PFEDMA: A FEDERATED LEARNING ALGORITHM WITH PRUNED MATCHED AVERAGING

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated aggregation-based methods such as FedAvg and FedProx perform aggregation across the same parameter indices of locally trained client models. This assumes that the clients represent comparable features at the same indices, which is not necessarily true, as fully connected and convolutional layers exhibit permutation invariance with respect to neurons or filters within a layer. FedMA was introduced to address this mismatch by probabilistically matching neurons or channels layer by layer before performing aggregation. However, unmatched units from client models are appended to the global model. This causes layer-wise growth, which is particularly difficult to accommodate in networks such as ResNets, where channel dimensions are constrained by cross-layer connections. We introduce Pruned Federated Matched Averaging (PFedMA), a variant of FedMA that preserves a fixed model topology. Rather than retaining every unmatched unit, PFedMA discards unmatched non-donor units and fills vacant global positions using unmatched units from a randomly selected donor client. We further extend matched aggregation to be compatible with residual networks. We do this by enforcing consistent channel assignment across Batch Normalization layers and residual connections. We evaluate PFedMA on four image-classification benchmarks under IID and multiple Dirichlet non-IID partitions, using both sequential CNNs and ResNet-18. FedMA obtains higher accuracy in some sequential-CNN settings. On CIFAR-10 with ResNet-18, PFedMA substantially outperforms the evaluated parameter-wise baselines across the tested partitions. These results suggest that residual networks can also benefit from permutation-aware aggregation.

1 INTRODUCTION

Federated learning (FL) enables multiple clients to collaboratively train a shared model without transferring their raw data, making it well suited to settings where data sharing is restricted by privacy, governance, or communication constraints (Wen et al., 2023). A central challenge in FL is statistical heterogeneity: clients may observe substantially different data distributions, causing locally trained models to diverge and making their aggregation increasingly difficult.

The foundational FEDAVG algorithm (McMahan et al., 2017) constructs a global model through coordinate-wise averaging of local parameters. Although effective in many settings, coordinate-wise aggregation does not account for the permutation symmetries of neural networks. Independently trained models with identical architectures can learn corresponding features at different neuron or channel indices. Consequently, parameters occupying the same position across clients need not represent corresponding features, and directly averaging them can degrade useful client representations, particularly under heterogeneous data.

FEDMA (Wang et al., 2020) addresses this problem by matching neurons or convolutional channels across client models before aggregation. Matched units are combined, while unmatched units may be introduced as new global neurons. This allows the global model to adapt its representation but can progressively alter layer widths and increase the size of the resulting network. Dynamic layer dimensions also complicate the application of matched aggregation to architectures with explicit structural dependencies, such as residual networks, where the outputs of multiple branches must share compatible channel dimensions and orderings.

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.

We introduce PFEDMA, a fixed-width variant of FEDMA that retains permutation-aware aggregation while preserving the prescribed global architecture. Unmatched non-donor units are excluded from the global update, while unmatched units from a randomly selected donor client are used to complete vacant global positions. This prevents matching-induced model growth and maintains constant layer dimensions across communication rounds.

We further extend matched aggregation to residual architectures. Batch Normalization parameters and running statistics are aligned using the assignments of their corresponding convolutional channels, while residual and shortcut branches are constrained to use consistent channel assignments at element-wise additions. Projection shortcuts are similarly aligned to the input and output channel bases of their corresponding transition blocks. These modifications enable architecture-preserving, permutation-aware aggregation of RESNET-based models under federated data heterogeneity.

2 RELATED WORKS

To address the instability of local optimisation under heterogeneous data, FEDPROX (Li et al., 2020) incorporates a proximal term into the objective to restrict deviations of local model updates from the global model, thereby improving convergence under data heterogeneity. Its behaviour depends on the proximal parameter μ: weak regularisation may provide insufficient control over local drift, whereas excessive regularisation can restrict useful local adaptation.

SCAFFOLD (Karimireddy et al., 2020) addresses client drift by introducing control variates that estimate the difference between local and global update directions. Rather than directly restricting local updates, it corrects the drift accumulated during local optimisation using client-side and server-side control variates. While FEDPROX and SCAFFOLD modify the optimisation dynamics to mitigate client drift, neither explicitly considers whether the internal features learned by different clients are structurally aligned.

FEDDYN (Acar et al., 2021) uses dynamic regularisation to introduce a correction term that aligns local and global optima, thereby mitigating client drift. However, like other optimisation-based approaches, it primarily addresses divergence in the optimisation process and does not explicitly establish a matching between the internal representations learned by different clients.

ELASTIC AGGREGATION (Chen et al., 2023) dynamically weights parameter updates according to their sensitivity, suppressing updates to sensitive parameters while allowing greater adaptation of less sensitive parameters. However, it operates at the parameter level without explicitly addressing representation misalignment between clients.

Probabilistic Federated Neural Matching (PFNM) (Yurochkin et al., 2019) frames model fusion as a probabilistic matching problem, which allows client (local) neurons to be matched with existing global neurons or added as new neurons. However, it is mainly designed for fully connected networks and can increase the size of the global model.

FEDMA (Wang et al., 2020) takes a fundamentally different approach by explicitly considering the learned representations of client models. Instead of assuming that parameters at corresponding architectural positions represent equivalent features, building on PFNM, FEDMA performs layer-wise matching of neurons or convolutional kernels before aggregation, while unmatched components can be incorporated into the global model.

This directly addresses the permutation-invariance problem not considered by parameter-wise aggregation methods. However, incorporating unmatched neurons can dynamically increase layer widths, resulting in additional computational and memory requirements. Dynamic layer dimensions are also problematic for architectures with structural dependencies: in residual networks such as ResNet, the residual and skip branches must retain compatible channel dimensions and orderings.

3 METHODOLOGY

Let J participating clients share a neural-network architecture with N matchable layers. For layer n, let $L_{j,n}$ denote the number of local units or convolutional output channels at client j, and let L_n denote the prescribed width of the corresponding global layer. The parameters associated with

local unit ℓ are denoted by $w_{j\ell}^{(n)} \in \mathbb{R}^{D_n}$, where convolutional filters are vectorised along their input-channel and spatial dimensions. The global parameters for layer n are represented by $W_n = \{\theta_g^{(n)}\}_{g=1}^{L_n}$.

Permutation invariance and layer-wise matching. Neural networks admit equivalent parameterisations under permutations of their hidden units. Consider two consecutive fully connected layers with weights W_1 and W_2, and let $\Pi \in \mathbb{R}^{L\times L}$ be a permutation matrix acting on a hidden layer of width L. For an element-wise activation function σ, simultaneously permuting the hidden units and the corresponding inputs of the subsequent layer preserves the network function:

$$\hat{y} = \sigma(xW_1\Pi)\,\Pi^\top W_2. \tag{1}$$

An analogous symmetry holds for convolutional networks, where a permutation of the output channels of one convolution must be applied to the input-channel dimension of each subsequent consumer of that representation. Consequently, independently trained client models may encode corresponding features at different parameter indices, and coordinate-wise averaging may combine non-corresponding units despite the clients sharing an identical architecture.

Following FEDMA (Wang et al., 2020), PFEDMA addresses this ambiguity through progressive, layer-wise matching. For client j and layer n, we define a partial assignment matrix

$$\Pi^{(j,n)} \in \{0,1\}^{L_{j,n}\times L_n}, \tag{2}$$

where $\pi_{\ell g}^{(j,n)} = 1$ indicates that local unit ℓ is assigned to global position g. Each local unit is assigned to at most one global position, and each client contributes at most one local unit to a given global position. For sequential architectures, the assignment obtained for layer n is also applied to the input-channel dimension of layer $n+1$, after which the matched prefix is reconstructed and the remaining client-specific suffix is locally adapted before matching proceeds.

Pruned fixed-width aggregation. In FEDMA, local units that cannot be associated with an existing global unit may be appended to the global layer, allowing its width to increase across matching steps and communication rounds. PFEDMA instead constrains every global layer to its prescribed width L_n. For global position g, let

$$\mathcal{S}_g^{(n)} = \left\{(j,\ell) \;:\; \pi_{\ell g}^{(j,n)} = 1\right\} \tag{3}$$

denote the set of matched local units contributing to that position. When $\mathcal{S}_g^{(n)}$ is non-empty, its members are combined using the probabilistic aggregation operator $\mathcal{A}_n(\cdot)$ defined in the following paragraph.

For each layer n, a donor client $d_n \in \{1,\dots,J\}$ is selected from the participating clients; our implementation selects the donor uniformly at random. Unmatched units from clients $j \neq d_n$ are pruned from the global update. Unmatched units belonging to the donor are instead used to complete global positions that remain vacant after matching. Let $\mathcal{V}_n$ denote the set of vacant global positions, and let $\rho_n(g)$ associate each $g \in \mathcal{V}_n$ with a distinct unmatched unit of donor d_n. The resulting global parameter is

$$\theta_g^{(n)} = \begin{cases} \mathcal{A}_n\left(\left\{w_{j\ell}^{(n)} : (j,\ell) \in \mathcal{S}_g^{(n)}\right\}\right), & \mathcal{S}_g^{(n)} \neq \emptyset, \\ \mathcal{A}_n\left(\left\{w_{d_n,\rho_n(g)}^{(n)}\right\}\right), & g \in \mathcal{V}_n. \end{cases} \tag{4}$$

Thus, the term *pruned* refers specifically to excluding unmatched non-donor units from the global aggregation. Donor completion maintains $|W_n| = L_n$ for every layer and prevents the matching-induced architecture growth of FEDMA.

Probabilistic matching and assignment. PFEDMA retains the probabilistic neuron-matching objective underlying PFNM and FEDMA (Yurochkin et al., 2019; Wang et al., 2020), while modifying the treatment of units that are not assigned to an existing global position. At layer n, each global unit is assigned a Gaussian prior with mean $\mu_{0,n}$ and precision $\tau_{0,n}$, while the local parameters are modelled as noisy observations of their corresponding global units with precision $\tau_{j,n}$. Our implementation uses an isotropic zero-mean prior and a shared layer-wise local precision, with $\tau_{0,n} = 1/\sigma_{0,n}$ and $\tau_{j,n} = 1/\sigma_n$.

For a global position g with matched set $\mathcal{S}_g^{(n)}$, define the precision-weighted sufficient statistics

$$q_g^{(n)} = \tau_{0,n}\mu_{0,n} + \sum_{(j,\ell)\in\mathcal{S}_g^{(n)}} \tau_{j,n} w_{j\ell}^{(n)}, \tag{5}$$

$$s_g^{(n)} = \tau_{0,n} + \sum_{(j,\ell)\in\mathcal{S}_g^{(n)}} \tau_{j,n}. \tag{6}$$

The aggregation operator introduced in Eq. 4 is the posterior mean

$$\mathcal{A}_n\left(\left\{w_{j\ell}^{(n)}\right\}_{(j,\ell)\in\mathcal{S}_g^{(n)}}\right) = \frac{q_g^{(n)}}{s_g^{(n)}}. \tag{7}$$

Assignments are obtained by evaluating the increase in the probabilistic objective produced by associating each local unit with each existing global position. For local unit ℓ of client j, the score for matching it to global position g is

$$C_{\ell g}^{(j,n)} = \frac{\left\|q_g^{(n)} + \tau_{j,n} w_{j\ell}^{(n)}\right\|_2^2}{s_g^{(n)} + \tau_{j,n}} - \frac{\left\|q_g^{(n)}\right\|_2^2}{s_g^{(n)}} + \log\frac{m_g^{(n)}}{J - m_g^{(n)}}, \tag{8}$$

where $m_g^{(n)}$ is the current popularity count of global position g, corresponding to the number of other clients supporting that position. The first two terms measure the change in the precision-weighted posterior objective, while the popularity term favours correspondences supported by multiple clients.

The matching matrix also contains candidate unmatched positions. The score for assigning local unit ℓ to the k-th such position is

$$\begin{aligned} C_{\ell k}^{(j,n),\mathrm{unm}} = \frac{\left\|\tau_{0,n}\mu_{0,n} + \tau_{j,n} w_{j\ell}^{(n)}\right\|_2^2}{\tau_{0,n} + \tau_{j,n}} - \frac{\|\tau_{0,n}\mu_{0,n}\|_2^2}{\tau_{0,n}} \\ - 2\log k + 2\log\left(\frac{\gamma_n}{J}\right), \end{aligned} \tag{9}$$

where γ_n controls the relative preference for leaving a local unit unmatched. The Hungarian algorithm is applied to the combined score matrix to obtain a one-to-one assignment that maximises the total matching score. In FEDMA, assignment to a new-unit position expands the global layer. In PFEDMA, architecture expansion is disabled: such an assignment instead marks the local unit as unmatched. After alternating assignment refinement across clients, unmatched non-donor units are pruned, while unmatched units from donor d_n complete the remaining global positions according to Eq. 4. The initial global sufficient statistics are seeded from a client with maximal layer width, and the assignments are subsequently refined over randomly permuted client orders for a fixed number of iterations.

Batch Normalization alignment. For a convolutional layer, Batch Normalization associates each output channel ℓ with learned affine parameters $\gamma_{j\ell}^{(n)}$ and $\beta_{j\ell}^{(n)}$, together with running statistics $m_{j\ell}^{(n)}$ and $v_{j\ell}^{(n)}$. We denote this channel-dependent state by

$$b_{j\ell}^{(n)} = \left(\gamma_{j\ell}^{(n)}, \beta_{j\ell}^{(n)}, m_{j\ell}^{(n)}, v_{j\ell}^{(n)}\right). \tag{10}$$

Because normalization is applied independently to each output channel, the Batch Normalization state must follow the same assignment as its corresponding convolutional filter. If $\pi_{\ell g}^{(j,n)} = 1$, both the local filter $w_{j\ell}^{(n)}$ and its normalization state $b_{j\ell}^{(n)}$ are associated with global channel g. Applying different assignments to the filter and its normalization state would cause the matched representation to be normalized using statistics collected for a different local channel. For a non-vacant global position, the normalization state is aggregated over the same matched set defined in Eq. 3:

$$b_g^{(n)} = \frac{1}{\left|\mathcal{S}_g^{(n)}\right|} \sum_{(j,\ell)\in\mathcal{S}_g^{(n)}} b_{j\ell}^{(n)}, \qquad \mathcal{S}_g^{(n)} \neq \emptyset. \tag{11}$$

For a vacant position completed by donor d_n, the corresponding normalization state is taken from the same unmatched donor channel:

$$b_g^{(n)} = b_{d_n,\rho_n(g)}^{(n)}, \qquad g \in \mathcal{V}_n. \tag{12}$$

This preserves the channel-wise association between each matched or donor-completed convolutional filter and the normalization state collected for that channel. The same principle is applied to the normalization state associated with projection shortcuts.

Residual-aware channel alignment. Residual connections add an additional constraint on top of the layer-wise matching described above: because a residual block sums an identity (or projection) branch with a convolutional branch, both branches must be expressed in the same global channel basis at the point where they are added, even though each branch's internal convolutions may still be matched independently. For identity blocks, we enforce this by fixing the output assignment of the branch's final convolution to the incoming stage-level assignment, so the residual sum always combines channels expressed in the same basis. For projection blocks – where the identity path is replaced by a learnable shortcut, typically at a change in spatial resolution or channel width – the shortcut and the residual branch instead share a newly matched output assignment, which then defines the channel basis of the following stage. Client models are reconstructed only at these stage-level boundaries, so no intermediate residual addition ever combines branches expressed in different channel orderings.

Figure 1 illustrates one matching-and-aggregation step of PFEDMA for a single layer, as described in Section 3. Panel (a) shows the per-client neurons at matching layer i being aligned against the fixed-width global layer, with colour indicating which clients contributed a matched neuron; panel (b) shows the resulting training layer $i{+}1$, distinguishing the frozen matched prefix from the layer still being locally trained.

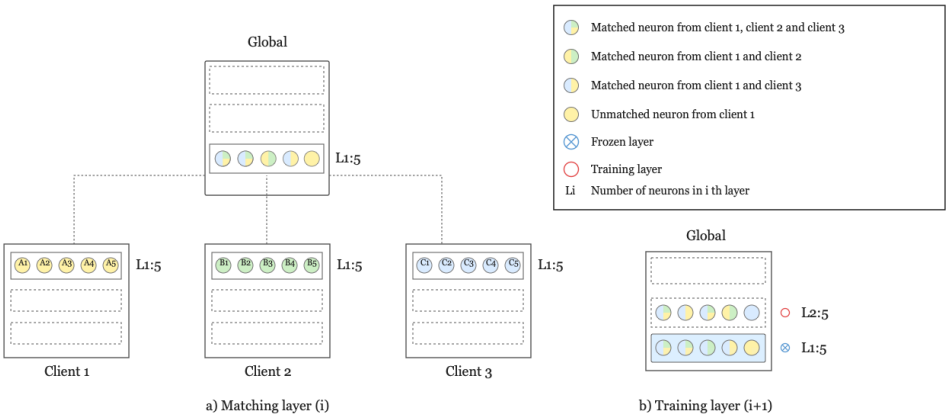


Figure 1: Illustration of layer-wise matching and pruned fixed-width aggregation in PFEDMA. (a) Neurons from clients 1–3 are matched against the global layer at matching layer i. (b) The resulting global model at training layer $i{+}1$, showing the frozen (matched) layer and the layer currently being trained.

For a sequential convolutional network, an assignment applied to the output channels of one layer can be propagated to the input-channel axis of its direct consumer. Residual networks introduce an additional constraint because independently computed branches are combined through element-wise addition. Matching the channels of each branch independently may align the corresponding layers across clients, but does not guarantee that the branches use the same channel ordering at the residual merge.

Identity residual blocks. Consider an identity residual block b in stage s of client j:

$$y_j^{(b)} = x_j^{(s)} + F_j^{(b)}\left(x_j^{(s)}\right), \tag{13}$$

where $x_j^{(s)}$ is the input representation and $F_j^{(b)}$ denotes the two-convolution residual branch. Let $A_j^{(s)}$ denote the local-to-global channel assignment of the stage representation. Since the two branches are added channel-wise, the residual branch must return to the same global channel basis as the identity branch.

The first convolution within the residual branch may still introduce an independently matched hidden representation. Let $H_j^{(b)}$ denote the assignment obtained for its output channels, and let $W_{1,j}^{(b)}$ and $W_{2,j}^{(b)}$ denote the kernels of the first and second convolutions, respectively. Suppressing the spatial kernel dimensions and treating the assignment matrices as channel-reindexing operators, the aligned kernels are

$$\widetilde{W}_{1,j}^{(b)} = {H_j^{(b)}}^\top W_{1,j}^{(b)} A_j^{(s)}, \tag{14}$$

$$\widetilde{W}_{2,j}^{(b)} = {A_j^{(s)}}^\top W_{2,j}^{(b)} H_j^{(b)}. \tag{15}$$

The first convolution therefore consumes the existing stage representation and produces an independently matched hidden basis, whereas the second convolution maps that hidden representation back to the stage basis before the residual addition. The Batch Normalization state following the first convolution uses assignment $H_j^{(b)}$, while the state following the second convolution uses $A_j^{(s)}$.

In particular, the output channels of the second convolution are not assigned through an independent matching operation. If a different assignment $B_j^{(b)}$ were applied to the residual output, the reconstructed block would combine

$${A_j^{(s)}}^\top x_j^{(s)} + {B_j^{(b)}}^\top F_j^{(b)}\left(x_j^{(s)}\right), \tag{16}$$

which does not generally preserve the channel-wise residual pairs learned by the local model. The second convolution is instead aggregated using the existing stage assignment $A_j^{(s)}$ as its output correspondence.

Projection residual blocks. At the beginning of a new stage, the spatial resolution or channel width may change, and the identity branch is replaced by a learnable projection:

$$y_j^{(b)} = P_j^{(b)}\left(x_j^{(s)}\right) + F_j^{(b)}\left(x_j^{(s)}\right), \tag{17}$$

where $P_j^{(b)}$ consists of a projection convolution followed by Batch Normalization. Unlike an identity block, both branches are learnable, allowing the block to establish a new global channel basis. Let $T_j^{(b)}$ denote the independently matched output assignment of the second convolution in the residual branch. The aligned kernels are then

$$\widetilde{W}_{1,j}^{(b)} = {H_j^{(b)}}^\top W_{1,j}^{(b)} A_j^{(s)}, \tag{18}$$

$$\widetilde{W}_{2,j}^{(b)} = {T_j^{(b)}}^\top W_{2,j}^{(b)} H_j^{(b)}, \tag{19}$$

$$\widetilde{W}_{P,j}^{(b)} = {T_j^{(b)}}^\top W_{P,j}^{(b)} A_j^{(s)}, \tag{20}$$

where $W_{P,j}^{(b)}$ is the projection-convolution kernel. The projection consumes the incoming stage basis $A_j^{(s)}$ and produces the same new output basis $T_j^{(b)}$ as the residual branch. Its output assignment is therefore inherited from the second convolution rather than determined through an independent matching operation.

The Batch Normalization state following the first convolution uses $H_j^{(b)}$, whereas the state following the second convolution and the projection Batch Normalization both use $T_j^{(b)}$. After the transition block is aligned, its output assignment defines the channel basis of the subsequent stage:

$$A_j^{(s+1)} = T_j^{(b)}. \tag{21}$$

Consequently, all later identity blocks within that stage consume and return to the same stage-level channel basis.

Following residual alignment, client models are reconstructed only at architecture-consistent boundaries. For sequential networks, this occurs after each matched layer. For residual networks, reconstruction and local adaptation are deferred until all residual merges within the current stage use a common channel basis. The matched convolutional prefix, its associated Batch Normalization state, and any completed projection shortcuts are then frozen together, while the remaining client-specific suffix is locally adapted before matching proceeds. This prevents intermediate client models from evaluating residual additions whose branches are expressed in different channel orderings.

4 EXPERIMENTATION

We evaluate on four image classification benchmarks in a federated learning setting: FMNIST (Xiao et al., 2017), MNIST (LeCun, 1998), USPS (Hull, 1994), and CIFAR-10 (Krizhevsky et al., 2009). Experiments are conducted under IID and Non-IID data partitions, with heterogeneity controlled by the Dirichlet distribution parameter $\alpha \in \{0.05, 0.1, 1\}$, where lower α values indicate higher heterogeneity. For CIFAR-10 we evaluate both a standard CNN and a ResNet-18 backbone, the latter exercising the residual-aware alignment introduced in Section 3.

FMNIST (Xiao et al., 2017) and MNIST (LeCun, 1998) each consist of 70,000 grayscale images from 10 classes, USPS (Hull, 1994) is a smaller handwritten-digit benchmark, and CIFAR-10 (Krizhevsky et al., 2009) contains 60,000 RGB images of objects from 10 categories.

We use J=16 clients, all of which participate in every communication round, and conduct 25 communication rounds for each experiment. For local training, each client trains its model for 5 local epochs using stochastic gradient descent with momentum 0.9 and batch size 64. For FEDMA and PFEDMA, following each matching and model reconstruction step, the model is retrained locally for 5 epochs using SGD with momentum 0.9. The training objective for both local training and retraining is the cross-entropy loss. Learning rates for all methods are selected by grid search, as described below. All experiments were run on a single NVIDIA A100 SXM GPU.

Before the final evaluation, we perform a grid search separately for each dataset, model, and data partition. For all methods, the local learning rate is selected from $\{0.01, 0.05, 0.1\}$. For FEDMA and PFEDMA, the retraining learning rate is also selected from $\{0.01, 0.05, 0.1\}$ using all combinations of local and retraining learning rates. For FEDPROX, the proximal coefficient is selected from $\mu \in \{0.0001, 0.001, 0.01, 0.1, 1.0\}$. The global learning rate is fixed at 1.0 for SCAFFOLD, while $\tau = 0.5$, $\mu = 0.95$, and the sample ratio is fixed at 0.3 for ELASTIC AGGREGATION.

For MNIST and USPS, we use a custom convolutional network that follows a LeNet-style design with two convolutional layers (20 and 50 channels, 5×5 kernels, no padding), each followed by ReLU and max-pooling, and a single-hidden-layer classifier (500 units, dropout $p = 0.1$) preceding the output layer. For CIFAR-10, we use a VGG-style backbone with three convolutional blocks ($32 \to 64$, $64 \to 128 \to 128$, $128 \to 256 \to 256$ channels), each using 3×3 kernels with padding, ReLU activations, and 2×2 max-pooling, with spatial dropout ($p = 0.05$) applied after the second block. The classifier is a three-layer multilayer perceptron ($1024 \to 512 \to$ num_classes) with dropout ($p = 0.1$) applied before the first and final layers.

We additionally evaluate CIFAR-10 using **ResNet-18**, consisting of an initial 7×7 convolutional stem (stride 2) followed by batch normalization, ReLU, and 3×3 max-pooling, succeeded by four stages of residual blocks with channel widths $64, 128, 256, 512$, each stage comprising two basic residual blocks with two 3×3 convolutions and skip connections (using 1×1 convolutional projections for downsampling stages). Global average pooling is applied before a final linear layer mapping to num_classes.

5 RESULTS

Table 1 reports test accuracy across five benchmark configurations: FMNIST, MNIST, USPS, and CIFAR-10 with both a moderate CNN and a ResNet-18 backbone. PFEDMA shows its strongest performance on ResNet-18, where FEDMA cannot be applied directly. It achieves the highest accuracy under all four partitions, improving over the next-best baseline by 8.23, 1.49, 15.00, and 10.10 percentage points under IID, α=0.05, α=0.1, and α=1.0, respectively. Results on the remaining benchmarks are mixed: PFEDMA achieves the highest accuracy on MNIST with α=1.0,

Table 1: Best final-round test accuracy (%) per method and setting, selected over lr $\in \{0.01, 0.05, 0.1\}$ (FedProx additionally $\mu \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$). Diverged runs (NaN) were excluded; "div." = diverged at every learning rate.

Method	Partition	FMNIST	MNIST	USPS	CIFAR-10 (CNN)	CIFAR-10 (ResNet-18)
FedAvg	IID	91.38	99.04	96.98	72.55	76.71
	α=0.05	84.09	97.71	91.50	67.78	61.70
	α=0.1	87.98	98.73	91.58	67.58	59.21
	α=1.0	**91.19**	99.14	96.68	61.96	75.36
FedProx	IID	91.55	99.05	97.07	73.09	76.37
	α=0.05	**84.62**	98.08	**91.59**	**68.06**	61.19
	α=0.1	**88.19**	**98.78**	80.71	67.54	60.37
	α=1.0	91.16	99.15	96.81	70.17	75.12
SCAFFOLD	IID	**91.92**	99.18	**97.67**	78.31	80.79
	α=0.05	24.20	div.	83.61	div.	35.71
	α=0.1	87.66	98.58	90.25	63.83	58.73
	α=1.0	91.37	99.22	**97.24**	63.27	75.98
Elastic	IID	91.20	98.99	96.42	66.88	73.91
	α=0.05	80.07	**98.27**	90.68	67.89	61.11
	α=0.1	87.90	98.66	90.20	64.77	56.75
	α=1.0	91.16	99.06	96.55	69.79	74.83
FedMA	IID	90.76	**99.31**	94.72	**83.95**	–
	α=0.05	51.15	89.45	90.48	65.21	–
	α=0.1	83.06	96.91	**91.63**	**77.15**	–
	α=1.0	90.36	99.29	94.77	**81.59**	–
PFedMA (ours)	IID	91.11	99.27	95.47	81.81	**89.02**
	α=0.05	66.38	93.96	90.08	57.26	**63.19**
	α=0.1	82.58	97.00	90.53	69.05	**75.37**
	α=1.0	90.38	**99.30**	95.52	73.49	**86.08**

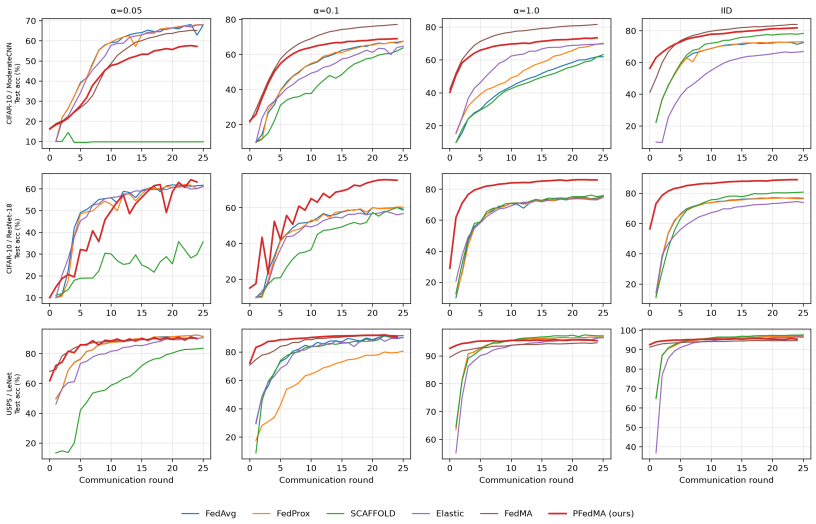


Figure 2: Test accuracy vs. communication round on USPS and CIFAR-10 (moderate CNN and ResNet-18 backbones) under an IID partition and non-IID Dirichlet partitions ($\alpha \in \{0.1, 1.0\}$), comparing PFEDMA to all baselines.

but trails FEDMA on CIFAR-10 with the moderate CNN. This reflects a trade-off between the fixed global width enforced by PFEDMA and the additional capacity obtained by allowing FEDMA to grow the global architecture. Figure 2 shows the corresponding accuracy curves for USPS and both CIFAR-10 backbones.

Figure 3 reports test accuracy vs. communication round on FMNIST and MNIST under an IID partition and non-IID Dirichlet partitions ($\alpha \in \{0.1, 1.0\}$), complementing the USPS and CIFAR-10 curves reported in Figure 2 and the full per-benchmark breakdown in Table 1.

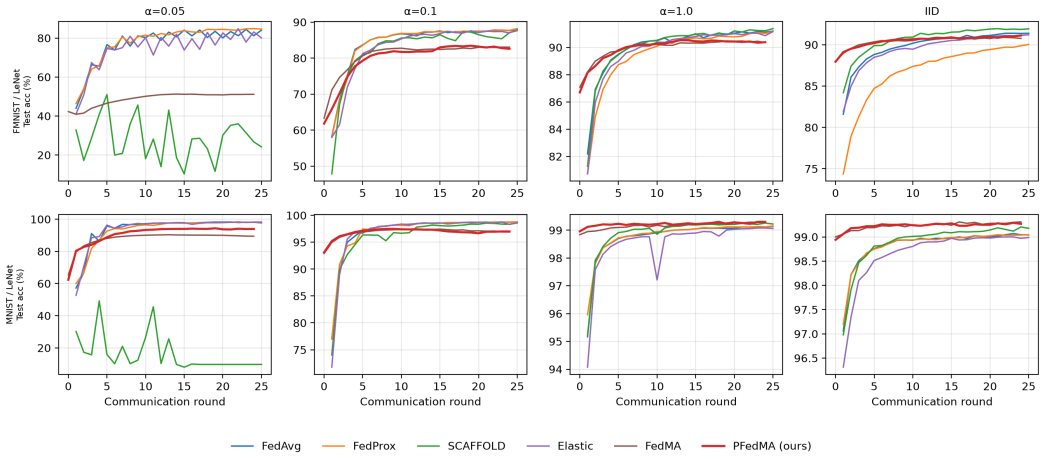


Figure 3: Test accuracy vs. communication round on FMNIST and MNIST under an IID partition and non-IID Dirichlet partitions ($\alpha \in \{0.1, 1.0\}$), comparing PFEDMA to all baselines.

6 CONCLUSION

PFEDMA introduces an architecture-stable neuron matching-based aggregation strategy that preserves representational alignment while preventing the model growth associated with FEDMA. By pruning unmatched non-donor neurons and using a donor-client mechanism to complete vacant global positions, PFEDMA maintains a fixed model architecture throughout training. We further extend the matched aggregation to residual architectures through consistent Batch Normalisation, alignment, and residual-channel assignments, enabling permutation-aware aggregation of RESNET-based models. Empirically, PFEDMA provides consistent gains on ResNet-18 across IID and non-IID partitions, while its performance on the The remaining datasets and architectures are mixed. A remaining limitation is the use of random donor selection, which may discard useful unmatched representations or introduce donor-specific bias under severe client heterogeneity. Developing similarity- or importance-aware donor selection strategies therefore, represent a promising direction for future work.

AI USE STATEMENT

In this work, we used generative AI tools for editing, formatting, and LaTeX preparation of the manuscript, for feedback on the clarity and internal consistency of the manuscript, and for drafting revised wording of portions of the text (including parts of the Results and Related Work sections) based on our own results, method, and experimental setup. Generative AI tools were not used to develop the PFEDMA method, design or run the experiments, or produce the reported results. All AI-assisted text was reviewed, edited, and verified by the authors against our implementation and experimental logs, and cited works suggested during revision were checked against the original papers. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N. Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. *CoRR*, abs/2111.04263, 2021. URL https://arxiv.org/abs/2111.04263.

Dengsheng Chen, Jie Hu, Vince Junkai Tan, Xiaoming Wei, and Enhua Wu. Elastic aggregation for federated optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12187–12197, 2023.

J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994. doi: 10.1109/34.291440.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pp. 5132–5143. PMLR, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

Jie Wen, Zhixia Zhang, Yang Lan, Zhihua Cui, Jianghui Cai, and Wensheng Zhang. A survey on federated learning: challenges and applications. *International journal of machine learning and cybernetics*, 14(2):513–535, 2023.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. In *International conference on machine learning*, pp. 7252–7261. PMLR, 2019.