Who Trains What? Disentangled Global Model Modularization for Federated Learning Guided by Client Metadata

Anonymous authors
Paper under double-blind review

Abstract

Federated Learning (FL) benefits from numerous client participations, yet resource-constrained clients with sufficient data may be excluded due to straggler effects. We study which parts should each client train given its local data and computation budget? This is difficult because the same model structure in a global model produces overlapping responses to different data characteristics, while one data characteristic may involve many structures. We propose MDMP, a metadata-guided method for disentangled global model partitioning. MDMP measures the responses of model structures to different data characteristics. It then groups structures with similar responses into functional modules and selects for each client a set that covers its major data characteristics within its training budget. Experiments on MNIST, CelebA, and CIFAR-10 show that, under a 95% target computation budget, MDMP reduces measured computation by 6.38–9.24% with at most a 0.94 percentage-point drop in mean client accuracy.

1 Introduction

Broad and diverse client participation can improve the generalization of the global model in federated learning (FL) (Collins et al., 2022). In practice, however, clients with valuable data may become stragglers or even be excluded from training because they lack the resources to train the full model. Existing approaches, including model pruning (Jiang et al., 2022; Diao et al., 2021), personalized FL (Fallah et al., 2020; Li et al., 2021a), and split FL (Thapa et al., 2022), have attempted to reduce or adapt client-side workloads, but few explicitly consider local data characteristics when deciding what each client should train. Therefore, assigning a smaller model based only on resource capacity may retain unnecessary parts while removing useful ones. This leads to the central question of this work: ***given both the data characteristics and resource limitations of a client, which parts of the global model should be assigned to it?*** We aim to answer this question by constructing client-side sub-models that match local data characteristics and resource limitation while preserving global model performance. However, the following challenges would be posed to solve this problem.

Challenge 1: Identifying the relationship between data characteristics and model structures. It is non-trivial to determine which parts of a global model are responsible for different local data characteristics. Under data heterogeneity, one data characteristic may be jointly represented by multiple parts of a global model, while the same parts may also contribute to several characteristics. Moreover, conventional parameter-importance or activation-based measurements cannot directly provide a reliable metric. Therefore, it is difficult to build a clear and reliable connection between the characteristics of a client's data and the corresponding parts of the global model;

Challenge 2: Partition the global model without its performance loss. Even if the relationship between data and model structures can be identified, the global model cannot be divided arbitrarily. Different parts of a neural network work together, and changing or removing one part may affect many others. An inappropriate partition may lose useful

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

knowledge, and significantly reduce model accuracy. The challenge is to divide the global model into modules that can be assigned separately while still working well together;

Challenge 3: Balancing client needs with its afford. The model parts that are most useful for a client may require more computation than the client can provide. In contrast, assigning only a very small part of the model may reduce the training cost but prevent the client from learning important patterns in its local data. This problem becomes more difficult because clients differ in both their data and their available resources. Therefore, we need to determine which modules should be assigned to each client so that the client can complete local training while still making a useful contribution to the global model.

To address these challenges, we propose **MDMP**, a metadata-guided framework that partitions a global model into functional modules and assigns suitable modules to each client. The key idea is to use the model's gradient responses to different types of data as a bridge between client data and model structures. To address **Challenge 1**, MDMP measures these responses and makes the relationship between data characteristics and model structures easier to identify. To address **Challenge 2**, it groups model structures with similar responses into balanced modules while preserving the shared components required by different clients. To address **Challenge 3**, it selects for each client a compact set of modules that covers its major data characteristics without exceeding its training budget. In this way, MDMP connects what data a client has, what parts of the model it needs, and what it can afford to train. Our main contributions are summarized as follows:

- We formulate a new data-aware model partition problem for resource-heterogeneous FL. Different from existing approaches that mainly adjust model size according to device capacity, our formulation jointly considers what information a client needs and how much computation it can provide;
- We propose MDMP, a method that connects data characteristics with model structures, and selects proper modules for clients. MDMP provides a new solution to the questions of *what to partition*, *how to partition*, and *who should train each part*;
- We conduct extensive experiments under different datasets, data characteristics, and computation budgets. The results show that MDMP can reduce client-side training computation with only a marginal loss of accuracy, while consistently constructing sub-models that match the data and resource conditions of individual clients.

2 Related Work

2.1 Data Heterogeneities in FL

Client data heterogeneity can arise from differences in label proportions, as well as differences in input features induced by acquisition conditions or visual attributes. FedAvg aggregates parameters after local updates (McMahan et al., 2017). When local objectives differ, FedProx constrains local updates with a proximal term (Li et al., 2020), while SCAFFOLD uses control variates to correct client drift (Karimireddy et al., 2020). To address feature shifts across clients, FedBN keeps batch-normalization parameters local, reducing the mismatch caused by directly aggregating them (Li et al., 2021b). These studies show that both label distributions and input conditions affect federated training, but primarily evaluate optimization and predictive performance.

2.2 Model Partition in FL

Structured pruning removes computational structures at the level of channels or parameter groups, providing a basis for constructing executable smaller models (Wen et al., 2016). In heterogeneous federated learning, HeteroFL assigns subnetworks of different widths according to client compute capacity (Diao et al., 2021), while FedRolex rotates the extracted subnetwork to distribute training more evenly across global parameters (Alam et al., 2022). These methods mainly use resource constraints to determine submodel capacity and extraction positions; they do not use a structural unit's dependence on specific sample conditions as the partitioning criterion.

Other work changes how models are deployed or composed. SplitFed places a split layer between clients and the server and jointly trains the two parts (Thapa et al., 2022). Split-Mix trains base subnetworks that can be combined on demand to meet different model-size and robustness requirements (Hong et al., 2022). Related work also trains and recombines class-specific sparse subnetworks (Caso et al., 2026). We instead start from an already trained federated global model. We identify candidate structures through metadata-conditioned, group-level gradient responses, then perform response calibration, independent measurement, module-removal tests, and resource-constrained client selection. The resulting partition is based on a validated condition–structure relationship, and each client receives an executable subnetwork after structural dependencies are resolved.

3 Problem Definition

Consider a FL system consisting of a central server and K clients. Each client $k \in \mathcal{K} = \{1, 2, \ldots, K\}$ holds a local dataset $\mathcal{D}_k = \{(x_{k,i}, y_{k,i})\}_{i=1}^{N_k}$, where N_k is the number of local training samples at client k. Let $\mathcal{D} = \bigcup_{k\in\mathcal{K}} \mathcal{D}_k$ denote the set of total local training samples in FL system with $N = |\mathcal{D}|$, where $|\boldsymbol{x}|$ represents the cardinality of $\boldsymbol{x}$. In general, the global optimization objective of FL is $\min_{\theta} F(\theta) = \sum_{k\in\mathcal{K}} \frac{N_k}{N} F_k(\theta)$, where $F_k(\theta) = \frac{1}{N_k} \sum_{i=1}^{N_k} \ell(f_\theta(x_{k,i}), y_{k,i})$ denotes the local loss function of client k, $\ell(\cdot, \cdot)$ is the loss function, θ is the model parameters, and $f_\theta(x_{k,i})$ is the model, which has an R layers, indexed by $r \in \mathcal{R} = \{1, 2, \ldots, R\}$. Upon receiving trained local models from clients, the server can perform global model aggregation by FedAvg (McMahan et al., 2017).

Let L be the features of a dataset $\mathcal{D}$, denoted by $\mathcal{F} = \{F^\ell\}_{\ell=1}^{L}$, each of which could be $\mathcal{V}^\ell = \{v_1^\ell, \ldots, v_j^\ell, \ldots, v_{J_\ell}^\ell\}$, where $J_\ell = |\mathcal{V}^\ell|$. For notational convenience, let $\ell \in \mathcal{L} = \{1, 2, \ldots, L\}$, and $j \in \mathcal{J}_\ell = \{1, 2, \ldots, J_\ell\}$ be the indices of feature and entry in $\mathcal{V}^\ell$, respectively. In this paper, the L features $\mathcal{F}$ and their values are referred to as the metadata of dataset $\mathcal{D}$. Please refer to Appendix A to extract metadata from an arbitrary dataset. For example, 40 features can be extracted from the CelebA dataset (Liu et al., 2015), including $F^1 =$ Eyeglasses with $\mathcal{V}^1 = \{\text{present}, \text{absent}\}$, and $F^2 =$ Smiling with $\mathcal{V}^2 = \{\text{present}, \text{absent}\}$. We define a data structure for Metadata, termed general matrix $\mathcal{V}$ with $V = |\mathcal{V}|$, where each row is $\mathcal{V}^\ell$ and each entry in $\mathcal{V}$, i.e., v_j^ℓ, is defined as a conditional feature in this paper. For any sample $(x_{k,i}, y_{k,i})$ from client k, its features can be represented as $\mathcal{Q}_{k,i} = \{\mathcal{Q}_{k,i}^1, \ldots, \mathcal{Q}_{k,i}^\ell, \ldots, \mathcal{Q}_{k,i}^L\}$, where $\mathcal{Q}_{k,i}^\ell \subseteq \mathcal{V}^\ell$ and $\mathcal{Q}_{k,i}^\ell \neq \emptyset$. If $|\mathcal{Q}_{k,i}^\ell| = 1$ holds for all samples of client k, then client k is considered a single-valued factor for ℓ-th feature; otherwise, a multiple-valued one. Let $\mathcal{I}_s$ denote the set of index of single-valued factors across all clients. Define the conditional data subset of client k as $\mathcal{D}_{k,j}^\ell = \{(x_{k,i}, y_{k,i}) \in \mathcal{D}_k \mid v_j^\ell \in \mathcal{Q}_{k,i}^\ell\}$, and let $n_{k,j}^\ell = |\mathcal{D}_{k,j}^\ell|$.

Assume that for model $f_\theta(x_{k,i})$, it can be partitioned into M functional modules based on the clients' metadata, denoted by $\mathcal{P} = \{\mathcal{U}_1, \ldots, \mathcal{U}_M\}$, where $\mathcal{U}_m$ denotes the set of model parameters in functional module m, indexed by $m \in \mathcal{M} = \{1, \ldots, M\}$, and the corresponding model is denoted by $f_{\theta_m}(x)$[1]. It should be noted that both the number of modules M and the set of model parameter $\mathcal{U}_m$ are unknown and need to be determined in this work. Since the number of functional modules affects the aggregation accuracy of the global model, we therefore consider the number of functional modules to satisfy $1 \leq M \leq M_{\max}$, where $M_{\max}$ denotes the maximal number of modules into which the global model can be partitioned. Client k selects at least one module from $\mathcal{P}$. Denote the set of the indices of selected modules by $\mathcal{M}_k \subseteq \mathcal{M}$, and the corresponding sub-model is denoted by $f_{\theta_k}(x)$. The module allocation results of all clients is denoted by $\mathcal{A} = \{\mathcal{M}_1, \ldots, \mathcal{M}_K\}$.

Moreover, based on (Liu et al., 2024), a functional module may correspond to multiple conditional features in practice. Then, the set of conditional features of all functional

[1]It should be noted that the model $f_{\theta_m}(x)$ of functional module m includes the complete input and output modules. For details, please refer to (Hong et al., 2022).

modules is denoted by $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_M\}$, where $\mathcal{S}_m \subseteq \mathcal{V}$ denotes the set of conditional features of module m. It should be noted that the conditional features of different modules may overlap, i.e., $\mathcal{S}_m \cap \mathcal{S}_{m'} \neq \varnothing$, as one conditional feature is often jointly affected by multiple modules. For any conditional feature $v_j^\ell \in \mathcal{V}$, the loss of functional module m on sub-dataset $\mathcal{D}_{k,j}^\ell$ is defined as $\mathcal{L}_{m,j}^\ell = \frac{1}{N_j^\ell} \sum_{k \in \mathcal{K}_j^\ell} \sum_{(x,y) \in \mathcal{D}_{k,j}^\ell} \ell\big(f_{\theta_m}(x), y\big)$. In addition, the average loss on all modules is defined as $\mathcal{L}_{\text{main}} = \frac{1}{M} \sum_{m=1}^{M} \frac{1}{|\mathcal{S}_m|} \sum_{\ell=1}^{L} \sum_{j=1}^{J_\ell} \mathcal{L}_{m,j}^\ell$. To evaluate the performance of the modules selected by each client, we define the training loss of the model $f_{\theta_k}(x)$ on its local dataset $\mathcal{D}_k$ as $\mathcal{L}_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \ell\big(f_{\theta_k}(x_{k,i}), y_{k,i}\big)$.

Objective and Requirements. To enable clients to select suitable functional modules, we define $\xi_k = \frac{\sum_{\ell=1}^{L} \sum_{j=1}^{J_\ell} \sum_{v_j^\ell \in \mathcal{V} \setminus \mathcal{T}_k} p_{k,j}^\ell}{\sum_{\ell=1}^{L} \sum_{j=1}^{J_\ell} \sum_{v_j^\ell \in \mathcal{V}} p_{k,j}^\ell}$ as the proportion of conditional features in the training data $\mathcal{D}_k$ of client k, where $\mathcal{T}_k = \bigcup_{m \in \mathcal{M}_k} \mathcal{S}_m$ denotes the set of conditional features of selected modules by client k. Note that $\xi_k \in [0, 1]$, with smaller values indicating a better match between the conditional features of the selected modules and client's metadata. Meanwhile, to characterize the resource consumption of a client when training sub-model $f_{\theta_k}(x)$, the number of multiply-accumulate operations (MACs) is adopted in this work as the computation metric (Kaplan et al., 2020a). Then, $\tau_k(\mathcal{M}_k, N_k) = 2 \left\lceil \frac{N_k}{B_k} \right\rceil ((1 + \gamma_0) Z_0 + \gamma_1 Z_{\mathcal{M}_k})$ denotes the number of MACs required to train the model with data $\mathcal{D}_k$ once, where B_k denotes the batch size of $\mathcal{D}_k$, $Z_{\mathcal{M}_k}$ is the number of parameters in module $\mathcal{M}_k$, Z_0 denotes the computational workload of one forward pass in a batch, and γ_0 and γ_1 denote the computation coefficients. For different network architectures, the values of Z_0, γ_0, and γ_1 are provided in Appendix B. Therefore, we require the computational workload of client k to satisfy $\tau_k(\mathcal{M}_k, N_k) \leq b_k$, where b_k is the maximum computational limit of client k.

Problem Description. The primary objective of this paper is to partition the global model according to the characteristics of the clients' training data, while considering the training accuracy of the global model and the limited computation capacities of clients. For more intuitive, the joint optimization problem is mathematically formulated as a problem $[\mathcal{P}]$ in Appendix C.1.

4 Our Proposed MDMP

Actually, to solve $[\mathcal{P}]$ is equivalent to figure out the partition of the global model and the allocated sub-models for each client. Hence, our proposed MDMP includes four steps, i.e., disentanglement of conditional features in the global model, construction of functional modules based on structural units, determination of primary conditional features for functional modules, and the allocation of functional modules to clients.

4.1 Disentanglement of Conditional Features

In this work, inspired by (Wen et al., 2016) and (Bau et al., 2017), the structural units are adopted as the minimum granularity for network partitioning. Intuitively, a model's parameters θ can be divided into a divisible part, denoted by θ_{open}, and a fixed part, denoted by θ_{fix}, such that $\theta = [\theta_{\text{open}}^\mathsf{T}, \theta_{\text{fix}}^\mathsf{T}]^\mathsf{T}$, where T denotes the transpose operation. For layer $r \in \mathcal{R}$ in θ_{open}, it can be divided into Q_r exclusively structural units, i.e., $G_{r,e} \cap G_{r,e'} = \varnothing$ $(e \neq e')$, where $G_{r,e}$ denotes the set of parameters of the e-th structural unit in the r-th layer. The rules for partitioning structural units in different model architectures can be referred to (Liu et al., 2017), (Fang et al., 2023). Let the set of all structural units in the model be $\mathcal{G} = \{G_{1,1}, \ldots, G_{1,Q_1}, \ldots, G_{R,1}, \ldots, G_{R,Q_R}\}$, which contains $Q = \sum_{r \in \mathcal{R}} Q_r$ structural units. For notational convenience, the index q is used in this paper to represent one structural unit in $\mathcal{G}$, i.e., $G_q \in \mathcal{G}$. Furthermore, let the parameters in the q-th structural unit be arranged in a vector with dimension d_q, denoted by $\theta_q = \text{vec}(G_q)$. Then, all structural units can be vectorized as $\theta_{\text{open}} = [\theta_1^\mathsf{T}, \theta_2^\mathsf{T}, \ldots, \theta_Q^\mathsf{T}]^\mathsf{T}$. Note that module partitioning applies only to θ_{open}, while θ_{fix} remains fixed in the model θ. However, as shown in Figure 3 in Appendix D, one

structural unit G_q may respond to nearly all conditional features, making functional module partitioning impossible. In this paper, we refer to this phenomenon as feature entanglement.

To achieve feature disentanglement and independently evaluate model partitioning performance, we first divide each client's dataset into three sub-datasets, i.e., $\mathcal{D}_k = \mathcal{D}_k^{\text{cal}} \cup \mathcal{D}_k^{\text{stat}} \cup \mathcal{D}_k^{\text{sel}}$, where $\mathcal{D}_k^{\text{cal}}$ is used to train the global model parameters and mitigate feature entanglement, $\mathcal{D}_k^{\text{stat}}$ is used to construct functional modules, and $\mathcal{D}_k^{\text{sel}}$ is used to determine the primary conditional features of a module and allocate modules for each client. Then, the conditional loss of client k on conditional feature v_j^ℓ in dataset $\mathcal{D}_k^{\text{cal}}$ and the weighted loss across clients are defined respectively as $F_{k,j}^{\ell,\text{cal}}(\theta) = \frac{1}{n_{k,j}^{\ell,\text{cal}}} \sum_{(x,y)\in\mathcal{D}_{k,j}^{\ell,\text{cal}}} \ell(f_\theta(x), y)$ and $F_j^{\ell,\text{cal}}(\theta) = \sum_{k\in\mathcal{K}_j^{\ell,\text{cal}}} \omega_{k,j}^\ell F_{k,j}^{\ell,\text{cal}}(\theta)$, where $n_{k,j}^{\ell,\text{cal}}$ and $\mathcal{K}_j^{\ell,\text{cal}}$ denote the number of clients with conditional feature v_j^ℓ in dataset $\mathcal{D}_k^{\text{cal}}$ and the set of clients with such feature v_j^ℓ, respectively. The weight is defined as $\omega_{k,j}^\ell = \dfrac{n_{k,j}^{\ell,\text{cal}}}{\sum\limits_{k\in\mathcal{K}_j^{\ell,\text{cal}}} n_{k,j}^{\ell,\text{cal}}}$. We define the gradient of structural unit G_q with respect to conditional feature v_j^ℓ (also called the conditional gradient) and the normalized conditional gradient, respectively, as

$$g_{q,j}^{\ell,\text{cal}} = \nabla_{\theta_q} F_j^{\ell,\text{cal}}(\theta), \qquad R_{q,j}^{\ell,\text{cal}} = \sqrt{\frac{\left\| g_{q,j}^{\ell,\text{cal}} \right\|_2^2 + \varepsilon_r^2}{d_q}}, \tag{1}$$

where $\varepsilon_r > 0$ is a constant that ensures the gradient is derivable everywhere (Sun et al., 2010). Our key idea is to highlight the response of conditional gradient of a structural unit to particular conditional features. Therefore, we first apply the normalized $R_{q,j}^{\ell,\text{cal}}$ to obtain $\phi_{q,j}^{\ell,\text{cal}} = \dfrac{R_{q,j}^{\ell,\text{cal}} + \varepsilon_\phi}{\sum\limits_{u\in\mathcal{J}_\ell} \left(R_{q,u}^{\ell,\text{cal}} + \varepsilon_\phi\right)}$, where constant $\varepsilon_\phi > 0$ improves the stability in the normalization process. With $\phi_{q,j}^{\ell,\text{cal}}$, the concentration loss is built to alleviate the entropy of a structural unit's responses to the multiple values of a single factor, thereby narrowing down its responses on a small number of conditional features. Accordingly, the concentration loss is defined as

$$\mathcal{L}_{\text{con}}^{\text{cal}} = -\frac{1}{Q|\mathcal{I}_s|} \sum_{q=1}^{Q} \sum_{\ell\in\mathcal{I}_s} \frac{\sum\limits_{j\in\mathcal{J}_\ell} \phi_{q,j}^{\ell,\text{cal}} \log \phi_{q,j}^{\ell,\text{cal}}}{\log |\mathcal{J}_\ell|}. \tag{2}$$

Note that (2) forces structural units to response more centrally on a conditional feature of the single-valued factor, while still be capable to response on different conditional features of a multi-valued factor. Let $\mathcal{E}_\ell = \{(j, j') : j < j',\ j, j' \in \mathcal{J}_\ell\}$ denote the set of pairs of different conditional features of single-valued factor $\ell \in \mathcal{I}_s$. To suppress the response of one conditional features of the single-valued factor affecting another single-valued one, and allow necessary response sharing under a threshold among different single-valued factors, we define the following constrained separation loss.

$$\mathcal{L}_{\text{sep}}^{\text{cal}} = \frac{1}{|\mathcal{I}_s|} \sum_{\ell\in\mathcal{I}_s} \frac{1}{|\mathcal{E}_\ell|} \sum_{(j,j')\in\mathcal{E}_\ell} \left[\frac{\left(r_j^{\ell,\text{cal}}\right)^{\mathsf{T}} r_{j'}^{\ell,\text{cal}}}{\left\| r_j^{\ell,\text{cal}} \right\|_2 \left\| r_{j'}^{\ell,\text{cal}} \right\|_2} - \tau_{\text{sep}} \right]_+^2, \tag{3}$$

where $[u]_+ = \max\{u, 0\}$, $0 < \tau_{\text{sep}} < 1$ is the response similarity threshold, and the vector $r_j^{\ell,\text{cal}} = \left[R_{1,j}^{\ell,\text{cal}}, \ldots, R_{Q,j}^{\ell,\text{cal}}\right]^{\mathsf{T}}$ is obtained on conditional feature v_j^ℓ over dataset $\bigcup_{k\in\mathcal{K}} \mathcal{D}_k^{\text{cal}}$. It is worth noting that considering only the concentration and separation loss functions may sacrifice model accuracy. Therefore, we further consider the following model loss functions. In order to ensure that model parameters do not deviate too far from a converged global model, the following model training loss function is considered

$\mathcal{L}_{\text{task}}^{\text{cal}} = \frac{1}{L}\sum_{\ell\in\mathcal{L}}\frac{1}{J_\ell}\sum_{j\in\mathcal{J}_\ell} F_j^{\ell,\text{cal}}(\theta) + \lambda_{\text{prox}}\left\|\theta_{\text{open}} - \theta_{\text{open}}^{(0)}\right\|_2^2$, where $\lambda_{\text{prox}} > 0$ denotes the weight, and $\theta_{\text{open}}^{(0)}$ is from a well-trained global model of $\theta^{(0)}$.

Furthermore, to prevent a large number of structural units from favoring only a small number of conditional features, this paper defines the following response coverage loss function $\mathcal{L}_{\text{cov}}^{\text{cal}} = \frac{1}{L}\sum_{\ell\in\mathcal{L}}\frac{1}{J_\ell}\sum_{j\in\mathcal{J}_\ell}\left[\rho_j^\ell - \bar{\phi}_j^\ell\right]_+^2$, where $\bar{\phi}_j^\ell = \frac{1}{Q}\sum_{q=1}^{Q}\phi_{q,j}^{\ell,\text{cal}}$ denotes the average normalized response of conditional feature v_j^ℓ across all structural units, and $0 < \rho_j^\ell \leq 1$ defines the minimum response coverage ratio. Based on the descriptions of the above loss functions, the overall loss function for achieving feature disentanglement can be written as $\mathcal{L}_{\text{cal}} = \mathcal{L}_{\text{task}}^{\text{cal}} + \mathcal{L}_{\text{str}}^{\text{cal}}$, where $\mathcal{L}_{\text{str}}^{\text{cal}} = \lambda_{\text{con}}\mathcal{L}_{\text{con}}^{\text{cal}} + \lambda_{\text{sep}}\mathcal{L}_{\text{sep}}^{\text{cal}} + \lambda_{\text{cov}}\mathcal{L}_{\text{cov}}^{\text{cal}}$, where λ_{con}, λ_{sep}, and λ_{cov} denote the weights. In general, the server itself cannot directly calculate the gradient of the loss function $\mathcal{L}_{\text{cal}}$, because $\mathcal{L}_{\text{cal}}$ contains client gradient information, namely $g_{q,j}^{\ell,(t)}$. To address this issue, this paper develops a more efficient two-stage method for updating the global model parameters, and details can be found in Appendix E. Let θ^* denote the updated global model parameters.

4.2 Construction of Functional Modules

Based on θ^*, different structural units should be integrated into multiple functional modules. To this end, this paper uses conditional gradients as a criterion to characterize model sensitivity, and gathering different structural units into a module based on the similarity of their conditional gradients. The reason for choosing conditional gradients as the criterion is provided in Appendix F. Similar to (1), the conditional gradient of g_q with respect to conditional feature v_j^ℓ can be written as $g_{q,j}^{\ell,\text{stat}} = \sum_{k\in\mathcal{K}_j^{\ell,\text{stat}}}\frac{n_{k,j}^{\ell,\text{stat}}}{n_j^{\ell,\text{stat}}}\nabla_{\theta_q}F_{k,j}^{\ell,\text{stat}}(\theta)\Big|_{\theta=\theta^*}$. To mitigate the influence caused by different parameter sizes of structural units, the normalized conditional gradient is defined as $R_{q,j}^{\ell,\text{stat}} = \frac{\left\|g_{q,j}^{\ell,\text{stat}}\right\|_2}{\sqrt{d_q}}$. According to Appendix F, it can be seen that the normed conditional gradient characterizes the sensitivity of the corresponding loss to structural unit G_q. Once the conditional gradient of $F_{k,j}^{\ell,\text{stat}}(\theta)$ is obtained locally, the server aggregates $g_{q,j}^{\ell,\text{stat}}$ to further obtain $R_{q,j}^{\ell,\text{stat}}$. Then, the corresponding response matrix is defined as

$$\mathbf{R}_G = \left[R_{q,j}^{\ell,\text{stat}}\right]_{Q\times V}. \tag{4}$$

The q-th row of $\mathbf{R}_G$ represents the response of structural unit G_q to conditional feature v_j^ℓ. Therefore, each row of $\mathbf{R}_G$ represents the sensitivity of a structural unit to different conditional features. To aggregate multiple structural units into functional modules, we define the response magnitude of structural unit G_q on conditional feature v_j^ℓ and the maximum response magnitude, respectively, as $a_{q,j}^\ell = \|\mathbf{R}_G[q,\ell,j]\|_2$ and $a_{\max}^\ell = \max_{q,\ell,j} a_{q,j}^\ell$. Moreover, given a threshold $\tau_{\text{resp}} > 0$, the index sets of shared structural units and feature-sensitive structural units are respectively defined as $Q_{\text{sh}} = \{q : a_{q,j}^\ell \leq \tau_{\text{resp}} a_{\max} \text{ for any } v_j^\ell\}$ and $Q_{\text{act}} = \{1,\ldots,Q\}\setminus Q_{\text{sh}}$. The structural units in Q_{act} need to be aggregated into functional modules, whereas the structural units in Q_{sh} do not participate and are shared by all functional modules. Note that no functional modules are constructed when $a_{\max} = 0$ or $Q_{\text{act}} = \varnothing$. Assume that the feature-sensitive structural units can form M functional modules, and let $\mathcal{I}_m$ denote the set of structural units' indices contained in the m-th functional module. The creation of functional modules must consider 1) no structural units are shared by other functional modules; 2) The number of parameters and the MACs cost of each functional module should not be excessively large. Therefore, determining the M functional modules can be equivalently formulated as solving the optimization problem $[\mathcal{P}_1]$, which is formulated in Appendix C.2. $[\mathcal{P}_1]$ can be approximately solved with both spherical k-means (Schubert et al., 2021) and a capacity-constrained assignment framework (Lähderanta et al., 2024). Detailed solutions are provided in Appendix G. Let the set $\hat{\mathcal{I}}_m$ be the index of structural units allocated in the functional modules m, and the corresponding sets of

structural units is $\hat{\mathcal{U}}_m = \bigcup_{q\in\hat{\mathcal{I}}_m} g_q$. The obtained set of M functional modules is denoted by $\hat{\mathcal{U}} = \{\hat{\mathcal{U}}_1, \hat{\mathcal{U}}_2, \ldots, \hat{\mathcal{U}}_M\}$.

4.3 Main Conditional Features in Functional Modules

The response matrix $\mathbf{R}_G$ only measures the relationship between structural units and conditional features, but does not capture the relationship between functional modules $\mathcal{U}_m$ and conditional features. Therefore, we construct a gradient-response matrix of the functional modules to evaluate their relationships with the various conditional features. Based on $\boldsymbol{r}_q$ in Appendix C.2, we define $e^{\ell}_{m,j} = \frac{\frac{1}{|\mathcal{I}_m|}\sum_{q\in\mathcal{I}_m} r^{\ell}_{q,j}}{\max\left\{\frac{1}{Q}\sum_{q=1}^{Q} r^{\ell}_{q,j}, \varepsilon_R\right\}}$ as the ratio of the averaged response of the structural units within a module m over the averaged response of Q structural units, where $r^{\ell}_{q,j}$ denotes the value of r_q for conditional feature v^{ℓ}_j, and $\varepsilon_R > 0$ is a numerical stability constant introduced to avoid division by zero. Note that $e^{\ell}_{m,j}$ actually represents the response of functional module m to conditional feature v^{ℓ}_j. A larger $e^{\ell}_{m,j}$ means more sensitive of conditional feature v^{ℓ}_j in functional module m. Let $\mathrm{Top}_{s_{\max}}$ denote the selection of at most $s_{\max}$ conditional features according to the values of $e^{\ell}_{m,j}$ in descending order. The set of primary conditional features of module m is then defined as $S_m = \mathrm{Top}_{s_{\max}}\left(e^{\ell}_{m,j},\, \ell \in \mathcal{C},\, j \in \mathcal{J}_\ell\right)$. To verify the effectiveness of the above method, we evaluate its effect on prediction results through module removal analysis. The specific procedure is provided in Appendix H.

4.4 The selection of Functional Modules for Clients

Considering the heterogeneities of both clients' data and computational resources in FL, different clients need to select different functional modules to construct their own sub-models. Let $A_m = \sum_{q\in\hat{\mathcal{I}}_m} d_q$ denote the number of parameters in functional module m, and let the binary variable $x_m \in \{0, 1\}$ indicate whether module m is selected. Since conditional features occupy different proportions at each client, we define $\mathcal{Q}_{\tau_k}$ as the set of conditional features whose proportion in the dataset is no less than τ_k, i.e., $\mathcal{Q}_{\tau_k} = \left\{v^{\ell}_j : p^{\ell}_{k,j} \geq \tau_k, \ell \in \mathcal{L},\ j \in \mathcal{J}_\ell\right\}$.We hope that the functional modules selected by each client can cover all conditional features in its set $\mathcal{Q}_{\tau_k}$. Therefore, the considered problem is a typical set cover issue, which is formulated as $[\mathcal{P}_2]$ in Appendix C.3. For each client k, $[\mathcal{P}_2]$ can be solved using conventional optimization method (Hua et al., 2010). Let x^*_m denote the solution, and the index set of the corresponding functional modules selected by client k is then $\mathcal{M}^*_k$. Combining $\mathcal{M}^*_k$ with the shared structural units $\mathcal{Q}_{\mathrm{sh}}$, and the required input/output structural components yields a sub-model f_{θ_k} for client k.

5 Experimental Evaluations

Datasets and protocols. We use ResNet-18 on MNIST, CelebA, and CIFAR-10 with ten non-IID clients and 28, 80, and 26 metadata conditions, respectively. At 95% and 90% target MAC budgets, we evaluate mean accuracy, measured MACs, and the largest client accuracy drop. The test set is used only after selection; full settings appear in Appendix Table 3. Controlled pruning baselines (L1, Taylor, HRank) share the trained model and client partitions with MDMP. End-to-end baselines (HeteroFL, FedRolex, FjORD without distillation) train heterogeneous subnetworks directly.

Evaluation criteria. We measure MACs on each executable subnetwork rather than inferring compute from its retained parameter count. A target budget specifies an upper limit, so the measured MAC ratio may be lower. For each client, accuracy drop is measured against the same client's full-model result; the maximum drop tests the pre-specified 5-percentage-point limit. Mean accuracy and worst-client drop therefore answer different questions. The main comparisons use one locked seed. Independent CIFAR-10 repeats assess the stability of MDMP, while their detailed outcomes and the complete training configuration remain in the appendix.

Table 1: Controlled module selection at the 95% target MAC budget. Acc. is mean client test accuracy (%); MACs is measured computation relative to the full model (%). Results use one locked seed.

	MNIST		CelebA		CIFAR-10	
Method	Acc. ↑	MACs ↓	Acc. ↑	MACs ↓	Acc. ↑	MACs ↓
L1	**99.68**	93.62	92.70	90.75	**88.04**	92.52
Taylor	99.64	93.61	**92.82**	90.75	87.72	92.52
HRank	97.46	93.61	92.46	90.75	86.84	92.52
Full model	99.70	100.00	92.82	100.00	88.51	100.00
MDMP	**98.78**	93.62	**92.82**	90.76	**87.81**	92.53

Red: MDMP accuracy; **blue**: highest competing accuracy at the same target budget.

Controlled module selection. At nearly matched measured MACs (Table 1), MDMP ties the best mean accuracy on CelebA but trails L1 on MNIST and CIFAR-10. It saves 6.38–9.24% MACs relative to the full model. We therefore assess client-level losses separately.

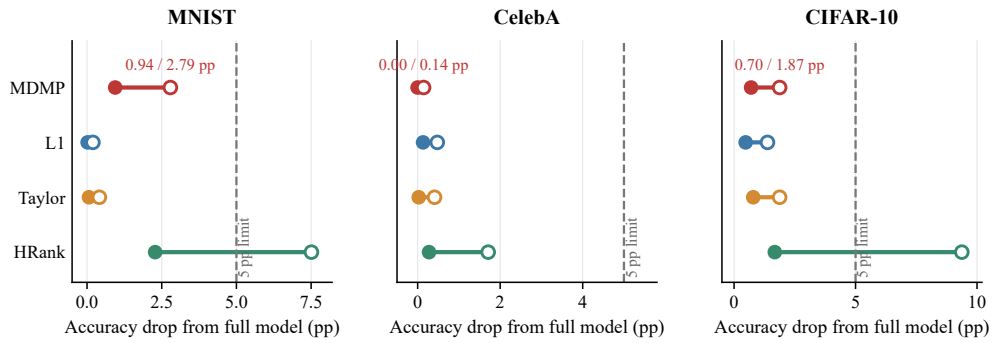


Figure 1: Client accuracy drops at the 95% target MAC budget. Filled and open markers denote mean and maximum drops, respectively; the dashed line marks the 5-percentage-point limit. MDMP meets this limit on all three datasets.

Client-level performance. At the 95% target, MDMP's maximum client drops are 2.79, 0.14, and 1.87 percentage points on the three datasets; all 30 clients meet the 5-point limit (Figure 1). HRank breaches the limit for some MNIST and CIFAR-10 clients, which mean accuracy obscures.

Across-seed stability. Across three CIFAR-10 training and client-partition seeds, MDMP obtains $87.70 \pm 0.93\%$ and $87.50 \pm 0.51\%$ mean accuracy at the two targets; all 60 seed–client evaluations meet the 5-point limit. Appendix Table 4 and Figure 5 report measured MACs and seed-level results. Baselines lack matching repeats, so these results establish repeatability, not cross-method statistical superiority.

End-to-end comparison. The second protocol compares final client accuracy with methods that train heterogeneous subnetworks directly. Its target percentages are shared, but the actual MACs are not: the complete measured values appear in Appendix Table 5. This comparison evaluates the resulting training pipelines and is separate from the controlled pruning comparison above.

At the 95% target, the three end-to-end baselines use 89.34%, 89.57%, and 92.02% of full-model MACs on MNIST, CelebA, and CIFAR-10, respectively, whereas MDMP uses 93.62%, 90.76%, and 92.53%. Consequently, a higher accuracy at the same target does not imply a higher accuracy at equal measured computation. We report the accuracy results for completeness, but interpret them alongside the measured resources rather than merging them with the pruning comparison.

Table 2: Mean client test accuracy (%) in the end-to-end comparison. Columns denote target MAC budgets; results use one locked seed.

	MNIST		CelebA		CIFAR-10	
Method	95%	90%	95%	90%	95%	90%
HeteroFL	**99.70**	99.68	92.35	**92.35**	**88.60**	**88.94**
FedRolex	99.64	99.58	92.10	92.29	87.83	86.67
FjORD (no KD)	99.62	**99.68**	**92.38**	92.32	84.66	84.30
Full model	99.70	99.70	92.82	92.82	88.51	88.51
MDMP	**98.78**	**97.24**	**92.82**	**92.74**	**87.81**	**87.28**

Red: MDMP; **blue**: highest-accuracy end-to-end baseline at each target budget.

At the 95% target, MDMP records 92.82% on CelebA versus 92.38% for the strongest baseline, but uses more measured MACs. HeteroFL is higher on MNIST and CIFAR-10 while using fewer MACs (Table 2). Thus, the clearest evidence for MDMP is its controlled client-level performance at the 95% target, not uniform end-to-end dominance. At the 90% target, one MNIST client under MDMP exceeds the 5-point drop limit (Appendix Table 6).

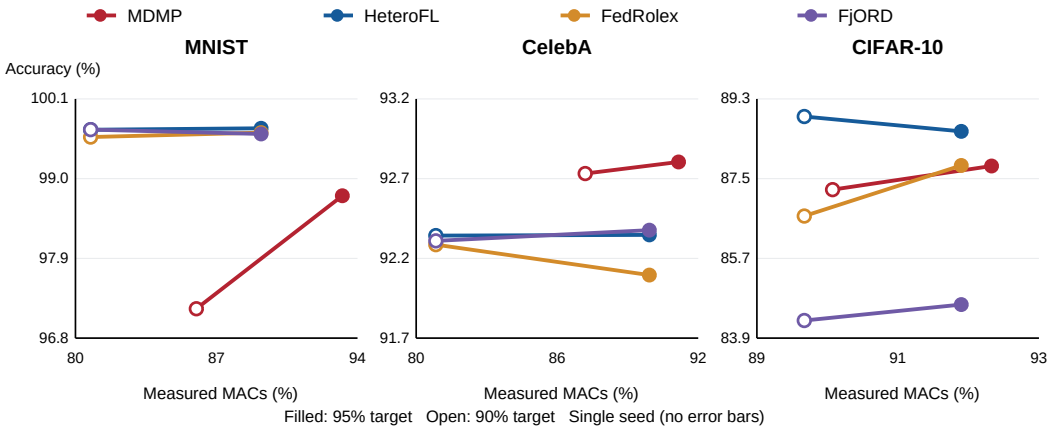


Figure 2: Mean client accuracy versus measured MACs on three datasets. Filled and open markers denote the 95% and 90% target budgets; red denotes MDMP. Vertical scales differ across panels.

6 Conclusion

This work studied who should train what in resource-heterogeneous FL. We propose MDMP, a metadata-guided framework that modularizes a trained global model and assigns each client modules matching its data characteristics and computation limitation. MDMP connects data characteristics with model structures through conditional-gradient responses, and groups structural units into functional modules. Experiment results show that MDMP reduces computation by 6.38–9.24% with at most a 0.94 percentage-point drop in mean client accuracy.

7 AI use statement

In this work, we used generative AI tools to assist with paper polish and code writing. We did not use generative AI tools for paper content generation. Additionally, we have reviewed all AI-assisted work. For example, LLM-generated code was verified and tested for correctness by at least 2 authors. We take responsibility for the final content of this work, including text, claims, or content produced with the aid of generative AI.

8 Ethics statement

This work adheres to the ICLR Code of Ethics. Our research does not involve human subjects, animal experiments, or the collection of sensitive personal data. The datasets used are publicly available benchmarks. We have considered the potential societal impacts and do not foresee risks of misuse, privacy violations, or discrimination.

9 Reproducibility statement

All datasets used in this work are publicly available. The datasets, model hyperparameters, computational resources, and other experimental settings are described in the Experimental Setup section. We believe that these materials are sufficient to reproduce the main findings of this paper.

References

Samiul Alam, Luyang Liu, Ming Yan, and Mi Zhang. FedRolex: Model-heterogeneous federated learning with rolling sub-model extraction. In *Proceedings of NeurIPS*, 2022.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of CVPR*, 2017.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html`.

Francesco Caso, Samuele Fonio, Simone Monaco, Nicola Saccomanno, and Fabrizio Silvestri. Composable sparse subnetworks via maximum-entropy principle. In *Proceedings of ICLR*, 2026.

Daniel C. Castro, Jeremy Tan, Bernhard Kainz, Ender Konukoglu, and Ben Glocker. Morpho-mnist: Quantitative assessment and diagnostics for representation learning. *Journal of Machine Learning Research*, 2019.

Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Fedavg with fine tuning: Local updates lead to representation learning. In *Advances in Neural Information Processing Systems*, 2022.

Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. In *International Conference on Learning Representations*, 2021.

Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016. doi: 10.48550/arXiv.1603.07285. URL `https://arxiv.org/abs/1603.07285`.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. In *Advances in Neural Information Processing Systems*, 2020.

Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. Depgraph: Towards any structural pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16091–16101, 2023.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL `https://www.deeplearningbook.org/`. See Chapter 9, "Convolutional Networks".

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022. doi: 10.48550/arXiv.2203.15556. URL `https://arxiv.org/abs/2203.15556`. Published at NeurIPS 2022 as "An empirical analysis of compute-optimal large language model training".

Junyuan Hong, Haotao Wang, Zhangyang Wang, and Jiayu Zhou. Efficient split-mix federated learning for on-demand and in-situ customization. In *International Conference on Learning Representations*, 2022.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Qiang-Sheng Hua, Yuexuan Wang, Dongxiao Yu, and Francis C. M. Lau. Dynamic programming based algorithms for set multicover and multiset multicover problems. *Theoretical Computer Science*, 411(26–28):2467–2474, 2010. doi: 10.1016/j.tcs.2010.02.016.

Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K. Leung, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Jared Kaplan, Sam McCandlish, Thomas Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020a. URL `https://api.semanticscholar.org/CorpusID:210861095`.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020b. doi: 10.48550/arXiv.2001.08361. URL `https://arxiv.org/abs/2001.08361`.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proceedings of ICML*, 2020.

Tero Lähderanta, Lauri Lovén, Leena Ruha, Teemu Leppänen, Ilkka Launonen, Jukka Riekki, and Mikko J. Sillanpää. Capacitated spatial clustering with multiple constraints and attributes. *Engineering Applications of Artificial Intelligence*, 2024.

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

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *Proceedings of the 38th International Conference on Machine Learning*, 2021a.

Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. FedBN: Federated learning on non-iid features via local batch normalization. In *Proceedings of ICLR*, 2021b.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2736–2744, 2017.

Ziming Liu, Eric Gan, and Max Tegmark. Seeing is believing: Brain-inspired modular training for mechanistic interpretability. *Entropy*, 2024.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of ICCV*, 2015.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of AISTATS*, 2017.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–15. ACM, 2021. doi: 10.1145/3458817.3476209. URL https://doi.org/10.1145/3458817.3476209.

A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, et al. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

Erich Schubert, Andreas Lang, and Gloria Feher. Accelerating spherical k-means. In *Proceedings of SISAP*, 2021.

Deqing Sun, Stefan Roth, and Michael J. Black. Secrets of optical flow estimation and their principles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Proceedings of NeurIPS*, 2016.

An Yan, Yu Wang, Yiwu Zhong, Chengyu Dong, Zexue He, Yujie Lu, William Yang Wang, Jingbo Shang, and Julian McAuley. Learning concise and descriptive attributes for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Appendix

CONTENTS

A metadata extraction **13**

A.1 CelebA 13

A.2 MNIST 14

A.3 CIFAR-10 14

B Computation Workload for Different Network Architecture **15**
B.1 Theoretical Computation Cost During Training 15
B.2 Computing F_{fwd} from the Network Architecture 16

C Mathematical Problem Formulations **17**
C.1 Problem Formulation for $[\mathcal{P}]$ 17
C.2 Problem Formulation for $[\mathcal{P}_1]$ 17
C.3 Problem Formulation for $[\mathcal{P}_2]$ 18

D feature entanglement **18**

E Two-Stage Gradient Update Procedure **20**

F Conditional Gradient for Local Sensitivity Analyses **21**

G Cosine-Centroid Clustering under Dual Capacity Constraints **21**

H Module Removal Test on the Selected Set **21**

I Supplementary Experimental Results **22**
I.1 Fluctuation of Multi-Seed Results 22
I.2 Client-Level Performance Retention 24
I.3 Module Granularity and Multi-Seed Stability 24
I.4 Resource Auditing and Multi-Seed Results 25
I.5 Condition Coverage and Module–Condition Response 26
I.6 Module Composition and Selection–Test Consistency 27
I.7 Training and Calibration Trajectories 28
I.8 Same-MAC Random Control 29

A METADATA EXTRACTION

A.1 CELEBA

For CelebA, we use the facial attribute annotations officially provided with the dataset as sample-level metadata. CelebA provides 40 binary attributes for each facial image, covering facial appearance, hair characteristics, expressions, accessories, and other information. This paper directly read these annotations from the official attribute file, `list_attr_celeba.txt`, and align them with individual images according to their filenames, without training an additional attribute recognition model.

We use *Smiling* as the binary classification label and the remaining 39 attributes as auxiliary metadata. We place *Smiling* as the first feature while preserving the relative order of the other attributes in the official CelebA annotation file. Thus, each sample contains $L = 40$ metadata features, denoted by $\mathcal{F} = \{F_l^e\}_{l=1}^{40}$.

where F^1 = Smiling; F^2 = Eyeglasses; F^3 = 5 o'Clock Shadow; F^4 = Arched Eyebrows; F^5 = Attractive; F^6 = Bags Under Eyes; F^7 = Bald; F^8 = Bangs; F^9 = Big Lips; F^{10} = Big Nose; F^{11} = Black Hair; F^{12} = Blond Hair; F^{13} = Blurry; F^{14} = Brown Hair; F^{15} = Bushy Eyebrows; F^{16} = Chubby; F^{17} = Double Chin; F^{18} = Goatee; F^{19} = Gray Hair;

F^{20} = Heavy Makeup; F^{21} = High Cheekbones; F^{22} = Male; F^{23} = Mouth Slightly Open; F^{24} = Mustache; F^{25} = Narrow Eyes; F^{26} = No Beard; F^{27} = Oval Face; F^{28} = Pale Skin; F^{29} = Pointy Nose; F^{30} = Receding Hairline; F^{31} = Rosy Cheeks; F^{32} = Sideburns; F^{33} = Straight Hair; F^{34} = Wavy Hair; F^{35} = Wearing Earrings; F^{36} = Wearing Hat; F^{37} = Wearing Lipstick; F^{38} = Wearing Necklace; F^{39} = Wearing Necktie; F^{40} = Young。

All CelebA attributes are binary. Therefore, for any $\ell \in \{1, \ldots, 40\}$, the value space is uniformly defined as $\mathcal{Y}^{\ell} = \{\text{present}, \text{absent}\}$.

where *present* indicates that the corresponding attribute is present, whereas *absent* indicates that it is not. The values 1 and -1 in the official CelebA annotations are converted to *present* and *absent*, respectively. For the task feature F^1, *present* indicates smiling and *absent* indicates not smiling. For F^2–F^{40}, *present* and *absent* indicate the presence and absence of the corresponding auxiliary attributes, respectively.

Thus, the 40 binary metadata features in CelebA jointly define $\sum_{\ell=1}^{40} |\mathcal{Y}^{\ell}| = 40 \times 2 = 80$ conditions. Each image takes exactly one value for each metadata feature and therefore corresponds to 40 individual conditions: one task-label condition and 39 auxiliary-metadata conditions.

A.2 MNIST

For MNIST, we use both the digit labels and the morphological attributes of each sample as metadata. The digit labels are provided by the original MNIST dataset, while the remaining attributes are obtained from Morpho-MNIST (Castro et al., 2019), which provides sample-aligned morphological measurements for the original MNIST images. We directly retrieve these measurements according to the sample indices.

Each MNIST sample contains $L = 7$ metadata features, denoted by $\mathcal{F} = \{F^{\ell}\}_{\ell=1}^{7}$, where F^1 = Digit, $\mathcal{V}^1 = \{0, 1, \ldots, 9\}$; F^2 = Area, $\mathcal{V}^2 = \{\text{small}, \text{medium}, \text{large}\}$; F^3 = Length, $\mathcal{V}^3 = \{\text{short}, \text{medium}, \text{long}\}$; F^4 = Thickness, $\mathcal{V}^4 = \{\text{thin}, \text{medium}, \text{thick}\}$; F^5 = Slant, $\mathcal{V}^5 = \{\text{low}, \text{medium}, \text{high}\}$; F^6 = Width, $\mathcal{V}^6 = \{\text{narrow}, \text{medium}, \text{wide}\}$; and F^7 = Height, $\mathcal{V}^7 = \{\text{short}, \text{medium}, \text{tall}\}$.

where F^1 is the task label, while F^2–F^7 are continuous morphological measurements provided by Morpho-MNIST. For each morphological attribute, we divide its empirical distribution on the training set into three value intervals—low, medium, and high—using the 1/3 and 2/3 quantiles. Note that the three states of Slant indicate only the relative magnitudes of the slant measurements and are not further interpreted as left-slanted, upright, or right-slanted.

Therefore, the MNIST metadata contains 10 digit-category states and 6×3 morphological-attribute states, yielding a total of $\sum_{\ell=1}^{7} |\mathcal{V}^{\ell}| = 10 + 6 \times 3 = 28$ conditions. Each sample corresponds to one digit-category condition and six morphological-attribute conditions.

A.3 CIFAR-10

For CIFAR-10, only image class labels are provided, with no directly usable attribute annotations. To obtain sample-level auxiliary metadata, this paper uses a frozen CLIP ViT-B/32 model to construct visual attributes from the semantic similarity between images and attribute descriptions. The attribute vocabulary is adopted from the CIFAR-10 attribute examples provided in LM4CV by (Yan et al., 2023). This paper uses only the attribute descriptions and doesn't reproduce the attribute search or classification procedure of LM4CV.

Specifically, we use the OpenAI-pretrained CLIP ViT-B/32 model as a fixed image–text encoder (Radford et al., 2021). For each attribute F^{ℓ}, its name is inserted into the unified text template "a photo of $\{F^{\ell}\}$" to obtain the corresponding attribute description. We then use the CLIP image encoder and text encoder to obtain the image and attribute-text representations, respectively, and apply L_2 normalization to both representations.

For image x_i and attribute F^ℓ, the attribute score is defined as the cosine similarity between the normalized image and text representations:

$$s_i^\ell = \frac{\langle g_{\text{img}}(x_i), g_{\text{text}}(p^\ell) \rangle}{\|g_{\text{img}}(x_i)\|_2 \, \|g_{\text{text}}(p^\ell)\|_2}. \tag{4}$$

Here, p^ℓ denotes the text prompt corresponding to attribute F^ℓ, while g_{img} and g_{text} denote the frozen CLIP image and text encoders, respectively. The CLIP parameters remain fixed throughout the attribute-extraction process.

Together with the CIFAR-10 task label, we construct $L = 9$ metadata features for each image, denoted by $\mathcal{F} = \{F^\ell\}_{\ell=1}^{9}$, where F^1 = Class Label; F^2 = Antlers in Males; F^3 = A Pointed Bow and Stern; F^4 = Propellers or Jet Engines; F^5 = Moist Slimy Skin; F^6 = A Long Head with a Mane and Tail; F^7 = Landing Gear; F^8 = Portholes along the Hull; and F^9 = Four Wheels.

F^1 is the task label, and its value space is

$$\mathcal{V}^1 = \{\text{airplane}, \text{automobile}, \text{bird}, \text{cat}, \text{deer}, \text{dog}, \text{frog}, \text{horse}, \text{ship}, \text{truck}\}.$$

F^2–F^9 are eight auxiliary visual attributes obtained using CLIP. Because CLIP produces continuous image–text similarity scores rather than discrete manually annotated attributes, we determine discrete thresholds using only the attribute scores on the CIFAR-10 training set. For each attribute F^ℓ, its threshold is defined as the median similarity score over all training samples:

$$\tau^\ell = \text{Median}\left(\{s_i^\ell : x_i \in \mathcal{D}_{\text{train}}\}\right), \qquad \ell = 2, \ldots, 9.$$

Accordingly, the value space of each of the eight visual attributes is uniformly defined as

$$\mathcal{V}^\ell = \{\text{low}, \text{high}\}, \qquad \ell = 2, \ldots, 9.$$

For any image x_i, its attribute value is determined as follows:

$$F^\ell(x_i) = \begin{cases} \text{high}, & s_i^\ell \geq \tau^\ell, \\ \text{low}, & s_i^\ell < \tau^\ell. \end{cases}$$

The thresholds determined from the training set remain fixed for all subsequent data. Here, high and low indicate that an image has relatively high or low CLIP semantic similarity to the corresponding attribute description, respectively, and should not be directly interpreted as manually verified "attribute present" or "attribute absent" labels. Therefore, these CIFAR-10 attributes are auxiliary metadata generated by a frozen vision–language model rather than human-annotated attributes officially provided with the dataset.

Finally, one ten-valued task feature and eight binary auxiliary features jointly yield $\sum_{\ell=1}^{9} |\mathcal{V}^\ell| = 10 + 8 \times 2 = 26$ conditions for CIFAR-10. Each image simultaneously corresponds to one class condition and eight visual-attribute conditions, activating nine conditions in total.

B Computation Workload for Different Network Architecture

B.1 Theoretical Computation Cost During Training

Let N denote the number of training samples, B the batch size, E the number of training epochs, F_{fwd} the forward-pass computation cost per batch, F_{bwd} the backward-pass computation cost per batch, and F_{opt} the optimizer-update computation cost per batch. The number of training steps is

$$S = E \left\lceil \frac{N}{B} \right\rceil.$$

Therefore, the total training computation cost can be written as

$$F_{\text{train}} = \sum_{e=1}^{E} \sum_{s=1}^{\lceil N/B \rceil} \left(F_{\text{fwd}}^{(e,s)} + F_{\text{bwd}}^{(e,s)} + F_{\text{opt}}^{(e,s)} \right).$$

If all batches have the same input size, this can be approximated as

$$F_{\text{train}} \approx E \left\lceil \frac{N}{B} \right\rceil \left(F_{\text{fwd}} + F_{\text{bwd}} + F_{\text{opt}} \right).$$

For commonly used dense neural networks, the backward-pass computation cost is typically about 1.5–2 times that of the forward pass. Therefore, the following approximation is commonly used:

$$F_{\text{train}} \approx E \left\lceil \frac{N}{B} \right\rceil \left((1+\gamma) F_{\text{fwd}} + F_{\text{opt}} \right),$$

where γ denotes the ratio of the backward-pass computation cost to the forward-pass computation cost. For many standard neural networks, $\gamma \approx 2$ can be used, so the combined computation cost of the forward and backward passes during training is approximately three times the forward-pass cost.

It should be noted that F_{opt} cannot always be neglected. The optimizer overhead of SGD is typically small; Momentum requires additional vector operations; and Adam maintains first- and second-order moments, resulting in a noticeably larger optimizer overhead. If the model contains P parameters, the computation cost of an Adam optimizer update can generally be approximated as being proportional to P:

$$F_{\text{opt}} \approx c_{\text{Adam}} P,$$

where c_{Adam} depends on the specific implementation and whether operations such as weight decay are included.

B.2 Computing F_{fwd} from the Network Architecture

The total forward-pass computation cost can be obtained by summing the costs of all layers:

$$F_{\text{fwd}} = \sum_{l=1}^{L} F_l.$$

For a fully connected layer with input dimension d_{in}, output dimension d_{out}, and batch size B, the computation cost is

$$F_{\text{linear}} \approx 2 B d_{\text{in}} d_{\text{out}}.$$

Here, one multiplication and one addition are counted as two FLOPs. When using MACs, the corresponding cost is

$$\text{MACs}_{\text{linear}} = B d_{\text{in}} d_{\text{out}}.$$

Therefore, the following relation is commonly used:

$$1 \text{ MAC} \approx 2 \text{ FLOPs}.$$

For a two-dimensional convolutional layer, the computation cost is

$$F_{\text{conv}} \approx 2 B H_{\text{out}} W_{\text{out}} C_{\text{out}} \left(\frac{C_{\text{in}}}{g} \right) K_h K_w,$$

where H_{out} and W_{out} are the output feature-map dimensions; C_{in} and C_{out} are the numbers of input and output channels; K_h and K_w are the kernel dimensions; and g is the number of groups.

For a Transformer self-attention layer with sequence length L and hidden dimension d, ignoring relatively small terms such as softmax and normalization, the forward-pass computation cost can be approximated as

$$F_{\text{attn}} \approx 8BLd^2 + 4BL^2d.$$

where $8BLd^2$ mainly arises from the Q, K, and V projections and the output projection, while $4BL^2d$ mainly arises from computing the attention matrix and the weighted sum. Therefore, when the sequence length is large, the L^2d term becomes the dominant source of computation.Kaplan et al. (Kaplan et al., 2020b) relate training compute to model size, dataset size, and the number of optimization steps, and estimate the computation based on the parameter count, batch size, and number of training steps. Narayanan et al. (Narayanan et al., 2021) explicitly use the number of training tokens, batch size, number of training iterations, and actual throughput to estimate end-to-end training time. Brown et al. (Brown et al., 2020) provide a method for estimating the training FLOPs of large-scale language models in the appendix. Hoffmann et al. (Hoffmann et al., 2022), commonly referred to as the Chinchilla paper, also employ a computational model that relates training FLOPs to model size and the number of training tokens. The derivation of convolutional computation can be found in (Dumoulin & Visin, 2016; Goodfellow et al., 2016; Vaswani et al., 2017).

C Mathematical Problem Formulations

C.1 Problem Formulation for $[\mathcal{P}]$

In this paper, to achieve our goal, we minimize a weighted objective consisting of the local task loss $\mathcal{L}_k$, the conditional-feature requirement matching measure ξ_k, and the resource cost τ_k, while satisfying the performance requirements. We jointly determine the number and partitioning of functional modules $(M, \mathcal{P})$, the set of primary conditional features $\mathcal{S}$ for each module, and the client module-selection set $\mathcal{A}$. Therefore, the joint optimization problem can be mathematically formulated as follows.

$$\begin{aligned}
[\mathcal{P}]: \quad \min_{M,\mathcal{P},\mathcal{S},\mathcal{A}} \quad & \gamma\mathcal{L}_{\text{main}} + \sum_{k=1}^{K} w_k \left[\mathcal{L}_k + \lambda\xi_k + \beta\tau_k\right] \\
\text{s.t.} \quad & 1 \le M \le M_{\max}, \quad 0 < \tau_k(\mathcal{M}_k, N_k) \le b_k, \\
& \varnothing \neq \mathcal{M}_k \subseteq \mathcal{M}, \quad \mathcal{S}_m \subseteq \mathcal{V}, \quad \mathcal{P} = \{\mathcal{U}_1, \ldots, \mathcal{U}_M\},
\end{aligned}$$

where $\gamma, \lambda, \beta > 0$ are the weights of $\mathcal{L}_{\text{main}}$, ξ_k, and τ_k, respectively, and $w_k = \frac{N_k}{N}$. The server determines the functional module partition and the primary conditional features, while each client uses its local training data to carry out the training of the functional sub-modules based on the decision results. It should be noted that solving problem $[\mathcal{P}]$ is highly challenging because: 1) multiple discrete decision variables are coupled with each other, resulting in a vast combinatorial search space; and 2) task performance depends on the specific combination of modules, making it difficult to obtain an optimal solution in analytical form.

C.2 Problem Formulation for $[\mathcal{P}_1]$

In Subsection 4.2, the partitioning of functional modules must satisfy the following requirements:

$$\bigcup_{m=1}^{M} \mathcal{I}_m = \mathcal{Q}_{\text{act}}, \qquad \mathcal{I}_m \cap \mathcal{I}_{m'} = \varnothing \quad (m \neq m'), \qquad \mathcal{I}_m \neq \varnothing, \tag{5}$$

$$\sum_{q\in\mathcal{I}_m} \nu_q \le \frac{\alpha}{M} \sum_{q\in\mathcal{Q}_{\text{act}}} \nu_q, \qquad \sum_{q\in\mathcal{I}_m} \mu_q \le \frac{\alpha}{M} \sum_{q\in\mathcal{Q}_{\text{act}}} \mu_q, \qquad m = 1, \ldots, M. \tag{6}$$

where ν_q and μ_q denote the proportions of the parameter count and multiply–accumulate operations of structural unit G_q, respectively, relative to those of all structural units. Given an input size, both quantities can be calculated from the parameter size and computational

structure of each structural unit and then normalized; the corresponding calculation method follows Howard et al. (Howard et al., 2017). The capacity relaxation factor $\alpha \geq 1$ limits the parameter count and computational cost of each module, preventing resource overhead from being excessively concentrated in a small number of modules. Therefore, the optimization problem can be formulated as

$$[\mathcal{P}_1]: \max_{\{\mathcal{I}_m\}_{m=1}^{M}} \frac{1}{Q_{\text{act}}} \sum_{m=1}^{M} \sum_{q \in \mathcal{I}_m} \boldsymbol{r}_q^{\top} \overline{\boldsymbol{r}}_m$$
$$\text{s.t. } (5), (6),$$

where

$$\overline{\boldsymbol{r}}_m = \frac{\sum_{q \in \mathcal{I}_m} \boldsymbol{r}_q}{\left\| \sum_{q \in \mathcal{I}_m} \boldsymbol{r}_q \right\|_2}, \qquad \boldsymbol{r}_q = \frac{\mathbf{R}_{\text{G}}[q,:]}{\|\mathbf{R}_{\text{G}}[q,:]\|_2}.$$

C.3 Problem Formulation for $[\mathcal{P}_2]$

$$[\mathcal{P}_2]: \quad \min_{x_m} \sum_{m \in \mathcal{M}} A_m x_m$$
$$\text{s.t.} \quad \sum_{\substack{m \in \mathcal{M}: \\ v_j^{\ell} \in \mathcal{S}_m}} x_m \geq 1, \quad \forall v_j^{\ell} \in \mathcal{Q}_{\tau_k},$$
$$r_k(x_m) \leq b_k, \quad x_m \in \{0,1\},$$

where $r_k(x_m) = 2\lceil \frac{N_k}{B_k} \rceil ((1+\gamma_0) \sum_{m \in \mathcal{M}} Z_m x_m + \gamma_1 \sum_{m \in \mathcal{M}} A_m x_m)$, Z_m denotes the computation cost of one forward pass through functional module m.

D Feature Entanglement

For a structural unit q with a nonzero response, given a relative threshold $\eta \in (0,1]$, its set of significantly responsive conditions is defined as

$$\mathcal{H}_q(\eta) = \left\{ v_j^{\ell} \in \mathcal{V} : R_{q,j}^{\ell,\text{stat}} \geq \eta \max_{\ell', j'} R_{q,j'}^{\ell',\text{stat}} \right\}.$$

If $|\mathcal{H}_q(\eta)|/|\mathcal{V}|$ is close to 1, the structural unit exhibits strong local responses to most conditions and is difficult to assign to a module specific to only a few conditions based on its original responses. This ratio serves as an operational definition for identifying broadly responsive units; broad responsiveness should not be directly interpreted as erroneous or useless sharing. The within-cluster, constrained-separation, and coverage terms in the main text adjust the row-wise distributions, process only inter-column similarities exceeding the threshold, and preserve condition coverage, respectively. The calibrated responses must be recomputed on an independent statistical dataset.

Matched Responses Before and After Calibration. Figure 3 uses the seed-0 runs for the three datasets. The original FedAvg checkpoint is paired with a predefined calibration checkpoint, and the structural-unit response matrices are computed under the same client partition using the independent statistical dataset $\mathcal{D}_k^{\text{stat}}$. The matrix dimensions for MNIST, CelebA, and CIFAR-10 are 4810×28, 4802×80, and 4810×26, respectively, and the selected numbers of calibration steps are 5, 3, and 3. Within each pair, the structural-unit indices and condition order are identical. The figure displays only row-normalized responses obtained by dividing each row by its maximum value; therefore, the color intensities cannot be used to compare the absolute response magnitudes before and after calibration.

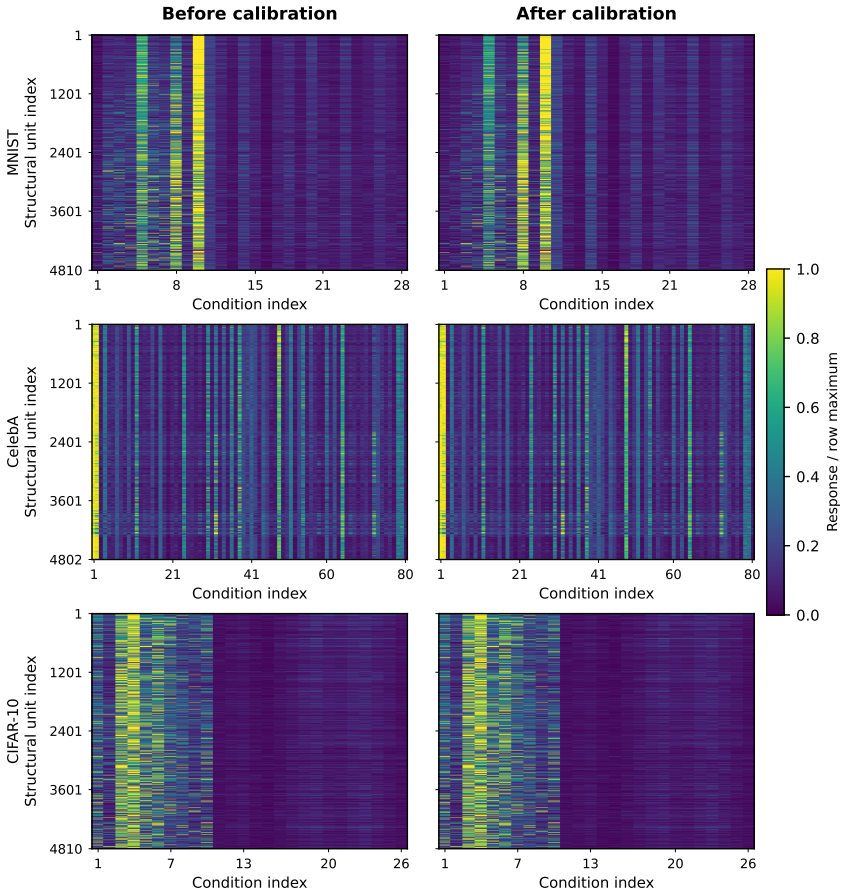


Figure 3: Paired structural-unit responses before and after calibration on the same seed-0 statistics split for MNIST, CelebA, and CIFAR-10. Rows retain structural-unit index order and are normalized by their own maximum separately at each stage; color therefore shows relative responses within a row, not changes in absolute gradient magnitude.

Figure Interpretation and Assessment. The left and right panels in each row correspond to the same structural unit, and the colors indicate the relative strengths of its responses to different conditions. Before calibration, multiple condition columns respond simultaneously for both MNIST and CIFAR-10. After calibration, the distributions of high-response columns change for some rows, but cross-condition responses remain present in all three datasets. Because CelebA contains more conditions, the number of visually prominent columns cannot be directly compared with those of the other two datasets. We therefore count the number of significantly responsive conditions within each dataset using the same threshold.

Setting $\eta = 0.5$ in the above definition, the proportion of structural units that respond significantly to at least two conditions decreases from 75.2% to 69.5% on MNIST, remains at 100.0% on CelebA, and increases from 79.3% to 80.3% on CIFAR-10. The corresponding mean numbers of significantly responsive conditions per row change from 2.26 to 2.11, from 6.05 to 5.85, and from 2.79 to 2.84, respectively. Thus, according to this threshold, calibration makes the responses slightly more concentrated on MNIST and CelebA, whereas CIFAR-10 does not exhibit the same trend, and simultaneous responses to multiple conditions remain present. This paired visualization provides only a descriptive diagnosis of the response distributions and cannot independently demonstrate that semantic features have been disentangled or that calibration yields causal performance gains. The sweep over $M = 4$–8 modules for CelebA and the evaluations of the candidate values $M = 12, 16, 20$

for CIFAR-10 reuse their respective post-calibration response matrices; the module count is varied only during the subsequent grouping stage.

These paired results support the modeling motivation described in the main text: a single structural unit may participate in multiple conditions, and the sets of primary conditions associated with candidate modules need not be mutually exclusive. The visualization and threshold-based statistics characterize only the distribution of local responses; whether a module affects predictions is still determined by the removal test in Appendix H.

E Two-Stage Gradient Update Procedure

To update the server-side global model θ using the loss function $\mathcal{L}_{\text{cal}}$, we adopt a more efficient two-stage distributed procedure to compute the global gradient of $\mathcal{L}_{\text{cal}}$. The entire global-model update process consists of multiple iterations. At iteration t, the conditional gradient of client k with respect to the parameters θ_{open} in the separable part of the model and the corresponding weighted gradient aggregated across all clients are respectively defined as $\boldsymbol{g}_{k,j}^{\ell,(t)} = \nabla_{\theta_{\text{open}}} F_{k,j}^{\ell}(\theta)\Big|_{\theta=\theta^{(t)}}$ and $\boldsymbol{g}_{j}^{\ell,(t)} = \sum_{k\in\mathcal{K}_j^{\ell}} \omega_{k,j}^{\ell} \boldsymbol{g}_{k,j}^{\ell,(t)}$. Note that the component of $\boldsymbol{g}_{j}^{\ell,(t)}$ corresponding to structural unit θ_q is precisely $\boldsymbol{g}_{q,j}^{\ell,(t)}$.

Stage 1. Each client computes $\boldsymbol{g}_{k,j}^{\ell,(t)}$ and uploads it to the server, which then constructs the loss function $\mathcal{L}_{\text{str}}$. Note that $\mathcal{L}_{\text{str}}$ is a function of the gradient $\boldsymbol{g}_{q,j}^{\ell,(t)}$. Therefore, the server can compute

$$\boldsymbol{a}_{q,j}^{\ell,(t)} = \frac{\partial \mathcal{L}_{\text{str}}}{\partial \boldsymbol{g}_{q,j}^{\ell,(t)}}. \tag{14}$$

Stage 2. The server sends $\boldsymbol{a}_{q,j}^{\ell,(t)}$ to the corresponding clients. Each client computes the gradient of $\boldsymbol{g}_{k,j}^{\ell,(t)}$ with respect to the parameters θ_{open} in the separable part of the model, denoted by $\boldsymbol{h}_{k,j}^{\ell,(t)} = \nabla_{\theta_{\text{open}}} \boldsymbol{g}_{k,j}^{\ell,(t)} = \nabla^2_{\theta_{\text{open}}} F_{k,j}^{\ell}(\theta)\Big|_{\theta=\theta^{(t)}}$. After obtaining $\boldsymbol{h}_{k,j}^{\ell,(t)}$, each client uploads it to the server for aggregation, yielding $\boldsymbol{h}_{j}^{\ell,(t)} = \sum_{k\in\mathcal{K}_j^{\ell}} \omega_{k,j}^{\ell} \boldsymbol{h}_{k,j}^{\ell,(t)}$. Similarly, the component of $\boldsymbol{h}_{j}^{\ell,(t)}$ corresponding to structural unit θ_q is denoted by $\boldsymbol{h}_{q,j}^{\ell,(t)}$. Therefore, the gradient of the server-side loss function $\mathcal{L}_{\text{str}}$ with respect to θ_{open} is

$$\nabla_{\theta_{\text{open}}} \mathcal{L}_{\text{str}}^{(t)} = \sum_{q=1}^{Q} \sum_{\ell\in\mathcal{I}} \sum_{j\in\mathcal{J}_\ell} \boldsymbol{h}_{q,j}^{\ell,(t)} \boldsymbol{a}_{q,j}^{\ell,(t)}. \tag{15}$$

Given the calibration step size $\eta_t > 0$ at iteration t, the model parameters are updated as

$$\theta_{\text{open}}^{(t+1)} = \theta_{\text{open}}^{(t)} - \eta_t \left[\frac{1}{|\mathcal{I}|} \sum_{\ell\in\mathcal{I}} \frac{1}{|\mathcal{J}_\ell|} \sum_{j\in\mathcal{J}_\ell} \boldsymbol{g}_{j}^{\ell,(t)} + 2\lambda_{\text{prox}} \left(\theta_{\text{open}}^{(t)} - \theta_{\text{open}}^{(0)} \right) + \nabla_{\theta_{\text{open}}} \mathcal{L}_{\text{str}}^{(t)} \right], \tag{16}$$

$$\theta_{\text{fix}}^{(t+1)} = \theta_{\text{fix}}^{(0)}. \tag{17}$$

Given the iteration thresholds $\delta_{\text{task}}, \delta_{\text{prox}} \geq 0$, the above iterative process continues until either of the following conditions is satisfied:

$$\mathcal{L}_{\text{task}}^{(t)} - \mathcal{L}_{\text{task}}^{(0)} \leq \delta_{\text{task}}, \qquad \mathcal{L}_{\text{prox}}^{(t)} \leq \delta_{\text{prox}}.$$

After the iterations are completed, the final updated parameters of the complete model are denoted by

$$\theta^{*} = \left[\left(\theta_{\text{open}}^{(*)} \right)^{\mathsf{T}}, \left(\theta_{\text{fix}}^{(0)} \right)^{\mathsf{T}} \right]^{\mathsf{T}}.$$

F Conditional Gradient for Local Sensitivity Analyses

This section justifies the choice of the conditional gradient as the response of structural units in our paper. With parameter θ^* and the conditional loss $F_j^{\ell,\mathrm{stat}}$ on the set, if the loss function is differentiable at θ^* and only the parameters of structural unit q are perturbed, the first-order expansion yields

$$F_j^{\ell,\mathrm{stat}}(\theta^* + \iota_q \Delta\theta_q) - F_j^{\ell,\mathrm{stat}}(\theta^*) = (g_{q,j}^{\ell,\mathrm{stat}})^\top \Delta\theta_q + o(\|\Delta\theta_q\|_2), \tag{15}$$

where ι_q embeds the within-group perturbation into the full model parameters. For $\|\Delta\theta_q\|_2 \le \varepsilon$, the Cauchy–Schwarz inequality implies that the maximum absolute value of the first-order variation is $\varepsilon\|g_{q,j}^{\ell,\mathrm{stat}}\|_2$. $R_{q,j}^{\ell,\mathrm{stat}}$ is further divided by $\sqrt{d_q}$ to mitigate the effect of dimensionality differences across groups. Therefore, the response matrix measures the conditional loss sensitivity in the neighborhood of a model's parameter, and is further used to construct modules.

G Cosine-Centroid Clustering under Dual Capacity Constraints

For problem $[\mathcal{P}_1]$ in the main text, we fix the candidate number of modules M and let $u_{qm} \in \{0,1\}$ indicate whether active response group $q \in \mathcal{Q}_{\mathrm{act}}$ is assigned to module m. We first initialize M unit centroids $\boldsymbol{c}_m$ from the unit response vectors $\boldsymbol{r}_q$. With the centroids fixed, we solve the following capacity-constrained assignment subproblem:

$$\begin{aligned}
\max_{\{u_{qm}\}} \quad & \sum_{q\in\mathcal{Q}_{\mathrm{act}}} \sum_{m=1}^{M} u_{qm} \boldsymbol{r}_q^\top \boldsymbol{c}_m, \\
\text{s.t.} \quad & \sum_{m=1}^{M} u_{qm} = 1 \quad (\forall q), \qquad \sum_{q\in\mathcal{Q}_{\mathrm{act}}} u_{qm} \ge 1 \quad (\forall m), \\
& \sum_{q\in\mathcal{Q}_{\mathrm{act}}} \nu_q u_{qm} \le \frac{\alpha}{M} \sum_{q\in\mathcal{Q}_{\mathrm{act}}} \nu_q, \qquad \sum_{q\in\mathcal{Q}_{\mathrm{act}}} \mu_q u_{qm} \le \frac{\alpha}{M} \sum_{q\in\mathcal{Q}_{\mathrm{act}}} \mu_q \quad (\forall m).
\end{aligned} \tag{16}$$

This binary assignment problem can be solved using integer programming. If the integer-programming solver determines that the constraints are infeasible for a fixed M, the corresponding candidate partition is not generated. After assignment, we set $\mathcal{I}_m = \{q : u_{qm} = 1\}$ and normalize $\sum_{q\in\mathcal{I}_m} \boldsymbol{r}_q$ according to the definition in the main text to obtain the updated centroid. Assignment and centroid updates are then repeated until the objective value no longer improves or a predefined maximum number of iterations is reached. We run this procedure with multiple initializations and retain the feasible result with the highest objective value. Each assignment satisfies both capacity upper bounds as well as the nonempty and mutually exclusive partition constraints. When the assignment subproblem is solved exactly, the alternating updates guarantee that the objective value does not decrease, but they do not guarantee global optimality. We therefore use this procedure as an approximate solution method for problem $[\mathcal{P}_1]$.

H Module Removal Test on the Selected Set

The response $e_{m,j}^{\ell}$ measures the sensitivity of a module to different conditional features, but does not by itself establish the module's contribution to prediction. We therefore perform a module-removal test on the selection set. For any retained-module set $A \subseteq \mathcal{M}$, let f_{θ_A} denote the corresponding model and define its mean task loss under condition v_j^{ℓ} as

$$\mathcal{L}_j^{\ell}(A) = \frac{1}{n_j^{\ell}} \sum_{k=1}^{K} \sum_{(x,y)\in\mathcal{D}_{k,j}^{\ell}} \ell\big(f_{\theta_A}(x), y\big). \tag{8}$$

After removing module m, the change in loss under condition v_j^ℓ is

$$\Delta_{m,j}^\ell = \mathcal{L}_j^\ell(\mathcal{M} \setminus \{m\}) - \mathcal{L}_j^\ell(\mathcal{M}). \tag{9}$$

When $\Delta_{m,j}^\ell > 0$, removing module m increases the task loss under condition v_j^ℓ, providing evidence that the module contributes to predictions for that condition. Ranking the values of $\Delta_{m,j}^\ell$ identifies conditions with the largest loss increase after removal. Equation (9) thus complements the response-based analysis with an observed change in prediction loss. The removal test neither replaces the definition of a module's primary conditions nor establishes that the module has an independent or uniquely causal function.

I Supplementary Experimental Results

This section sequentially examines per-client performance, module granularity, resource usage, and the selection process. Controlled pruning examines whether condition-aware selection near the same converged model can maintain performance; end-to-end comparisons examine the competitiveness of the post-training pipeline relative to directly training heterogeneous models. The two types of protocols are not merged into a single leaderboard. All conclusions are based on the locked test set, and the calibration and module selection stages do not access test labels.

Boundaries of Result Interpretation. The same-budget baseline comparison in the main text uses a single locked seed; for CIFAR-10, MDMP has been supplemented with three independent training and client-partition seeds, but the competing methods do not have three-seed results under the same protocol. Therefore, we do not claim statistically significant cross-method superiority on this basis. End-to-end methods directly train heterogeneous subnetworks, whereas MDMP selects modules from a converged model; this protocol difference, along with the fact that the measured MACs are not strictly equal, should be interpreted in conjunction with Table 5 . Some end-to-end baselines on MNIST and HeteroFL on CIFAR-10 achieve higher accuracy; these results limit the applicable scope of MDMP's current conclusions.

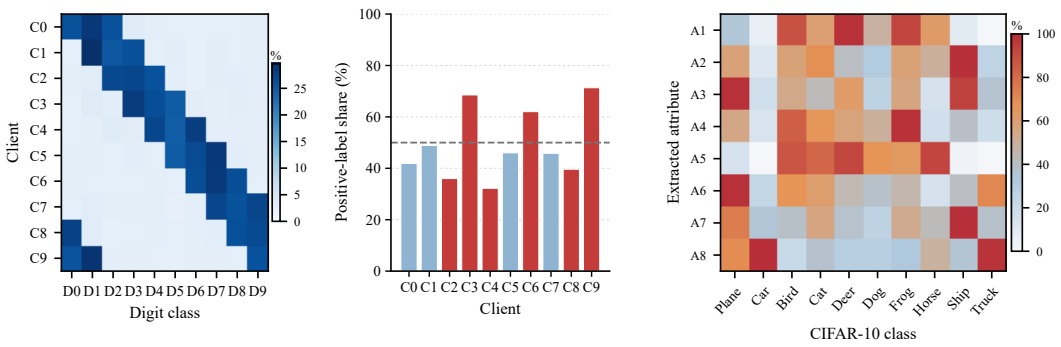
Table 3 summarizes the training settings for controlled selection and end-to-end comparison. The two types of experiments adopt different model selection pipelines; the target MAC ratios in the table are the specified budgets, and the actual ratios should still be based on the measured values in Table 5.

Table 3: Training and evaluation settings for controlled selection and end-to-end baselines.

Setting	MNIST	CelebA	CIFAR-10 MDMP	CIFAR-10 E2E
Clients	10	10	10	10
Global rounds	50	30	60	60
Local epochs	1	1	1	1
Batch size	64	128	128	128
Initial learning rate	0.03	0.05	0.1	0.1
Momentum	0.9	0.9	0.9	0.9
Weight decay	0	5e-4	5e-4	5e-4
Conditions	28	80	26	–
Modules reported	M8	M4–M8	M12/16/20; M20 final	Nested widths
Target MAC budgets	95, 90	95, 90, 85, 80	95, 90	95, 90
Locked seeds	0,1,2	0,1,2	0,1,2	0

I.1 Fluctuation of Multi-Seed Results

Figure 4 is obtained based on metadata of clients. We can observe that 1) label skew in MNIST means that clients encounter different digits; 2) Attribute proportions in CelebA change the composition of local positive and negative samples; 3) Class-dependent attribute



(a) MNIST label skew (b) CelebA label prevalence (c) CIFAR-10 attribute heterogeneity

Figure 4: Different heterogeneities: MNIST label skew, CelebA positive-attribute prevalence, and class-dependent attribute activation in CIFAR-10.

responses in CIFAR-10 cause the same module to operate under different visual conditions. These observations explain why pruning method based on the weight magnitudes of global model may not be suitable for every client.

Table 4 summarizes three completed seed-level MDMP runs. The reported error is the sample standard deviation of the client-averaged results across the three seeds ($n = 3$); the ten clients are not treated as independent repetitions. MNIST and CelebA both use $M = 8$ modules. This CelebA module-count setting does not directly correspond to the single-seed method comparison in Table 1. For CIFAR-10, the three-seed results with $M = 20$ cover both the 95% and 90% target budgets, with training and client partitioning repeated independently for each seed. The comparison methods remain single-seed results, so no inferred error terms are reported for them.

Table 4: Three-seed MDMP checks at the indicated target MAC budgets. Entries are mean $\pm$ sample standard deviation across seeds ($n = 3$), not across clients.

Dataset	Mean client accuracy (%)	Measured MAC (%)
MNIST ($M = 8$)	98.81 ± 0.35	93.49 ± 0.68
CelebA ($M = 8$)	92.51 ± 0.08	90.76 ± 2.06
CIFAR-10 ($M = 20$, 95%)	87.70 ± 0.93	92.39 ± 0.24
CIFAR-10 ($M = 20$, 90%)	87.50 ± 0.51	89.21 ± 0.55

The repeated CIFAR-10 runs further assess the compute budget and client-level constraint. At the 95% target budget, the seed-level mean accuracy drop is 0.72 ± 0.19 percentage points; at the 90% target budget, it is 0.91 ± 0.38 percentage points. The largest single-client drop observed across the three seeds is 2.88 percentage points, below the predefined 5-percentage-point limit. Figure 5 shows each seed's accuracy against measured MACs and its maximum client accuracy drop. Different budgets can yield different feasible physical subnetworks, so accuracy need not change monotonically for every seed. Seed 0 was confirmed retrospectively during development, whereas seeds 1 and 2 independently repeated training and client partitioning; all three used the same official test set. These results assess repeatability and do not constitute a new blind evaluation.

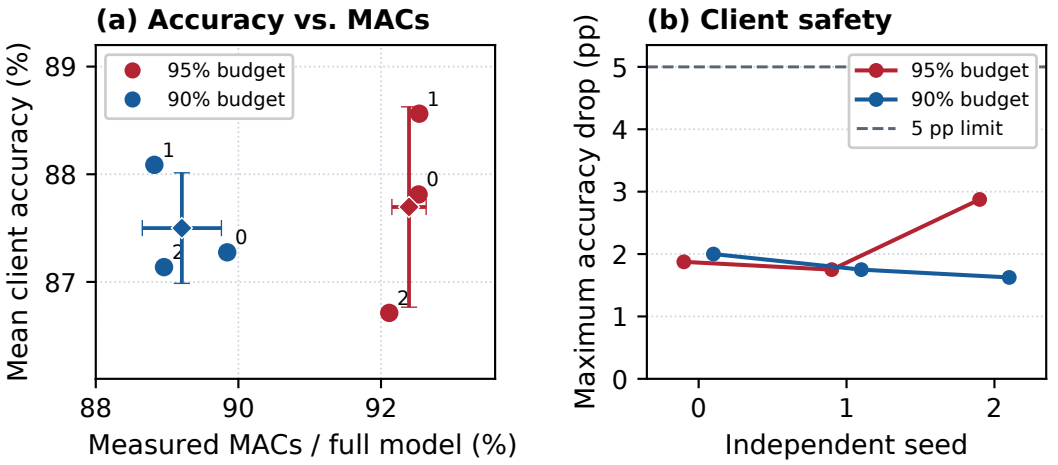


Figure 5: CIFAR-10 MDMP stability over three independent seeds. Left: seed-level accuracy and measured MAC, with diamonds and error bars showing the mean and sample standard deviation. Right: the largest client accuracy drop within each seed; the dashed line marks the 5 pp limit.

I.2 Client-Level Performance Retention

Figure 6 pairs each client's accuracy under the full model with its accuracy under the selected subnetwork. The smaller changes on CelebA may indicate that the retained attribute responses cover most clients. On MNIST and CIFAR-10, the drops are concentrated among a few clients, consistent with uneven dependence on local features under non-IID data. Connecting lines indicate paired observations, not independent samples.

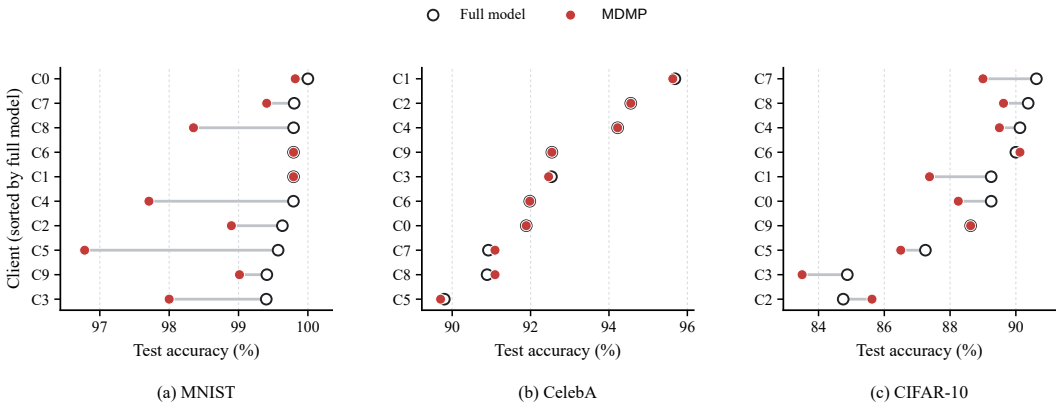
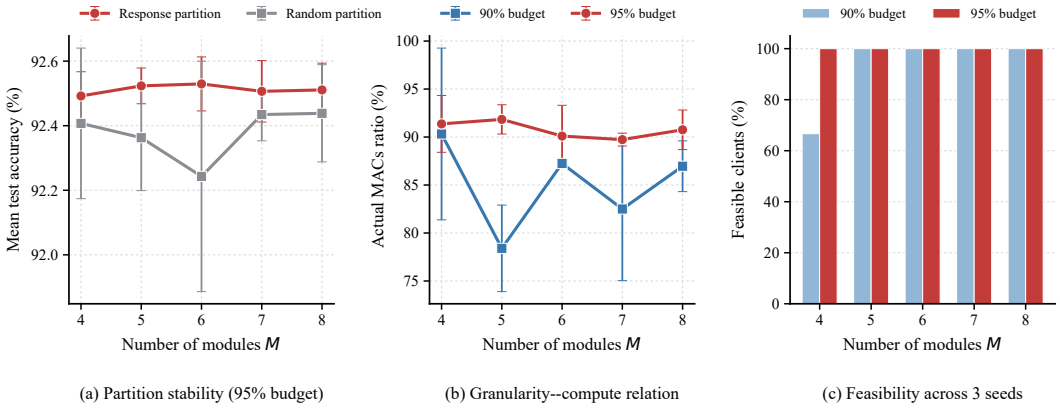


Figure 6: Per-client accuracy at the 95% target MACs budget. Each line pairs one client's full-model and MDMP-subnetwork results.

I.3 Module Granularity and Multi-Seed Stability

Figure 7 summarizes the module-count sweep for CelebA. Under three seeds, the 95% budget maintains an average accuracy of about 92.5% over the range $M = 4$ to $M = 8$; increasing the number of modules makes the selection finer but also enlarges the candidate combinations, while coarser groupings may fail to precisely match the target computation under the 90% budget. Error bars come only from the three seeds.

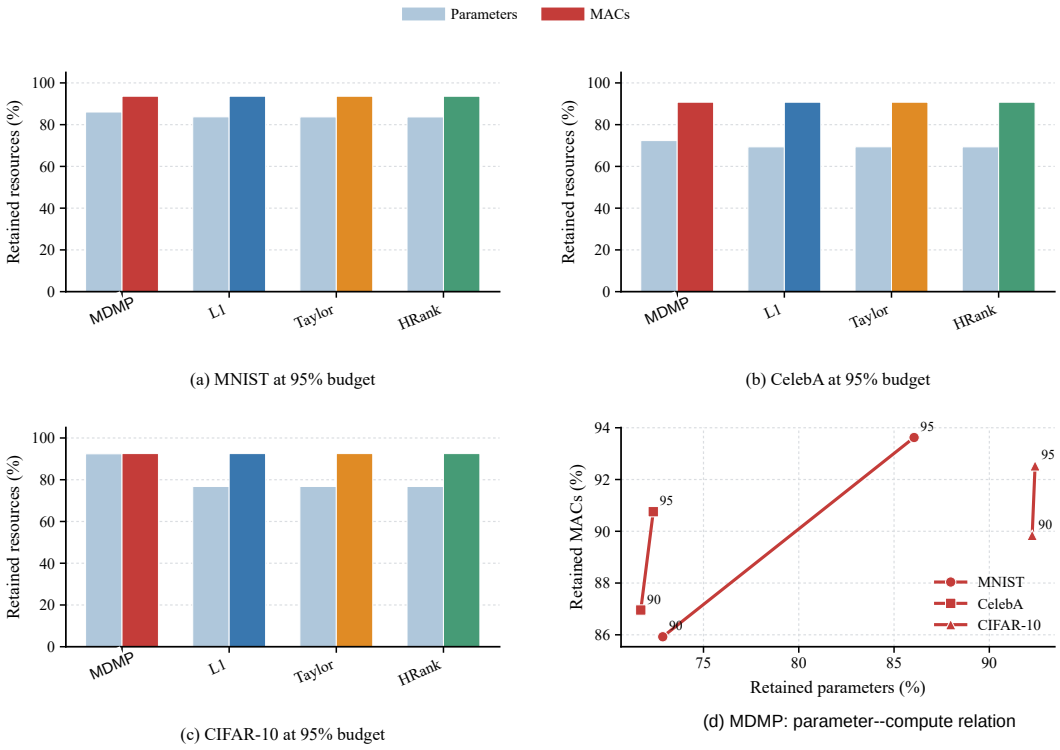


(a) Partition stability (95% budget) (b) Granularity--compute relation (c) Feasibility across 3 seeds

Figure 7: CelebA module-count sweep: accuracy, measured MAC, and client feasibility. Error bars summarize variation across three seeds.

I.4 Resource Auditing and Multi-Seed Results

Figure 8 reports the retained parameter ratio and the measured MAC separately. Convolutional layers, classification heads, and normalization parameters have different invocation frequencies, so the same parameter retention ratio does not necessarily bring the same computational savings; therefore, resource gains should be verified against the actual MACs.



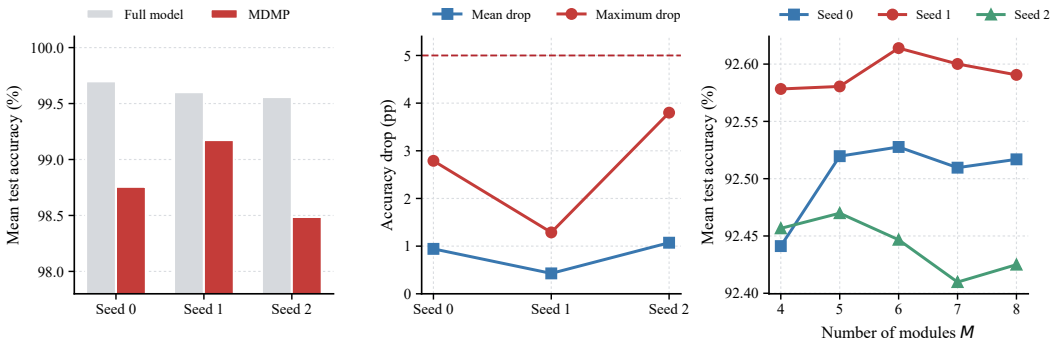
(a) MNIST at 95% budget (b) CelebA at 95% budget

(c) CIFAR-10 at 95% budget (d) MDMP: parameter--compute relation

Figure 8: Resource audit on three datasets: retained parameters, measured MAC, and the parameter–compute relation of MDMP.

Figure 9 shows the accuracy and client drop for three seeds on MNIST, as well as the module-count sweep for CelebA. Variation across seeds indicates that client data partitioning and

training trajectories affect module responses; therefore, the locked-seed results in the main text should not be misread as having zero variance.

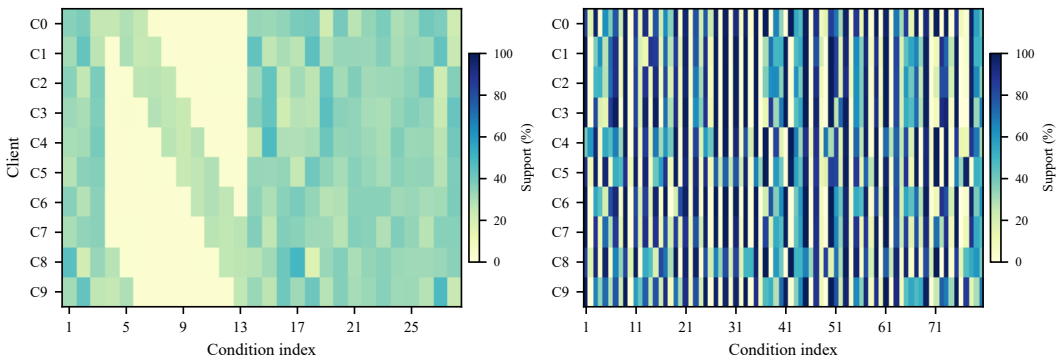


(a) MNIST seed-level accuracy (b) MNIST client-safety stability (c) CelebA response partition

Figure 9: Multi-seed stability: MNIST accuracy and client drops, and CelebA response partitions. The dashed line marks the client-drop limit.

I.5 Condition Coverage and Module–Condition Response

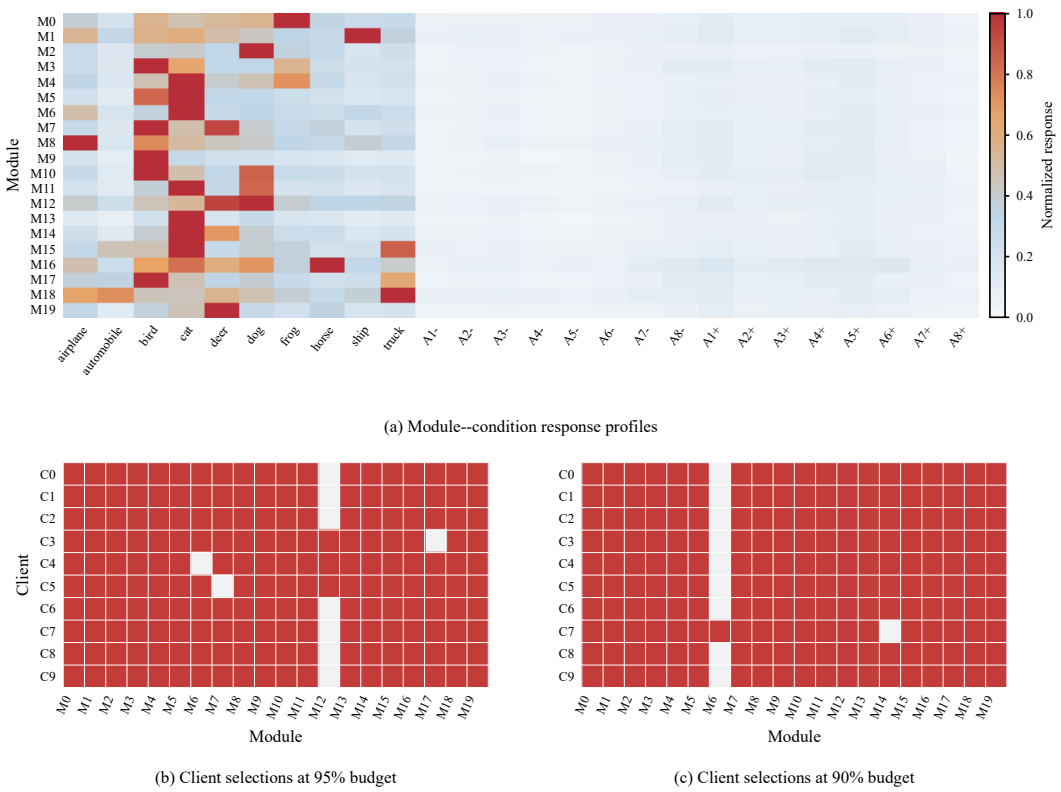
Figure 10 first examines the client–condition coverage for MNIST and CelebA. If a condition appears in only a few clients, its estimated condition gradient is easily affected by sample size; therefore, a low response does not necessarily mean that the module is insensitive to that condition.



(a) MNIST: 28 conditions (b) CelebA: 80 conditions

Figure 10: Client–condition coverage for MNIST and CelebA. Color encodes the sample-support fraction for each condition.

Figure 11 further shows the module–condition response for CIFAR-10 and the client selection under two budget levels. The response patterns and the conditions covered by each client jointly constrain the selectable modules; therefore, different clients may obtain different combinations even with the same resource budget.



(a) Module--condition response profiles

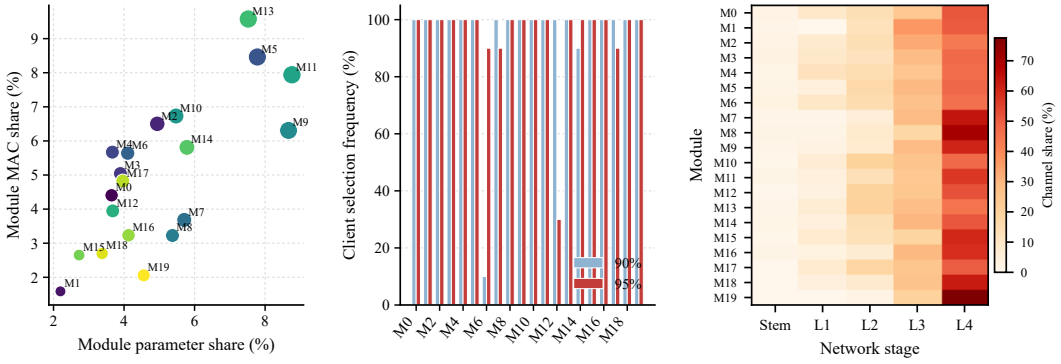
(b) Client selections at 95% budget

(c) Client selections at 90% budget

Figure 11: CIFAR-10 module–condition responses and client module choices at two target budgets.

I.6 Module Composition and Selection–Test Consistency

Figure 12 audits candidate modules in terms of resource proportion, selection frequency, and network-stage composition. If high-frequency modules are concentrated in certain stages, this indicates that the features at those stages may be shared by multiple clients; however, frequency alone does not prove that a module has a single semantic meaning.



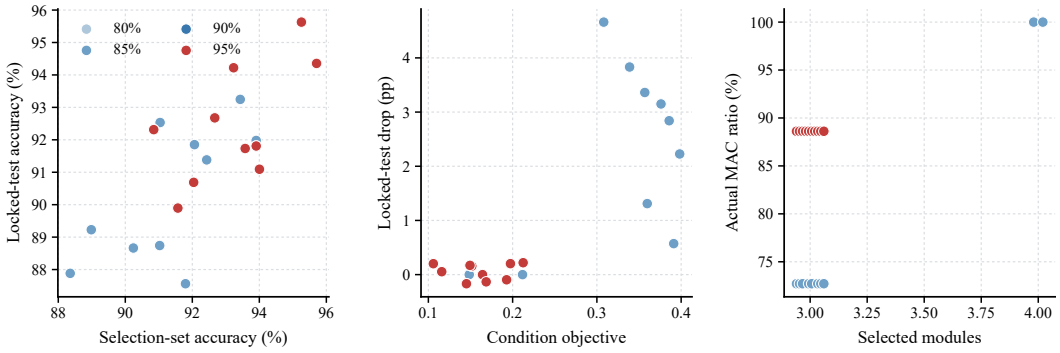
(a) Module resource footprints

(b) Selection frequency

(c) Layer-stage composition

Figure 12: Module composition and usage: resource fraction, client selection frequency, and network-stage makeup.

Figure 13 compares the results on the selection set and the locked test set. If the two are largely consistent, this indicates that the module ranking does not obviously depend on random fluctuations in the selection set; a small number of deviations still suggest changes in the condition distribution or finite-sample estimation error. The test set is used only for final verification and does not participate in re-selecting modules.

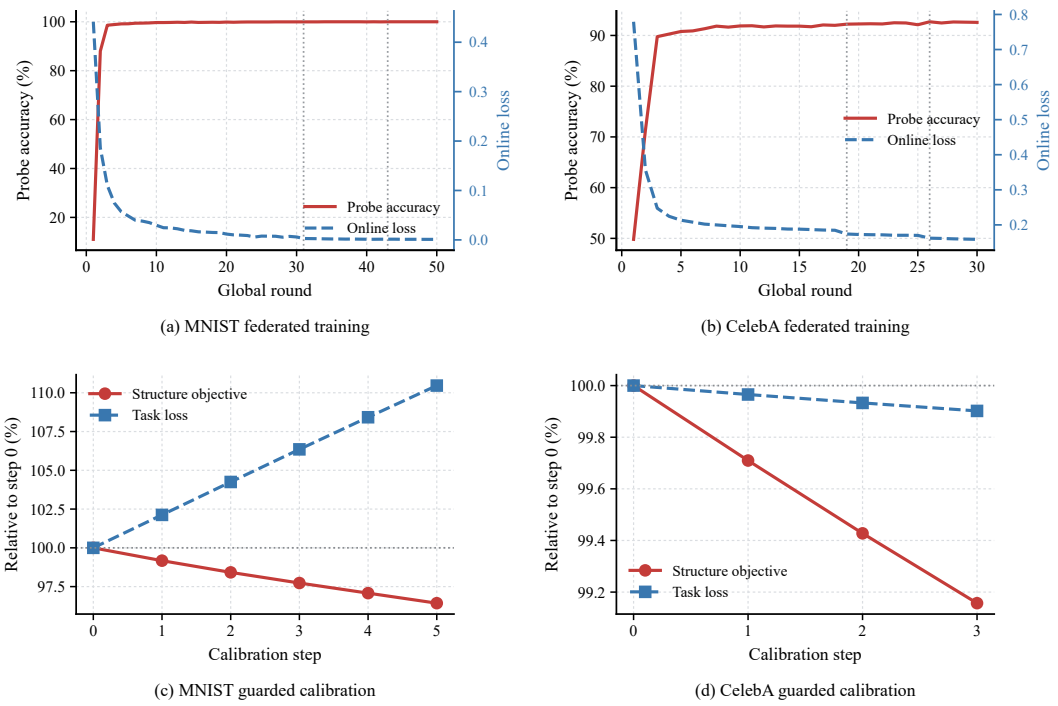


(a) Selection-to-test agreement (b) Objective-to-test relation (c) Module count-to-compute relation

Figure 13: Selection-to-locked-test consistency, alongside the condition objective, module count, and measured resource use.

I.7 Training and Calibration Trajectories

Figure 14 shows the federated training and limited-step calibration trajectories. During the calibration stage, the structural objective decreases while the task loss does not rise sharply in sync, which is consistent with the constrained objective of "adjusting condition responses while preserving the original task performance as much as possible"; the trajectories only describe this run and cannot serve as a universal proof of convergence.



(a) MNIST federated training

(b) CelebA federated training

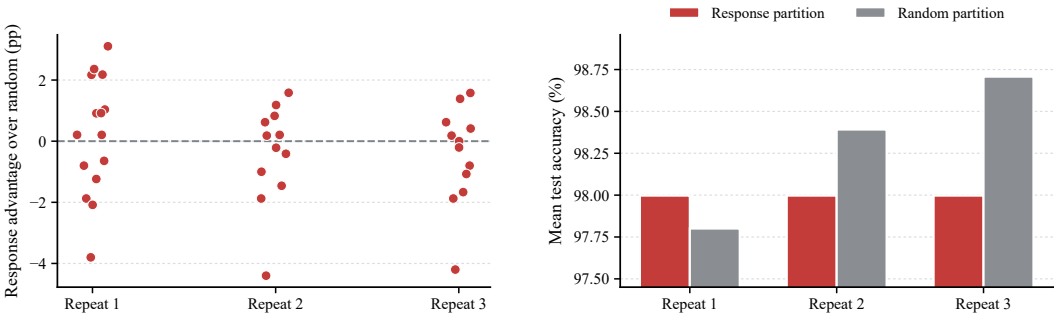
(c) MNIST guarded calibration

(d) CelebA guarded calibration

Figure 14: Federated training and constrained calibration trajectories on MNIST and CelebA.

I.8 Same-MAC Random Control

The same-MAC random control in Figure 15 did not let the response partition win across all three repetitions: the pooled random partition achieved an average accuracy of 98.26%, while the response partition achieved 98.00%. MNIST accuracy is already close to saturation, and fluctuations from a few clients or random partitions can change the mean; therefore, this result does not support the conclusion that "response partition is generally superior to random partition."



(a) Paired client-level differences

(b) Same-MAC random control

Figure 15: MNIST equal-MAC random control: paired client differences and mean test accuracy across three repeats.

Table 5: Measured MAC ratio (%) in the end-to-end comparison. Column headers are target budgets; entries are actual computation relative to the full model.

	MNIST		CelebA		CIFAR-10	
Method	95%	90%	95%	90%	95%	90%
HeteroFL	89.34	80.35	89.57	80.88	92.02	89.37
FedRolex	89.34	80.35	89.57	80.88	92.02	89.37
FjORD (no KD)	89.34	80.35	89.57	80.88	92.02	89.37
Full model	100.00	100.00	100.00	100.00	100.00	100.00
MDMP	**93.62**	**85.92**	**90.76**	**86.96**	**92.53**	**89.84**

Table 6: Worst-client test accuracy (%) in the end-to-end comparison. One MNIST client exceeds the 5 pp drop limit for MDMP at the 90% target budget.

	MNIST		CelebA		CIFAR-10	
Method	95%	90%	95%	90%	95%	90%
HeteroFL	99.21	99.20	89.13	89.51	84.88	85.50
FedRolex	98.82	99.21	88.66	89.51	85.00	82.00
FjORD (no KD)	99.21	99.21	89.23	89.04	79.62	80.25
Full model	99.40	99.40	89.51	89.51	84.75	84.75
MDMP	**96.78**	**94.00**	**89.55**	**89.58**	**83.50**	**82.75**