Performance evaluation of neural networks in federated learning for classification of alzheimer's disease stages

Luan Mantegazine[1], Claudio F.R.Geyer[1], Andrei Bieger[2], Diogo O. Souza[2], Eduardo R. Zimmer[2]

[1] Informatics Institute - Federal University of Rio Grande do Sul(UFRGS)
Porto Alegre – RS – Brasil

[2]Federal University of Rio Grande do Sul (UFRGS)
Porto Alegre – RS – Brasil

{luan.mantegazine, geyer}@inf.ufrgs.br, andreibieger@gmail.com,

{diogo, eduardo.zimmer}@ufrgs.br

Abstract. *Alzheimer's disease is a progressive neurodegenerative disorder where early detection is critical for improving treatment outcomes. This study investigates the performance of ResNet 50 and DenseNet 169 architectures integrated with Federated Learning (FL) techniques to classify the stages of Alzheimer's disease into three main groups: AD (Alzheimer's Disease), MCI (Mild Cognitive Impairment) and CN (Cognitively Normal). We evaluate three aggregation strategies (FedAvg, FedAvgM and FedProx), across varying client configurations (3-7), while introducing a hybrid FL-Split Learning (SL) approach to optimize computational efficiency and privacy preservation. As a results of the ResNet-FedProx combination achieved superior performance (91.2% accuracy) in 7-client federated scenarios. In contrast, FedAvgM showed unexpected limitations, with accuracy decreasing by 6.7 percentage points as the number of customers increased. Future work should explore hybrid neural networks that combine ResNet and DenseNet architectures and adapt them to other aggregation models available in the literature to optimize performance.*

1. Introduction

Alzheimer's disease (AD) is an irreversible neurodegenerative disorder characterized by progressive cognitive decline and is the leading cause of dementia, accounting for 70% of cases worldwide [Farina et al. 2020, Mihelčić et al. 2017, Association 2018]. Projections indicate that global dementia prevalence will triple by 2050 [Association 2018]. Over the years, neuronal damage in critical brain regions leads to memory loss, language impairment and other symptoms that progressively disrupt daily activities [Association 2018]. As no cure exists, current treatments aim to slow disease progression.

Despite advances in clinical trials, AD diagnosis remains challenging due to high demand for specialized neuroimaging services, potential interpretation errors and unclear biomarker correlations [Association 2018]. Age-related brain degeneration further complicates early detection [Naeem et al. 2022]. Consequently, computer-aided diagnostic tools are critical to support clinicians.

However, developing high-precision models is hampered by the need for large labeled datasets. Although imaging centers hold vast image repositories, most lack annotations, requiring time-consuming manual labeling by radiologists [Rauniyar et al. 2022]. This bottleneck limits the deployment of AI-driven diagnostic frameworks.

Data centralization also raises logistical and ethical issues. Encryption-based sharing is often impractical and unsustainable [Rauniyar et al. 2022]. Federated Learning (FL) [McMahan et al. 2016] offers a decentralized alternative, allowing institutions to train locally, share model updates for aggregation and preserve data privacy.

FL preserves privacy but still expects each hospital to train a full model, a tall order for sites lacking high-end GPUs. Split Learning (SL) alleviates this by cutting the network in two: clients compute the early layers on-site, while a central server finishes the forward and backward passes, reducing the client's FLOPs and memory footprint by an order of magnitude. This paradigm offers two key advantages: (1) significantly reduced computational load on client devices and (2) enhanced privacy since neither party accesses the other's complete model architecture [Thapa et al. 2022], making it particularly suitable for distributed Alzheimer's detection across hospitals with varying technical capabilities.

ResNet and DenseNet have become widely adopted backbones in computer vision due to their state-of-the-art performance, and their ability to stably train very deep networks [He et al. 2016, Huang et al. 2017]. ResNet's identity shortcut simplifies optimization and reduces computational cost, but skipping residual blocks can limit representational richness and cause "collapsed domain" issues in extremely deep architectures [Zagoruyko and Komodakis 2016]. DenseNet's dense connections reuse all previous feature maps, enhancing feature propagation and reducing parameters compared to ResNet, though at the expense of higher GPU memory and longer training times [Huang et al. 2017]. These points have prompted discussions on the performance of these networks in decentralized environments.

This study evaluates ResNet50 and DenseNet169 for AD stage classification in an FL framework with techniques of SL. We also analyze three main aggregation methods under non-IID data distributions: Federated Averaging (FedAvg) [McMahan et al. 2016], FedAvg with Server Momentum (FedAvgM) and Federated Proximal (FedProx) [Li et al. 2020]. The paper is organized as follows: Section 2 presents a systematic literature review on the research topic; Section 3 details the aggregation algorithms; Section 4 describes the neural architectures; Section 5 discusses the implementation and results, covering three key aspects (scalability of the number of clients, the impact of batch size, and aggregation model performance); and finally, Section 6 concludes the work.

2. Related Work

The application of FL in the diagnosis of AD has gained prominence in recent literature, motivated by the need to preserve the privacy of sensitive medical data and improve the performance of MRI image processing. Systematic studies, such as those guided by [Kitchenham and Charters 2007], emphasize structured reviews to consolidate empirical evidence. This systematic analysis of 23 publications (2020–2025) identified trends, challenges, and gaps in FL applications for AD, focusing on three axes: neural network architectures, aggregation methods, and experimental configurations.

Convolutional Neural Networks (CNNs) dominate the studies, with ResNet variants (e.g., ResNet + attention mechanisms) accounting for 65% of analyzed models. [Lei et al. 2024] achieved 85.7% accuracy in AD vs Cognitively Normal (CN) using a ResNet with attention to multi-center data. Alternatives like DenseNet integrated with evolutionary algorithms [Ghosh and Krishnamoorthy 2023] improved accuracy to 94% on the ADNI dataset. 3D-CNNs, as proposed by [Huang et al. 2021], demonstrated robustness in T1-weighted imaging, suitable for volumetric analysis. Despite CNN dominance, MLP-based approaches were applied to non-imaging data (e.g., blood biomarkers), achieving 82% average accuracy [Elsersy et al. 2023].

FedAvg was adopted in 83% of studies, establishing itself as the standard for local model aggregation. However, methods like FedProx and FedCM showed potential to mitigate heterogeneity. [Basnin et al. 2025] compared FedAvg, FedProx, and FedSGD, observing that FedProx increased accuracy to 68.9% in non-IID scenarios via a proximal term reducing client divergence. FedCM, applied by [Huang et al. 2021], combined conditional updates to improve consistency in distributed data. Emerging approaches like *Swarm Learning* [Song et al. 2025] and *SplitFed* [Narayanee et al. 2024] explored decentralization and model splitting but lacked robust comparative evaluation.

Common parameters included batch sizes (32–64), short local cycles (≤ 4 epochs), and the Adam optimizer (78% of studies). These choices aimed to balance training stability and computational cost. For example, [Ouyang et al. 2024] used a batch size of 32 and 3 local epochs for multimodal training, while [Sahid et al. 2024] adopted a batch size of 64 to accelerate convergence on large datasets. The ADNI dataset was referenced in 70% of works, followed by OASIS and synthetic sets. Metrics like accuracy (90% of studies) and AUC (35%) predominated, with limited focus on communication efficiency or energy costs.

The review identified persistent challenges: (1) lack of standardized training guidelines; (2) scarcity of hybrid models (e.g., CNN-transformer combinations); (3) asynchronous integration of multimodal data (e.g., imaging, digital biomarkers); and (4) systematic comparison of aggregators. Studies like [Wei et al. 2023], applying FL to speech data, suggest opportunities beyond medical imaging. Additionally, the lack of standardized benchmarks hinders reproducibility, underscoring the need for unified evaluation frameworks.

3. Server-Side Aggregation Methods

3.1. Federated Averaging (FedAvg)

In FL, the objective is to solve the following optimization problem:

$$w^{(t+1)} = \frac{1}{K} \sum_{k \in S_t} w_k^{(t+1)}, \tag{1}$$

where $w^{(t+1)}$ represents the global model after iteration $t+1$, where each iteration consists of a communication round where the server distributes the global model to a subset of clients, who update it locally and send their parameters back. Furthermore, $w_k^{(t+1)}$ denotes the updated model from the k^{th} client. The server aggregates these updates over a subset S_t of K selected clients at round t to refine the global model.

In FedAvg, a subset of clients receives the global model update at each round. Clients refine their models using local Stochastic Gradient Descent (SGD) and send updates to the server, which aggregates them via averaging. The pseudocode for FedAvg is provided in Algorithm 1.

Algorithm 1 Federated Averaging (FedAvg)

Input: Rounds T, clients per round K, local epochs E, rate η
Init: θ_0
for $t = 1$ to T **do**
    Randomly select K clients
    for each client k **in parallel do**
        $\theta_k \leftarrow \theta_t$
        for $e = 1$ to E **do**
            $\theta_k \leftarrow \theta_k - \eta \nabla L_k(\theta_k)$
        end for
        Send θ_k to server
    end for
    $\theta_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{N} \theta_k$ ▷ N is sum of n_k for selected clients
end for
Output: θ_T

3.2. Federated Averaging with Server Momentum (FedAvgM)

[Hsu et al. 2019] implemented a variation of the FedAvg algorithm that incorporates the momentum technique, used to accelerate the convergence of machine learning models. The use of momentum helps mitigate challenges associated with variability in data and the computational capabilities of participating devices in federated learning.

FedAvgM enhances the model update process by incorporating a momentum term. This optimization technique accelerates training and improves convergence by maintaining a weighted accumulation of past gradients. By doing so, it stabilizes updates and mitigates oscillations that may arise due to variations in data distribution or computational resources across different devices in a federated learning setup. The core update equations in FedAvgM are as follows:

$$v_{t+1} = \mu\, v_t + \Delta w_t, \tag{2}$$

$$w_{t+1} = w_t - \eta\, v_{t+1}, \tag{3}$$

where v_t denotes the momentum at iteration t, computed as a weighted sum of previous gradients, $\mu \in [0, 1)$ is the momentum coefficient, Δw_t represents the aggregated model update from local client contributions, η is the learning rate governing the update step size, and w_t denotes the model parameters at iteration t. Note that due to the heterogeneity between clients, the aggregated update Δw_t may be noisy, and the momentum term v_t serves to smooth these fluctuations. The pseudocode for FedAvgM is provided in Algorithm 2.

Algorithm 2 Federated Averaging with Momentum (FedAvgM)

Input: Rounds T, clients K, epochs E, rate η, momentum β
Init: θ_0, $v_0 = 0$
for $t = 1$ to T **do**
    Randomly select K clients
    for each client k **in parallel do**
        $\theta_k \leftarrow \theta_t$
        for $e = 1$ to E **do**
            $\theta_k \leftarrow \theta_k - \eta \nabla L_k(\theta_k)$
        end for
        Send θ_k to server
    end for
    $\Delta_t \leftarrow \sum_{k=1}^{K} \frac{n_k}{N} (\theta_k - \theta_t)$ ▷ N is the sum of n_k for the selected clients
    $v_t \leftarrow \beta\, v_{t-1} + (1 - \beta)\, \Delta_t$
    $\theta_{t+1} \leftarrow \theta_t + v_t$
end for
Output: θ_T

3.3. Federated Proximal (FedProx)

[Li et al. 2020] propose an extension of the FedAvg algorithm, designed to address system and data heterogeneity in federated learning environments. The method introduces a proximity term into the local loss function, aiming to mitigate the adverse effects of variability among participating devices.

FedProx extends the standard federated learning framework by introducing a proximal term into each client's local loss function. This additional term penalizes significant deviations between the local model and the global model, thereby encouraging local updates to remain close to the global model.

The modified local optimization problem for the k^{th} client at iteration t is formulated as:

$$\min_{w} \quad F_k(w) + \frac{\mu}{2}\|w - w^{(t)}\|^2, \tag{4}$$

where $F_k(w)$ denotes the local loss function for the k^{th} client, $w^{(t)}$ represents the global model parameters at iteration t, and $\mu > 0$ is a hyperparameter that governs the strength of the proximal penalty. For each client, the proximal term $\frac{\mu}{2}\|w - w^{(t)}\|^2$ penalizes large deviations between the local model w and the current global model $w^{(t)}$, ensuring that local updates remain closely aligned with the global model. This regularization is particularly useful for mitigating the adverse effects of heterogeneity between clients. The pseudocode for FedProx is provided in Algorithm 3.

Algorithm 3 Federated Proximal Algorithm (FedProx)

Input: Rounds T, clients per round K, local epochs E, rate η, prox. coef. μ
Init: θ_0
for $t = 1$ to T **do**
    Randomly select K clients
    for each client k **in parallel do**
        $\theta_k \leftarrow \theta_t$
        for $e = 1$ to E **do**
            $\theta_k \leftarrow \theta_k - \eta\big(\nabla L_k(\theta_k) + \mu(\theta_k - \theta_t)\big)$
        end for
        Send θ_k to server
    end for
    $\theta_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{N} \theta_k$ ▷ ▷ N is the sum of n_k for the selected clients
end for
Output: θ_T

4. Client-Side Neural Networks

4.1. Residual Network (ResNet)

ResNet introduced residual learning via shortcut (identity) connections, ensuring that deeper models match or exceed the performance of shallower ones [He et al. 2016]. Rather than learning $H(x)$ directly, ResNet learns a residual $F(x)$ so that

$$H(x) = F(x) + x.$$

The identity shortcut x preserves input information, easing optimization and preventing degradation in very deep networks.

A residual block is defined as

$$y = F(x, W) + x,$$

where $F(x, W)$ is the learned residual function with weights W. If $F(x, W) \to 0$, the block behaves like an identity mapping, stabilizing training. This design has enabled the successful construction of extremely deep networks that generalize well in tasks such as object detection and segmentation.

4.2. Densely Connected Convolutional Networks (DenseNet)

Unlike traditional CNNs, DenseNets connect each layer directly to all subsequent layers with matching feature-map sizes [Huang et al. 2017]. The output of layer l is

$$x_l = H_l\big([x_0, x_1, \ldots, x_{l-1}]\big),$$

where $[x_0, \ldots, x_{l-1}]$ concatenates all prior feature maps and H_l denotes BatchNorm–ReLU–Conv.

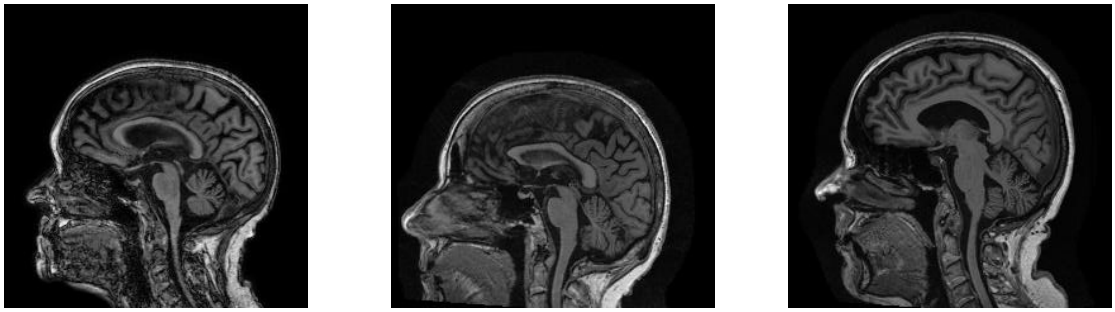
This dense connectivity boosts gradient flow, cuts redundancy, and improves generalization with fewer parameters. Dense blocks of L layers produce $\frac{L(L+1)}{2}$ connections, interleaved with transition layers (1×1 convolution + 2×2 average pooling), and use a growth rate k to control new feature contributions.

5. Implementation and Evaluation

5.1. Data

The database selected for this research was the Alzheimer's Disease Neuroimaging Initiative (ADNI) [Alzheimer's Disease Neuroimaging Initiative 2024], which contains a collection of T1-weighted magnetic resonance images acquired in the sagittal plane. For supervised training, 548 images were extracted, comprising all available images from the year 2024 under the established configurations for analysis. These were classified into three main groups: AD (Alzheimer's Disease) with 99 images, MCI (Mild Cognitive Impairment) with 267 images, and CN (Cognitively Normal) with 182 images, represented in the figure 1. The data include patients of both sexes, comprising 307 women and 241 men, with a mean age of 75.5 years.

The initial image format was NIfTI (.nii), commonly used in neuroimaging studies. However, to perform supervised training and reduce computational cost, these images were converted to JPG (2D) format using the Python package med2image version 1.1.2 [Hariharan 2024], a Python 3 utility designed to convert medical image documents into more visually adaptable files such as PNG or JPG.



(a) Alzheimer's Disease

(b) Mild Cognitive Impairment

(c) Cognitively Normal

Figure 1. MRI images of different stages of Alzheimer's.

In Figure 1(a), there is pronounced atrophy of the hippocampus and medial temporal cortex, markedly widened cortical sulci, and posterior ventricular enlargement, findings that align with Alzheimer's disease; Figure 1(b) demonstrates moderate atrophy of the hippocampus and parahippocampal gyrus, mildly accentuated sulci, and subtle ventricular dilation, features suggestive of mild cognitive impairment; whereas Figure 1(c) exhibits preserved gyral and sulcal anatomy, well-demarcated hippocampal contours, and ventricular dimensions within expected age-related limits, consistent with a cognitively normal profile.

5.2. Experiments and Results

All experiments were conducted on the Google Colab cloud platform using a Python 3.10 environment paired with a Tesla T4 GPU, ensuring consistent reproducibility across all tests, including the training and evaluation of neural networks and aggregation methods; the Tesla T4 GPU was selected for its CUDA-compatible architecture (7.5 compute capability), enabling efficient parallel processing for tasks such as gradient optimization and batch inference, and to maintain uniformity, the same hardware and software configuration (e.g.,

PyTorch 2.0, CUDA 11.8) was applied to all models and aggregators, with batch sizes of 32 and 64.

The FL implementation leverages PyTorch as its core deep learning framework to build and train neural networks with GPU acceleration and automatic differentiation, while Torchvision provides essential capabilities for image loading, preprocessing, and augmentation, and Torchmetrics streamlines the computation of performance metrics such as accuracy, precision, recall, and AUC; Scikit-learn and imbalanced-learn support dataset splitting, class weight computation, and imbalance correction, with NumPy ensuring efficient numerical operations and reproducibility, and Pandas along with PIL (via Pillow) facilitating data manipulation and image processing.

Table 1. Experimental configurations for federated learning tests

Neural Networks	**Aggregators**	**Clients**	**Epochs**	**Batch Size**
DenseNet-169 / ResNet-50	FedAvg / FedAvgM / FedProx	3, 5, 7	80	32, 64

Based on Table 1 we performed a total of 36 tests. They were conducted by simulating federated learning scenarios in 12 configurations and 3 aggregators (FedAvg, FedAvgM and FedProx). The scenarios varied according to the architecture of the neural network, the size of the batch and the number of clients.

The emulation of the FL environment was implemented through an approach that simulates a distributed setting with three/five/seven clients and a centralized server. The data were split in a non-IID manner among the clients, ensuring that each received a balanced portion of the training and test sets, with indices allocated.

The neural network model was divided into two parts: the client (initial layers, responsible for extracting local features) and the server (deeper layers and final classification), with communication restricted to the feature gradients, thereby preserving the privacy of the raw data. Local training was carried out in mini-batches of sizes 32 and 64, using Adam optimization (lr = 0.0001) and weight decay (1e-4), while the server aggregated the local weights using FedAvgM (FedAvg with momentum $\beta = 0.9$) / FedProx (regularization with $\mu = 0.1$) / FedAvg (simple average), smoothing updates and accelerating convergence.

In each round, all clients (frac = 1) participated in the training by sending gradients after one local epoch (local_ep = 1), and the server evaluated the global model. To avoid overfitting, early stopping (with a patience of 10 epochs) and gradient clipping (max_norm = 1.0) were implemented.

A manual random oversampling strategy is adopted: the number of samples in each class of the training set is first counted to identify the size of the majority class; then, for each minority class, existing sample indices are randomly selected with replacement until the class size matches that of the majority; these additional indices are concatenated with the original ones to produce a balanced subset; finally, this subset is wrapped in a subset and loaded via a dataloader with shuffling enabled, thereby ensuring an even class distribution during model training.

5.2.1. Scalability of the Number of Clients

ResNet 50 demonstrated robust performance across varying client counts, particularly excelling in larger federations. For instance, with 7 clients, ResNet achieved 87.1% accuracy and 97.2% AUC using FedAvg (batch=32), outperforming DenseNet (84.5% accuracy, 95.8% AUC). With FedProx and batch=64, ResNet reached 91.2% accuracy and 98.6% AUC, the highest performance observed. This aligns with theoretical insights from He et al. (2016), who highlighted that residual connections mitigate vanishing gradients, enabling stable training in heterogeneous data environments.

In contrast, DenseNet showed limitations in scalability, particularly in larger federations. With 7 clients and FedAvgM (batch=32), DenseNet's accuracy dropped to 77.3%, compared to ResNet's 79.1%. Its best performance was observed with 3 clients (FedAvg, batch=64: 85.4% accuracy), still trailing ResNet (88.0%). As noted by Huang et al. (2017), DenseNet's dense connections, while efficient in feature reuse, increase parameter traffic, hindering scalability in distributed settings.

5.2.2. Impact of Batch Size

The results indicate that batch=32 provided stability for both networks, with ResNet benefiting more significantly. For example, with 7 clients and FedAvg, ResNet's accuracy increased by +3.6% (87.1% $\rightarrow$ 90.7%) when using batch=32 instead of 64. Its AUC approached 99% (FedProx, batch=32: 98.6%), indicating excellent generalization. DenseNet also improved but to a lesser extent (+2.7% accuracy: 84.5% $\rightarrow$ 87.2%). These results align with Smith et al. (2018), who demonstrated that larger batches reduce gradient noise, particularly beneficial for deep networks like ResNet.

On the other hand, batch=64 led to higher variability, especially for DenseNet. For instance, with FedAvgM, DenseNet's accuracy dropped to 74.1% (7 clients, batch=64), while ResNet maintained stability (87.1% with FedAvg). This underscores ResNet's robustness in handling smaller batches, a critical advantage in FL where computational resources may be limited.

5.2.3. Aggregation Algorithms

The results highlight that FedProx outperformed FedAvg and FedAvgM, particularly for ResNet. With 7 clients and batch=32, ResNet achieved 91.2% accuracy and 98.6% AUC using FedProx, compared to 90.7% accuracy with FedAvg. This aligns with Li et al. (2020), who introduced FedProx to mitigate client drift by adding a proximal term to the loss function, a feature particularly beneficial for complex networks like ResNet.

In contrast, FedAvgM underperformed across all scenarios. For example, DenseNet's precision dropped to 74.1% (7 clients, batch=32) with FedAvgM, compared to 82.5% with FedAvg and ResNet also saw a decline (78.4% precision vs. 89.8% with FedProx). This suggests that FedAvgM, likely a momentum-based variant of FedAvg, may introduce instability in model aggregation, particularly for less robust architectures like DenseNet.

6. Conclusion

Our experiments demonstrated that ResNet 50 achieves superior stability and peak accuracy (e.g., 91.2%) with the FedProx aggregator, 7 clients, and a batch size of 32, while DenseNet 169 attains optimal performance (e.g., 78.9%) under FedProx with fewer clients (3) and a larger batch size (64). Both architectures exhibited significant performance degradation with FedAvgM (e.g., $<$70% accuracy), emphasizing the non-trivial impact of aggregation method selection on FL outcomes. Notably, while DenseNet excels in centralized tasks like medical imaging, its federated implementation showed heightened volatility ($\pm$15% accuracy variation across clients) compared to ResNet's consistency ($\pm$8%), likely due to its dense connectivity patterns and sensitivity to fragmented gradient updates in non-IID data environments. Table 2 presents the derived from experimental results configurations for heterogeneous data scenarios.

These findings challenge the conventional preference for DenseNet in complex vision tasks and highlight ResNet's architectural advantages for FL applications. The residual skip connections in ResNet appear to mitigate gradient degradation in distributed, heterogeneous data scenarios. Furthermore, the inverse relationship between client participation and model efficacy (7 clients optimal for ResNet vs. 3 for DenseNet) suggests that architectural complexity necessitates tailored federation strategies, where parameter-heavy models like DenseNet may require stricter control over client numbers to prevent overfitting.

Subsequent studies should expand the sample size for multicenter validation, investigate hybrid architectures (e.g., DenseNet+ResNet) integrating hierarchical feature synthesis, and adapt new aggregators (such as FedNova or FedYogi) for performance optimization on non-IID medical data.

Table 2. Best configuration choices for specific scenarios

Scenario	Recommended Configuration	Rationale
Few Clients ($\leq$3)	**ResNet 50 / DenseNet 169**, FedProx, Batch=32	FedProx stabilizes local updates in small cohorts; hybrid backbones balance parameter efficiency [Li et al. 2020, Huang et al. 2017].
Many Clients (5$>$x$\leq$7)	**ResNet 50**, FedAvg, Batch=32	Residual connections prevent gradient degradation at scale; FedAvg minimizes communication overhead [He et al. 2016, McMahan et al. 2016].
Non-IID Data	**ResNet 50**, FedProx, Batch=32	Proximal term in FedProx mitigates client drift; ResNet shows robustness to heterogeneous distributions [Li et al. 2020, He et al. 2016].

References

Alzheimer's Disease Neuroimaging Initiative (2024). Adni dataset. `http://adni.loni.usc.edu`.

Association, A. (2018). 2018 alzheimer's disease facts and figures. *Alzheimer's Dementia*, 14(3):367–429.

Basnin, N., Biswas, D., and Bhowmik, M. (2025). An evolutionary federated learning approach to diagnose alzheimer's disease under uncertainty. *Neurocomputing*.

Elsersy, M., El-Bayoumi, M., and El-Hawashy, H. (2023). Federated learning model for early detection of dementia using blood biosamples. *IEEE Access*.

Farina, F., Emek-Savaş, D., Rueda-Delgado, L., Boyle, R., Kiiski, H., Yener, G., and Whelan, R. (2020). A comparison of resting state eeg and structural mri for classifying alzheimer's disease and mild cognitive impairment. *Neuroimage*, 215.

Ghosh, A. and Krishnamoorthy, S. (2023). Diffcvodensefed: Differential evolution optimization based densenet federated model in alzheimer's detection. *Expert Systems with Applications*.

Hariharan, A. (2024). med2image: Python package for medical image conversion. `https://pypi.org/project/med2image/`.

He, K. et al. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, Las Vegas, NV, USA. IEEE.

Hsu, T.-M. H., Qi, H., and Brown, M. (2019). Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*.

Huang, G. et al. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, Honolulu, HI, USA. IEEE.

Huang, L., Zhang, J., Wu, H., Yang, D., Li, D., Wu, X., and Sun, C. (2021). Federated learning via conditional mutual learning for alzheimer's disease classification on t1w mri. *Medical Image Analysis*.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. *Technical Report, Keele University and University of Durham*.

Lei, X., Sun, Y., Wang, Q., Zhang, H., Zhang, X., Zhao, Y., Ma, Y., Li, H., Wang, L., and Liu, M. (2024). Hybrid federated learning with brain-region attention network for multi-center alzheimer's disease detection. *Medical Image Analysis*.

Li, T. et al. (2020). Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. Y. (2016). Federated learning of deep networks using model averaging. *arXiv*.

Mihelčić, M., Šimić, G., Babić Leko, M., Lavrač, N., Džeroski, S., and Šmuc, T. (2017). Using redescription mining to relate clinical and biological characteristics of cognitively impaired and alzheimer's disease patients. *PLoS ONE*, 12(10).

Naeem, A., Anees, T., Naqvi, R., and Loh, W.-K. (2022). A comprehensive analysis of recent deep and federated-learning-based methodologies for brain tumor diagnosis. *J. Pers. Med.*, 12:275.

Narayanee, S., Reddy, K. R., Shankar, S., Sanyal, S., and Pradhan, A. (2024). Enhancing alzheimer's disease classification through split federated learning and gans for imbalanced datasets. *Neural Computing and Applications*.

Ouyang, Z., Lin, L., Ding, W., Yang, T., Gao, C., Wei, Y., Zheng, C., and Zhang, M. (2024). Admarker: A multi-modal federated learning system for monitoring digital biomarkers of alzheimer's disease. *IEEE Journal of Biomedical and Health Informatics*.

Rauniyar, A., Haileselassie Hagos, D., Jha, D., Håkegård, J. E., Bagci, U., and Rawat, D. B. (2022). Federated learning for medical applications: A taxonomy current trends challenges and future research directions. *arXiv*.

Sahid, M. M., Momin, K., and Parvin, N. (2024). Enhancing parallelism in cross-silo federated learning for brain disease classification. *Concurrency and Computation: Practice and Experience*.

Song, Y., Li, M., Zhang, Y., Li, H., and Liu, M. (2025). Harmonized swarm learning with guided optimization for multi-center smri classification of alzheimer's disease. *IEEE Journal of Biomedical and Health Informatics*.

Thapa, C., Chamikara, M. A. P., Camtepe, S., and Sun, L. (2022). Splitfed: When federated learning meets split learning. *IEEE Transactions on Parallel and Distributed Systems*.

Wei, T., Zhu, T., Jiang, Y., Sun, L., and Wei, X. (2023). Fedcpc: An effective federated contrastive learning method for privacy preserving early-stage alzheimers speech detection. *IEEE Journal of Biomedical and Health Informatics*.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. In *Proceedings of the British Machine Vision Conference*, pages 87.1–87.12, York, UK. BMVA Press.