1}^{M} \sum_{i=1}^{N} \hat{g}_i^{(m)}$ exists in the parameter update of both MTL (Eq. (16)) and FL (Eq. (30)), showcasing their similarity in optimization of multiple tasks. Furthermore, we can calculate the difference between them:

$$\Delta\theta_{MTL} - \Delta\tilde{\theta}_{FL} \approx \eta^2 \sum_{m=1}^{M} \sum_{i=1}^{N} \left(\hat{H}_i^{(m)} \sum_{k=1}^{m-1} \sum_{j=1}^{N} g_j^{(k)} - \hat{H}_i^{(m)} \sum_{k=1}^{m-1} g_i^{(k)} \right) \tag{31}$$

$$= \eta^2 \sum_{m=1}^{M} \sum_{i=1}^{N} \left(\hat{H}_i^{(m)} \sum_{k=1}^{m-1} \sum_{j=1, j\neq i}^{N} g_j^{(k)} \right) \tag{32}$$

$$= \eta^2 \sum_{m=1}^{M} \sum_{k=1}^{m-1} \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \hat{H}_i^{(m)} g_j^{(k)}. \tag{33}$$

From Eq. (12) we know the approximation that $g_i^{(m)} = \hat{g}_i^{(m)} + O(\eta)$, thus there is

$$\Delta\theta_{MTL} - \Delta\tilde{\theta}_{FL} = \eta^2 \sum_{m=1}^{M} \sum_{k=1}^{m-1} \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \hat{H}_i^{(m)} \hat{g}_j^{(k)}. \tag{34}$$

Since the sequence of mini-batches is randomly shuffled in every local epoch, the ordering of iterations within a local epoch is actually randomized. Consequently, we can compute the expectation of Eq. (34) as follows:

$$\mathbb{E}[\Delta\theta_{MTL} - \Delta\tilde{\theta}_{FL}] = \eta^2 \mathbb{E} \left[\sum_{m=1}^{M} \sum_{k=1}^{m-1} \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \hat{H}_i^{(m)} \hat{g}_j^{(k)} \right] \tag{35}$$

$$= \frac{\eta^2}{2} \sum_{p=1}^{M} \sum_{q=1, q\neq p}^{M} \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} \hat{H}_i^{(p)} \hat{g}_j^{(q)} \tag{36}$$

$$= \frac{\eta^2}{2} \sum_{p=1}^{M} \sum_{q=p+1}^{M} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \left(\hat{H}_i^{(p)} \hat{g}_j^{(q)} + \hat{H}_j^{(p)} \hat{g}_i^{(q)} + \hat{H}_i^{(q)} \hat{g}_j^{(p)} + \hat{H}_j^{(q)} \hat{g}_i^{(p)} \right), \tag{37}$$

where the restriction $k < m$ is relaxed under the expectation, note that the indexes are changed in Eq. (37) for re-organizing the terms in summation. From the definition that $\hat{H}_i = \nabla_{\theta^{(0)}} \hat{g}_i$, we can take a closer look at the inner term:

$$\hat{H}_i^{(p)} \hat{g}_j^{(q)} + \hat{H}_j^{(q)} \hat{g}_i^{(p)} = (\nabla_{\theta^{(0)}} \hat{g}_i^{(p)}) \hat{g}_j^{(q)} + (\nabla_{\theta^{(0)}} \hat{g}_j^{(q)}) \hat{g}_i^{(p)} \tag{38}$$

$$= \nabla_{\theta^{(0)}} \langle \hat{g}_i^{(p)}, \hat{g}_j^{(q)} \rangle, \tag{39}$$

where $\langle \hat{g}_i^{(p)}, \hat{g}_j^{(q)} \rangle$ is the inner product of gradient of task i and j at initial weight, computed with mini-batch $\mathcal{B}_i^{(p)}$ and $\mathcal{B}_j^{(q)}$. Similarly, we have

$$\hat{H}_i^{(q)} \hat{g}_j^{(p)} + \hat{H}_j^{(p)} \hat{g}_i^{(q)} = (\nabla_{\theta^{(0)}} \hat{g}_i^{(q)}) \hat{g}_j^{(p)} + (\nabla_{\theta^{(0)}} \hat{g}_j^{(p)}) \hat{g}_i^{(q)} \tag{40}$$

$$= \nabla_{\theta^{(0)}} \langle \hat{g}_i^{(q)}, \hat{g}_j^{(p)} \rangle. \tag{41}$$

Then we can bring Eq. (39) and Eq. (41) back to Eq. (37):

$$\mathbb{E}[\Delta\theta_{MTL} - \Delta\tilde{\theta}_{FL}] = \frac{\eta^2}{2} \sum_{p=1}^{M} \sum_{q=p+1}^{M} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \left(\nabla_{\theta^{(0)}} \langle \hat{g}_i^{(p)}, \hat{g}_j^{(q)} \rangle + \nabla_{\theta^{(0)}} \langle \hat{g}_i^{(q)}, \hat{g}_j^{(p)} \rangle \right) \tag{42}$$

$$= \frac{\eta^2}{2} \sum_{p=1}^{M} \sum_{q=1, q \neq p}^{M} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \left(\nabla_{\theta^{(0)}} \langle \hat{g}_i^{(p)}, \hat{g}_j^{(q)} \rangle \right). \tag{43}$$

This can be viewed as the summation of gradients of the inner products in Eq. (39) between all $N(N-1)/2$ pairs of tasks and all $M(M-1)$ pairs of mini-batches.

To understand the effect of Eq. (43) in the optimization process of MTL, we can have a quick review at the gradient descent algorithm:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta), \tag{44}$$

which can minimize the loss $\mathcal{L}(\theta)$:

$$\Delta\theta = -\eta \nabla_\theta \mathcal{L}(\theta) \Leftrightarrow \min \mathcal{L}(\theta). \tag{45}$$

Then the term $\nabla_{\theta^{(0)}} \langle \hat{g}_i^{(p)}, \hat{g}_j^{(q)} \rangle$ in parameter update of MTL is equivalent to maximizing the inner product:

$$\nabla_{\theta^{(0)}} \langle \hat{g}_i^{(p)}, \hat{g}_j^{(q)} \rangle \Leftrightarrow \max \langle \hat{g}_i^{(p)}, \hat{g}_j^{(q)} \rangle, \tag{46}$$

since its coefficient $\eta^2/2$ in Eq. (43) is positive.

From existing works in multi-object and multi-task optimization [7, 11, 19, 22], we know that the inner product $\langle \hat{g}_i^{(p)}, \hat{g}_j^{(q)} \rangle$ is a measurement of accordance between gradients $\hat{g}_i^{(p)}$ and $\hat{g}_j^{(q)}$, and maximizing this inner product in parameter update is equal to reducing the conflict of gradients. This complete the proof.

Therefore, the parameter sharing mechanism in MTL can implicitly help mitigate the conflict of gradients across different tasks, yet the parameter aggregation in FL could be limited in this function.

B. Algorithm Pseudo-codes

We provide detailed illustrations of the proposed Hyper Conflict-Averse Aggregation scheme from Section 3.3 and Hyper Cross Attention Aggregation scheme from Section 3.4 in Algorithm 1 and Algorithm 2, respectively.

Algorithm 1 Hyper Conflict-Averse Aggregation

Input: Previous round encoder parameters $\boldsymbol{\theta}^{E,(r-1)} = \{\theta_1^{E,(r-1)}, \dots, \theta_N^{E,(r-1)}\}$, current round encoder updates $\Delta\boldsymbol{\theta}^{E,(r)} = \{\Delta\theta_1^{E,(r)}, \dots, \Delta\theta_N^{E,(r)}\}$, a hyper-parameter $c \in [0, 1)$, Hyper Aggregation Weights for encoders $\boldsymbol{\alpha} = \{\alpha_1, \dots, \alpha_N\}$

Output: Personalized encoder parameters $\boldsymbol{\theta}^{E,(r)}$

1: Define $\Delta\bar{\theta}^E = \frac{1}{N}\sum_{i=1}^{N}\Delta\theta_i^{E,(r)}, \phi = c^2\|\Delta\bar{\theta}^E\|^2$

2: Solve for w^* and λ^*

$$\min_w F(w) = U_w^\top \Delta\bar{\theta}^E + \sqrt{\phi}\|U_w\| \tag{47}$$

$$\text{where } U_w = \frac{1}{N}\sum_{i=1}^{N} w_i \Delta\theta_i^E \tag{48}$$

3: Compute aggregated update

$$\tilde{U} = \Delta\bar{\theta}^E + U_{w^*}/\lambda^* = \Delta\bar{\theta}^E + \frac{\sqrt{\phi}}{\|U_w\|}U_w$$

4: **for** $i \in \{1, \dots, N\}$ **do**

5: Compute personalized update with Hyper Aggregation Weights

$$\theta_i^{E,(r)} = \theta_i^{E,(r-1)} + \Delta\theta_i^{E,(r)} + \alpha_i\tilde{U}$$

6: **end for**

Algorithm 2 Hyper Cross Attention Aggregation

Input: Previous round decoder parameters $\boldsymbol{\theta}^{D,(r-1)} = \{\theta_1^{D,(r-1)}, \dots, \theta_K^{D,(r-1)}\}$, current round decoder updates $\Delta\boldsymbol{\theta}^{D,(r)} = \{\Delta\theta_1^{D,(r)}, \dots, \Delta\theta_K^{D,(r)}\}$, Hyper Aggregation Weights for decoders $\boldsymbol{\beta} = \{\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_K\}$

Output: Personalized decoder parameters $\boldsymbol{\theta}^{(r)}$

1: **for** each layer l in decoder **do**

2: $V_l = [\Delta\theta_{1,l}^{D,(r)}, \dots, \Delta\theta_{K,l}^{D,(r)}]^\top$

3: **for** $i \in \{1, \dots, K\}$ **do**

4: Compute Cross Attention

$$\tilde{A}_{i,l} = \text{Softmax}(\Delta\theta_{i,l}^{D,(r)} \, V_l^\top/\sqrt{d})V_l$$

5: Compute personalized update with Hyper Aggregation Weights

$$\theta_{i,l}^{D,(r)} = \theta_{i,l}^{D,(r-1)} + \Delta\theta_{i,l}^{D,(r)} + \beta_{i,l}\tilde{A}_{i,l}$$

6: **end for**

7: **end for**

C. Additional Implementation Details

Data augmentation. Our methodology follows the established data augmentation procedures in previous studies [4, 12, 21]. To augment the training images, we use random scaling with factors ranging from 0.5 to 2.0, random cropping to the specified input resolutions (which are 512×512 for the PASCAL-Context dataset [15] and 448×576 for the NYUD-v2 dataset [20], in accordance with Swin Transformer's requirements), random horizontal flipping, and random color jittering. Surface normal labels are corrected for horizontal flipping, and depth labels are corrected for scaling. Additionally, we perform image normalization during both the training and evaluation phases.

Loss functions and weights. In alignment with established practices in the field [4, 12, 21], our approach employs distinct loss functions for each specific task. For semantic segmentation and human parts segmentation, we utilize the cross-entropy loss. For saliency detection we use the balanced cross-entropy loss. We adopt the $\mathcal{L}_1$ loss for surface normal estimation and depth estimation. For edge detection, we use the weighted binary cross-entropy loss, assigning a weight of 0.95 to positive pixels and 0.05 to negative ones. For the losses in multi-task clients, we employ a weighted sum of the individual losses to ensure task balancing. The respective loss weights for SemSeg, Parts, Normals, Sal, Edge, and Depth tasks are set at 1, 2, 10, 5, 50, and 1.

Implementation. Our models are optimized using the AdamW optimizer [10] with an initial learning rate of 1e-4 and a weight decay rate of 1e-4. Additionally, we employ a cosine decay learning rate scheduler [9] complemented by a 5-epoch warm-up phase. The Hyper Aggregation Weights are clamped between 0 to 1, and are initially set to 0.01 and updated via the SGD optimizer, which operates at a learning rate of 1e-2, a weight decay rate of 1e-4, and a momentum of 0.9. The process of fine-tuning these hyper-parameters is demonstrated in subsequent sections, specifically in Tab. 4, Tab. 5, and Tab. 6. For the Hyper Conflict-Averse Aggregation hyper-parameter c, we explore a range of values including $\{0.2, 0.4, 0.6, 0.8\}$, with the results of this exploration presented in Tab. 7 and Tab. 8.

To evaluate the performance of our method, we compare with representative works including two traditional FL approaches FedAvg [14] and FedProx [5], four pFL methods FedPer [1], Ditto [6], FedAMP [3], FedBABU [16] and two FMTL methods FedSTA [17] and MaT-FL [2]. These methods are adapted for HC-FMTL setting, with adjustments made to the algorithm implementations to fit this novel setting while striving to retain as much of the original algorithms' integrity as possible. For both FedAvg and FedProx, we apply their aggregation mechanisms to the encoders and decoders separately. The update strategies of FedAMP and Ditto are similarly extended to both encoders and decoders. In the case of FedPer, FedBABU, FedSTA and MaT-FL, ag-

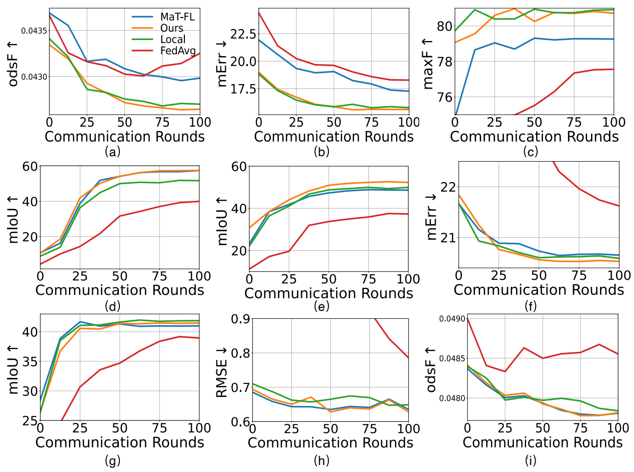

Figure 1. Evaluation results during training, using PASCAL-Context for five single-task clients and NYUD-v2 for one multi-task client. (a) Edge from PASCAL-Context (hereinafter called P) on single-task client. (b) Normals from P on single-task client. (c) Sal from P on single-task client. (d) SemSeg from P on single-task client. (e) Parts from P on single-task client. (f) Normals from NYUD-v2 (hereinafter called N) on multi-task client. (g) SemSeg from N on multi-task client. (h) Depth from N on multi-task client. (i) Edge from N on multi-task client.

gregation is confined to the encoders.

Regarding hyperparameters, we adhere to the default configurations as specified in the original publications or source code of these methods. Specifically, for FedProx, the hyperparameter μ is set to 0.01. For Ditto, the regularization parameter λ is set to 0.1. Within FedAMP, the hyperparameters λ, α_k, and σ are uniformly assigned a value of 1. For MaT-FL, the number of clusters K is set to 2.

Metric evaluation. To evaluate edge detection results, we use the SEISM package [18] and set the maximum allowed mis-localization of the optimal dataset F-measure (odsF) [13] to 0.0075 and 0.011 for PASCAL-Context and NYUD-v2, respectively.

D. Additional Experimental Results

Evaluation results on all tasks. As the supplementary of Fig. 4 in our paper, Fig. 1 shows that FEDHCA^2 converges faster to a better result on most tasks of PASCAL-Context and NYUD-v2, compared to local training baseline, FedAvg and MaT-FL.

Impact of the number of clients. To assess the effectiveness of FEDHCA^2 across varying client counts, we conduct tests by scaling the number of clients per task by factors of 2 and 4, with the datasets evenly split. In our paper, we demonstrate the consistent superior performance and the overall growth trend of our method in Fig. 5. Here, the detailed results of Fig. 5 are illustrated in Tab. 1.

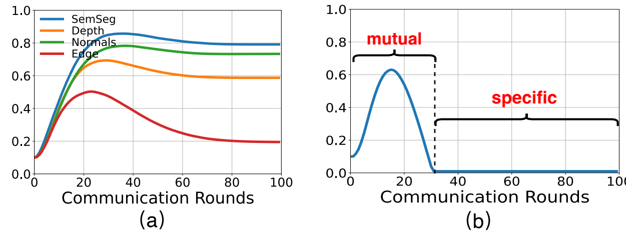

Figure 2. Hyper Aggregation Weights $\boldsymbol{\alpha}$ for encoders of the client models, using NYUD-v2 for four single-task clients and PASCAL-Context for one multi-task client. (a) Weights of four single-task clients. (b) Weights of the multi-task client which differs in two stages.

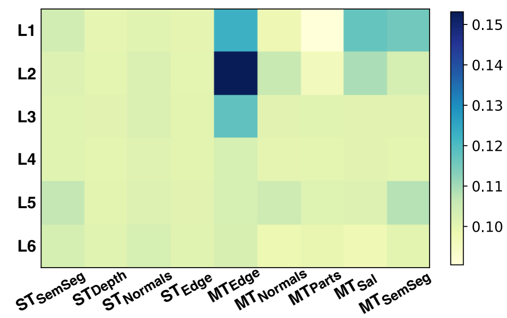

Figure 3. Learned Hyper Aggregation Weights $\boldsymbol{\beta}$ across decoders for different tasks, spanning layers from L1 to L6, using NYUD-v2 for four single-task clients and PASCAL-Context for one multi-task client.

Impact of different FMTL scenarios. To further verify the necessity of introducing our new setting as we do in the paper, we conduct experiments comparing two scenarios: 1) each client handles a single task, and 2) HC-FMTL encompasses both single-task and multi-task clients. These experiments are carried out on the PASCAL-Context, as a supplementary of the NYUD-v2 used and illustrated in Tab. 4 in the paper. As Tab. 2 illustrates, similar results are obtained that while FEDHCA^2 improves upon the local baseline in the single-task client scenario, integrating the multi-task client results in a greater enhancement.

Impact of different backbones. We also conduct a series of experiments using Swin-T, Swin-S, and Swin-B [8] as backbones within the HC-FMTL benchmark setting to evaluate the consistent performance of FEDHCA^2. The results, detailed in Tab. 3, confirm our expectations: as the complexity of the backbone increases from Swin-T to Swin-B, we observe a corresponding improvement in performance across all tasks, further validating the effectiveness of FEDHCA^2.

Interaction between tasks. We investigate the dynamic learning process of Hyper Aggregation Weights for both encoders and decoders, aiming to understand their role in facilitating personalized aggregation for different clients. In Fig. 6 and Fig. 7 in our paper, we show the evolution of weights $\boldsymbol{\alpha}$ for encoders and the last learned weights $\boldsymbol{\beta}$ across decoders on the first benchmark setting described in Sec. 4.1. Here, similar results are obtained on the second benchmark setting as depicted in Fig. 2 and Fig. 3.

Hyper-parameter tuning. Additional experiments are carried out to assess the impact of different initialization values for Hyper Aggregation Weights $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. As depicted in Tab. 4 and Tab. 5, we explore a range from 0.01 to 1, across two HC-FMTL benchmark settings. Optimal results on task metrics within PASCAL-Context are achieved when the initialization value is set to 0.1, which also yield the best average per-task performance drop across both settings. We also investigate the effect of the learning rate for Hyper Aggregation Weights, as illustrated in Tab. 6. When the learning rate varies from 0.005 to 0.1, FedHCA^2 achieves the best overall performance at a learning rate of 0.01. Moreover, as indicated in Tab. 7 and Tab. 8, we experiment with various values of the hyper-parameter c in the Hyper Conflict-Averse Aggregation across the two benchmark settings, scaling from 0.2 to 0.8. In both instances, setting c to 0.4 results in the best results.

E. Qualitative results

To intuitively compare the proposed FedHCA^2 with existing methods, we visualize the task predictions of local training, FedAvg, MaT-FL and our model with an example from the PASCAL-Context dataset in Fig. 4 and an example from the NYUD-v2 dataset in Fig. 5. All methods are trained on the first benchmark scenario. Our model obviously generates better details with less error, especially in semantic segmentation and human parts segmentation in PASCAL-Context and semantic segmentation in NYUD-v2. Our results also more closely resemble the ground truths in other tasks.

Table 1. Comparison to representative methods with the number of clients scaling to 2 and 4 times, using PASCAL-Context for five single-task clients and NYUD-v2 for one multi-task client. 'Δ_m' is calculated w.r.t. corresponding local baseline of 1C, 2C and 4C.

Scale	Method	PASCAL-Context (**ST**)					NYUD-v2 (**MT**)				$\Delta_m\% \uparrow$
		SemSeg mIoU↑	Parts mIoU↑	Sal maxF↑	Normal mErr↓	Edge odsF↑	SemSeg mIoU↑	Depth RMSE↓	Normals mErr↓	Edge odsF↑	
1C	Local	51.69	49.94	**80.91**	15.76	71.95	41.86	0.6487	20.59	76.46	0.00
	FedAvg [14]	39.98	37.33	77.56	18.27	69.17	38.94	0.7858	21.62	75.77	-11.76
	FedProx [5]	44.42	38.10	77.26	18.03	69.39	39.19	0.8068	21.52	76.03	-10.68
	FedPer [1]	54.51	46.56	78.85	16.95	71.00	**44.02**	0.6467	21.19	**76.61**	-1.11
	FedAMP [3]	55.98	52.05	80.79	15.74	72.02	41.67	0.6428	20.54	76.40	1.47
	MaT-FL [2]	57.45	48.63	79.26	17.26	71.23	40.99	0.6352	20.65	76.59	-0.46
	FEDHCA^2	**57.55**	**52.30**	80.71	**15.60**	**72.08**	41.47	**0.6281**	**20.53**	76.50	**2.18**
2C	Local	42.21	47.22	78.64	16.62	70.66	36.77	0.7300	21.96	76.06	0.00
	FedAvg [14]	14.99	29.69	76.68	18.34	69.07	33.01	0.7081	22.15	75.49	-13.95
	FedProx [5]	14.93	30.01	77.03	18.36	68.90	32.47	0.7837	22.14	75.43	-15.20
	FedPer [1]	33.68	42.32	77.72	17.34	70.11	**40.99**	0.6694	21.93	76.02	-1.89
	FedAMP [3]	**45.83**	46.78	78.31	16.62	70.62	36.45	0.7093	22.00	75.53	0.91
	MaT-FL [2]	35.23	43.23	78.14	18.11	70.40	39.91	0.6607	**20.85**	76.16	-1.30
	FEDHCA^2	45.80	**49.52**	**79.73**	**15.98**	**71.61**	39.66	**0.6603**	20.92	**76.21**	**4.70**
4C	Local	21.85	34.84	76.59	17.57	69.17	31.49	0.7433	23.55	74.53	0.00
	FedAvg [14]	9.32	17.22	75.76	18.99	68.42	21.89	0.7725	23.37	74.86	-16.82
	FedProx [5]	7.76	13.93	74.38	20.64	68.04	14.78	0.8147	25.35	**75.87**	-24.04
	FedPer [1]	15.38	31.21	76.00	18.21	**76.00**	34.27	0.6877	23.35	75.20	-2.96
	FedAMP [3]	**25.48**	**38.38**	76.44	17.65	69.22	31.98	0.7908	23.56	74.45	2.36
	MaT-FL [2]	20.10	31.74	76.68	19.06	69.48	33.77	0.7547	22.87	74.74	-1.77
	FEDHCA^2	22.36	36.54	**79.10**	**16.49**	70.72	**36.20**	**0.6510**	**21.30**	75.62	**6.36**

Table 2. Comparison between different settings. 'ST+Local' and 'ST+Ours' denote the setting with five single-task clients on PASCAL-Context, trained with local baseline and FEDHCA^2, respectively. 'ST+MT+Ours' denotes the setting in Tab. 1 trained with our framework. 'Δ_m' is calculated w.r.t. 'ST+Local' baseline.

Method	PASCAL-Context (**ST**)					$\Delta_m\% \uparrow$
	SemSeg mIoU↑	Parts mIoU↑	Sal maxF↑	Normal mErr↓	Edge odsF↑	
ST+Local	51.69	49.94	**80.91**	15.76	71.95	0.00
ST+Ours	**57.76**	51.38	80.75	15.66	**72.08**	3.05
ST+MT+Ours	57.55	**52.30**	80.71	**15.60**	**72.08**	**3.40**

Table 3. Comparison between different backbones on the first benchmark scenario.

Method	PASCAL-Context (**ST**)					NYUD-v2 (**MT**)			
	SemSeg mIoU↑	Parts mIoU↑	Sal maxF↑	Normal mErr↓	Edge odsF↑	SemSeg mIoU↑	Depth RMSE↓	Normals mErr↓	Edge odsF↑
Swin-T	57.55	52.30	80.71	15.60	72.08	41.47	0.6281	20.53	76.50
Swin-S	62.09	56.42	81.35	15.70	73.20	44.80	0.6247	20.44	76.94
Swin-B	65.30	60.73	81.76	15.53	74.65	47.26	0.6055	19.85	77.58

Table 4. Comparison between different initialized values of Hyper Aggregation Weights $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ on the first benchmark scenario.

Init	PASCAL-Context (**ST**)					NYUD-v2 (**MT**)				$\Delta_m\% \uparrow$
	SemSeg mIoU↑	Parts mIoU↑	Sal maxF↑	Normal mErr↓	Edge odsF↑	SemSeg mIoU↑	Depth RMSE↓	Normals mErr↓	Edge odsF↑	
0.01	57.03	51.65	80.69	15.64	72.02	41.21	0.6339	20.62	**76.54**	1.67
0.1	57.55	52.30	80.71	15.60	72.08	**41.47**	**0.6281**	**20.53**	76.50	**2.18**
0.5	57.15	**52.59**	80.66	15.55	**72.14**	40.93	0.6348	20.57	76.45	1.91
1	**58.68**	51.91	**80.79**	**15.52**	72.04	40.96	0.6396	20.61	76.52	2.02

Table 5. Comparison between different initialized values of Hyper Aggregation Weights $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ on the second benchmark scenario.

Init	NYUD-v2 (**ST**)				PASCAL-Context (**MT**)					$\Delta_m\% \uparrow$
	SemSeg mIoU↑	Depth RMSE↓	Normals mErr↓	Edge odsF↑	SemSeg mIoU↑	Parts mIoU↑	Sal maxF↑	Normals mErr↓	Edge odsF↑	
0.01	34.84	0.7126	23.24	74.98	**66.21**	**55.17**	83.29	14.08	71.95	0.62
0.1	**34.95**	**0.7018**	23.19	75.03	65.81	55.01	83.18	14.08	71.97	**0.75**
0.5	34.85	0.7120	**23.15**	**75.12**	65.62	55.02	83.33	14.09	**71.99**	0.57
1	34.74	0.7113	**23.15**	74.99	65.45	55.01	**83.37**	**14.02**	71.95	0.55

Table 6. Comparison between different learning rates of Hyper Aggregation Weights $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ on the first benchmark scenario.

lr	PASCAL-Context (**ST**)					NYUD-v2 (**MT**)				$\Delta_m\% \uparrow$
	SemSeg mIoU↑	Parts mIoU↑	Sal maxF↑	Normal mErr↓	Edge odsF↑	SemSeg mIoU↑	Depth RMSE↓	Normals mErr↓	Edge odsF↑	
0.005	**58.18**	52.06	**80.73**	15.63	72.08	41.03	0.6337	20.61	**76.61**	2.00
0.01	57.55	52.30	80.71	**15.60**	72.08	**41.47**	**0.6281**	**20.53**	76.50	**2.18**
0.05	57.26	**52.46**	80.64	15.64	72.09	41.05	0.6338	20.64	76.48	1.84
0.1	57.90	52.13	80.60	15.62	**72.10**	40.54	0.6332	20.55	76.52	1.85

Table 7. Comparison between different values of hyper-parameter c in Hyper Conflict-Averse Aggregation on the first benchmark scenario.

c	PASCAL-Context (**ST**)					NYUD-v2 (**MT**)				$\Delta_m\% \uparrow$
	SemSeg mIoU↑	Parts mIoU↑	Sal maxF↑	Normal mErr↓	Edge odsF↑	SemSeg mIoU↑	Depth RMSE↓	Normals mErr↓	Edge odsF↑	
0.2	57.12	52.15	80.60	15.63	**72.12**	**41.77**	0.6300	20.63	76.53	2.02
0.4	57.55	**52.30**	**80.71**	**15.60**	72.08	41.47	**0.6281**	**20.53**	76.50	**2.18**
0.6	**57.60**	51.94	80.64	15.62	72.10	40.87	0.6337	20.66	76.46	1.76
0.8	57.18	52.07	80.56	15.65	72.05	40.94	0.6353	20.61	**76.54**	1.69

Table 8. Comparison between different values of hyper-parameter c in Hyper Conflict-Averse Aggregation on the second benchmark scenario.

c	NYUD-v2 (**ST**)				PASCAL-Context (**MT**)					$\Delta_m\% \uparrow$
	SemSeg mIoU↑	Depth RMSE↓	Normals mErr↓	Edge odsF↑	SemSeg mIoU↑	Parts mIoU↑	Sal maxF↑	Normals mErr↓	Edge odsF↑	
0.2	34.47	0.7135	23.20	75.07	65.63	**55.09**	**83.36**	14.08	71.97	0.42
0.4	**34.95**	**0.7018**	**23.19**	75.03	65.81	55.01	83.18	14.08	71.97	**0.75**
0.6	34.48	0.7135	23.22	74.96	**66.02**	55.05	83.35	14.09	**72.01**	0.45
0.8	34.56	0.7105	23.25	**75.17**	65.83	54.89	83.32	**14.06**	71.94	0.48

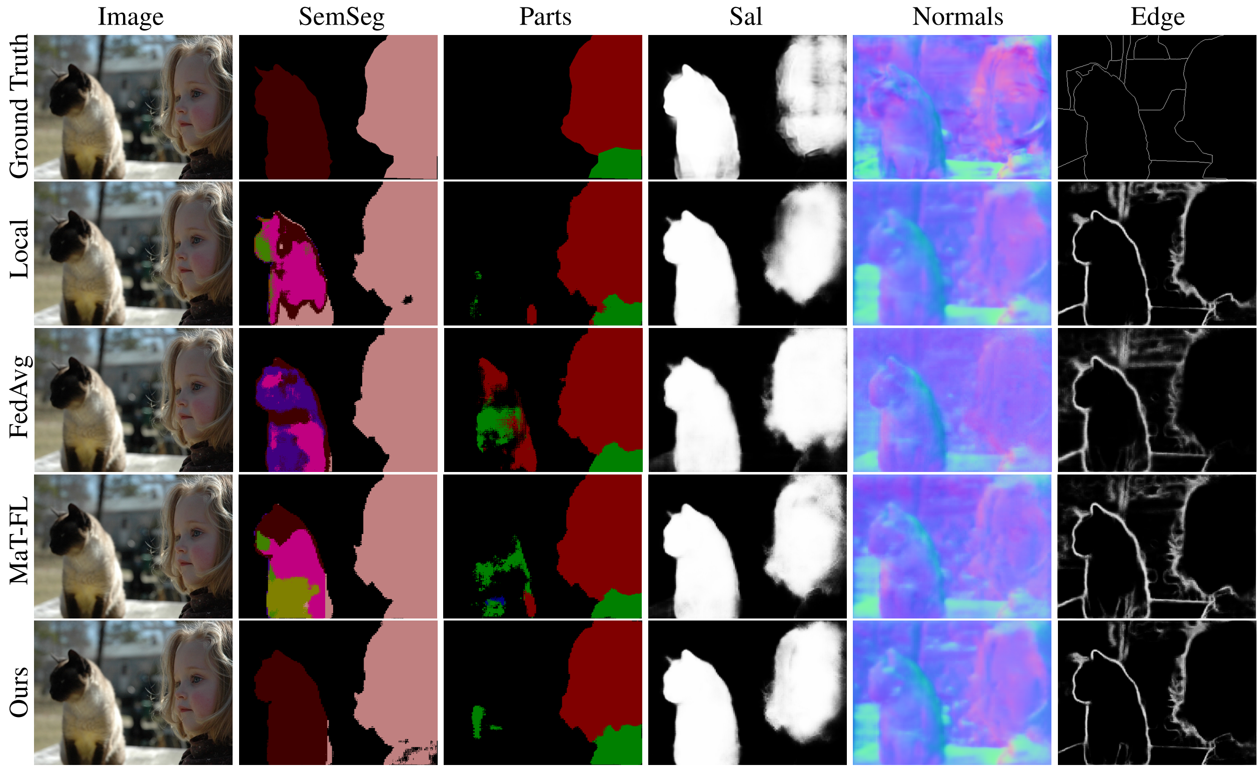

Figure 4. Qualitative results compared with representative methods on PASCAL-Context dataset.

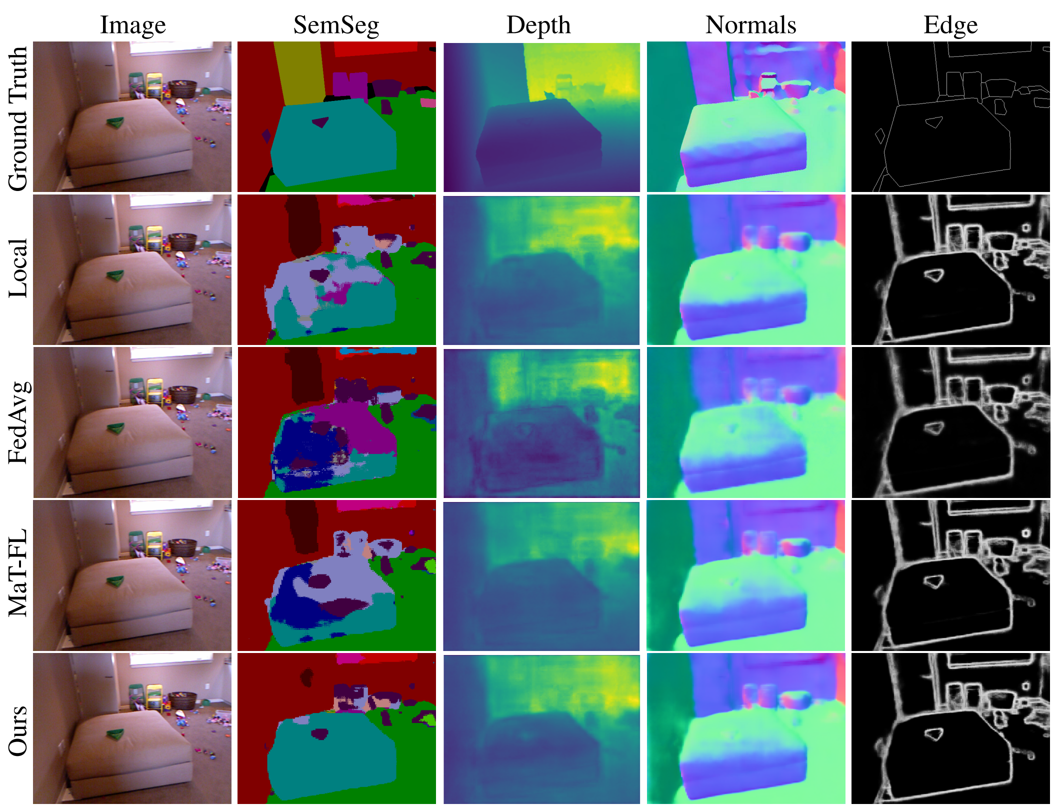

Figure 5. Qualitative results compared with representative methods on NYUD-v2 dataset.

References

[1] Manoj Ghuhan Arivazhagan, Vinay Aggarwal, Aaditya Kumar Singh, and Sunav Choudhary. Federated learning with personalization layers. *arXiv preprint arXiv:1912.00818*, 2019. 5, 8

[2] Ruisi Cai, Xiaohan Chen, Shiwei Liu, Jayanth Srinivasa, Myungjin Lee, Ramana Kompella, and Zhangyang Wang. Many-task federated learning: A new problem setting and a simple baseline. In *CVPR*, pages 5037–5045, 2023. 5, 8

[3] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. Personalized cross-silo federated learning on non-iid data. In *AAAI*, pages 7865–7873, 2021. 5, 8

[4] Menelaos Kanakis, David Bruggemann, Suman Saha, Stamatios Georgoulis, Anton Obukhov, and Luc Van Gool. Reparameterizing convolutions for incremental multi-task learning without task interference. In *ECCV*, pages 689–707, 2020. 5

[5] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In *MLSys*, 2020. 5, 8

[6] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *ICML*, pages 6357–6368, 2021. 5

[7] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *NeurIPS*, 34:18878–18890, 2021. 4

[8] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, pages 10012–10022, 2021. 6

[9] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 5

[10] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 5

[11] Linhao Luo, Yumeng Li, Buyu Gao, Shuai Tang, Sinan Wang, Jiancheng Li, Tanchao Zhu, Jiancai Liu, Zhao Li, and Shirui Pan. MAMDR: A model agnostic learning framework for multi-domain recommendation. In *ICDE*, pages 3079–3092, 2023. 4

[12] Kevis-Kokitsi Maninis, Ilija Radosavovic, and Iasonas Kokkinos. Attentive single-tasking of multiple tasks. In *CVPR*, pages 1851–1860, 2019. 5

[13] David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE TPAMI*, 26(5): 530–549, 2004. 6

[14] Brendan McMahan, Eider Moore, Daniel Ramage, et al. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, pages 1273–1282, 2017. 3, 5, 8

[15] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, pages 891–898, 2014. 5

[16] Jaehoon Oh, Sangmook Kim, and Se-Young Yun. Fedbabu: Toward enhanced representation for federated image classification. In *ECCV*, 2022. 5

[17] Sangjoon Park, Gwanghyun Kim, Jeongsol Kim, Boah Kim, and Jong Chul Ye. Federated split task-agnostic vision transformer for covid-19 cxr diagnosis. *NeurIPS*, 34:24617–24630, 2021. 5

[18] Jordi Pont-Tuset and Ferran Marques. Supervised evaluation of image segmentation and object proposal techniques. *IEEE TPAMI*, 38(7):1465–1478, 2015. 6

[19] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018. 4

[20] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, pages 746–760, 2012. 5

[21] Hanrong Ye and Dan Xu. Inverted pyramid multi-task transformer for dense scene understanding. In *ECCV*, pages 514–530, 2022. 5

[22] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *NeurIPS*, 33:5824–5836, 2020. 4