UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

LUAN MANTEGAZINE

Avaliação de desempenho preditivo de redes neurais em aprendizagem federada para classificação de estágios da doença de Alzheimer

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Orientador: Prof. Dr. Cláudio Geyer

Porto Alegre
2026

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Mantegazine, Luan

Avaliação de desempenho preditivo de redes neurais em aprendizagem federada para classificação de estágios da doença de Alzheimer / Luan Mantegazine. – Porto Alegre: PPGC da UFRGS, 2026.

122 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2026. Orientador: Cláudio Geyer.

1. Doença de Alzheimer. 2. Aprendizado Federado. 3. Aprendizado Profundo. 4. Aprendizado Dividido. 5. Processamento de Imagem. I. Geyer, Cláudio. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Prof[a]. Marcia Cristina Bernardes Barbosa
Vice-Reitor: Prof. Pedro de Almeida Costa
Pró-Reitora de Pós-Graduação: Prof[a]. Claudia Wasserman
Diretor do Instituto de Informática: Prof. Luciano Paschoal Gaspary
Coordenador do PPGC: Prof. Gabriel Luca Nazar
Bibliotecário-chefe do Instituto de Informática: Alexsander Borges Ribeiro

"Everyone knows someone who has survived cancer,
but until now no one knows anyone who has survived Alzheimer's Disease."
— Dale E. Bredesen

AGRADECIMENTOS

À minha família, pelo amor incondicional, pela paciência nas ausências e pela confiança que sempre depositaram em mim. Vocês foram a base que sustentou cada etapa desta trajetória e o porto seguro nos dias desafiadores. Tudo o que realizei aqui nasce do exemplo e do apoio de vocês.

Ao meu orientador, Prof. Dr. Cláudio Geyer, pela orientação firme e generosa, pelo rigor científico aliado à liberdade. Agradeço a disponibilidade constante, a confiança e o compromisso com a formação que ultrapassa as páginas desta dissertação.

Por fim, aos meus amigos, pela torcida, pela leveza nos momentos difíceis e pela companhia que torna o caminho mais humano. Obrigado pelas mensagens de incentivo e por celebrarem cada pequena conquista ao meu lado.

A todas e todos que, de alguma forma, contribuíram para que este trabalho se tornasse possível, deixo meu sincero muito obrigado.

RESUMO

A Doença de Alzheimer (AD) é um distúrbio neurodegenerativo progressivo para o qual a detecção precoce é essencial para retardar a progressão da doença e melhorar a qualidade de vida dos pacientes. A prevalência global deve triplicar até 2050. Este estudo investiga a aplicação de Aprendizado Federado (FL) combinado com Aprendizado Dividido (SL) e Redes Neurais Convolucionais (CNNs) para classificar indivíduos em três estágios: AD, Comprometimento Cognitivo Leve (MCI) e Cognitivamente Normal (CN). Um aspecto crítico da implantação de FL é a análise cuidadosa dos hiperparâmetros e a seleção da arquitetura neural ideal. Avaliamos duas arquiteturas (ResNet-50 e DenseNet-169) em cenários federados com 3, 5 e 7 clientes, utilizando três algoritmos de agregação: FedAvg, FedAvgM e FedProx. Os experimentos foram conduzidos em duas fases: Fase I (548 pacientes) e Fase II (1.287 pacientes). Na Fase I, DenseNet-169 + FedAvg alcançou o melhor desempenho (86,97% de acurácia, 3 clientes), demonstrando maior eficiência com dados limitados. Na Fase II, ResNet-50 + FedProx obteve a maior acurácia, seguida pela DenseNet-169 + FedAvg, com uma diferença estatisticamente não significativa entre os cenários. A configuração de melhor desempenho atingiu 99,26% de acurácia, 98,88% de *recall* e 99,35% de precisão. FedAvgM exibiu instabilidade persistente atribuída ao acúmulo excessivo de *momentum*. Os resultados demonstram que o modelo é eficaz para detecção de AD, com DenseNet-169 preferível para conjuntos de dados menores, indicando que, ao escalar o volume de dados, a escolha da arquitetura neural torna-se menos crítica. Este estudo, além de validar o potencial do FL para aplicações clínicas seguras e escaláveis e fornecer diretrizes para otimização de hiperparâmetros também comprova o impacto do aumento de dados em um ambiente federado, em que há uma distribuição heterogênea.

Palavras-chave: Doença de Alzheimer. Aprendizado Federado. Aprendizado Profundo. Aprendizado Dividido. Processamento de Imagem.

Predictive performance evaluation of neural networks in federated learning for Alzheimer's disease stage classification.

ABSTRACT

Alzheimer's Disease (AD) is a progressive neurodegenerative disorder for which early detection is essential to slow disease progression and improve patients' quality of life. Global prevalence is projected to triple by 2050. This study investigates the application of Federated Learning (FL) combined with Split Learning (SL) and Convolutional Neural Networks (CNNs) to classify individuals into three stages: AD, Mild Cognitive Impairment (MCI), and Cognitively Normal (CN). A critical aspect of FL deployment is the careful analysis of hyperparameters and the selection of the optimal neural architecture. We evaluated two architectures (ResNet-50 and DenseNet-169) in federated scenarios with 3, 5, and 7 clients, using three aggregation algorithms: FedAvg, FedAvgM, and FedProx. Experiments were conducted in two phases: Phase I (548 patients) and Phase II (1,287 patients). In Phase I, DenseNet-169 + FedAvg achieved the best performance (86.97% accuracy, 3 clients), demonstrating greater efficiency with limited data. In Phase II, ResNet-50 + FedProx obtained the highest accuracy, followed by DenseNet-169 + FedAvg, with a statistically non-significant difference between the scenarios. The top-performing configuration achieved 99.26% accuracy, 98.88% *recall*, and 99.35% precision. FedAvgM exhibited persistent instability attributed to excessive *momentum* accumulation. The results demonstrate that the model is effective for AD detection, with DenseNet-169 being preferable for smaller datasets, and indicate that, as data volume scales, the choice of neural architecture becomes less critical. This study, in addition to validating the potential of FL for secure and scalable clinical applications and providing guidelines for hyperparameter optimization, also demonstrates the impact of increased data volume in a federated environment characterized by heterogeneous distribution.

Keywords: Alzheimer's Disease. Federated Learning. Deep Learning. Split Learning. Medical Image Processing.

SUMÁRIO

LISTA DE FIGURAS **9**
LISTA DE TABELAS **10**
LISTA DE ABREVIATURAS E SIGLAS **11**
LISTA DE SÍMBOLOS **13**
1 INTRODUÇÃO **15**
1.1 Motivação **16**
1.2 Problema de Pesquisa **16**
1.3 Objetivos **17**
1.4 Organização do texto **17**
2 CONTEXTO **18**
2.1 Panorama da Doença de Alzheimer no Brasil **18**
2.2 Procedimentos diagnósticos da Doença de Alzheimer **20**
2.2.1 Estágios clínicos da demência 20
2.2.2 Diagnóstico clínico 21
2.2.3 Diagnóstico por imagem 22
2.3 Aprendizado profundo aplicado a processamento de imagens **27**
2.4 Redes neurais convolucionais **28**
2.5 Advento do aprendizado federado aplicado à saúde **30**
2.6 Algoritmos de agregação **32**
2.7 Considerações finais **34**
3 TRABALHOS RELACIONADOS **35**
3.1 Revisão Sistemática da Literatura **35**
3.1.1 Objetivos 36
3.1.2 Questões 37
3.1.3 Estratégia 39
3.1.4 Critérios de inclusão e exclusão 39
3.1.5 Resultados 44
3.1.6 Lacunas de Pesquisa e Oportunidades 50
3.1.6.1 Ausência de diretrizes padronizadas para o treinamento. 50
3.1.6.2 Comparação sistemática de agregadores sob heterogeneidade. 50
3.1.6.3 Escassez de avaliações de arquiteturas híbridas e do efeito do volume de dados. 50
3.1.6.4 Heterogeneidade de protocolos e métricas. 50
3.1.6.5 Integração multimodal e garantias de privacidade 51
3.2 Considerações finais **51**
4 MODELO **52**
4.1 Visão Geral da Arquitetura Proposta **52**
4.2 Cliente **54**
4.2.1 Particionamento de Dados 54
4.2.2 Arquitetura do Cliente 54
4.2.2.1 ResNet-50 55
4.2.2.2 DenseNet-169 55
4.2.3 Treinamento Local 57
4.2.4 Balanceamento em Batch 57

4.2.5 Comunicação com o Servidor ... 58
4.3 Servidor ... 58
4.3.1 Forward Pass e Cálculo da Perda ... 58
4.3.2 Retropropagação e Gradientes ... 59
4.3.3 Agregação Federada ... 59
4.3.4 Avaliação Global ... 59
4.4 Considerações finais ... 60
5 METODOLOGIA ... 61
5.1 Dados utilizados ... 61
5.2 Pré-processamento de dados ... 62
5.3 Particionamento dos dados ... 63
5.4 Balanceamento dos dados ... 64
5.5 Protocolos experimentais ... 65
5.6 Limitações ... 67
5.7 Considerações finais ... 68
6 RESULTADOS E AVALIAÇÕES ... 69
6.1 Visão geral ... 69
6.2 Fase I ... 71
6.2.1 Escalabilidade em Função do Número de Clientes ... 74
6.2.2 Modelos de Agregação ... 77
6.2.3 Impacto do *Batch Size* ... 78
6.2.4 *Recall* vs Precisão ... 79
6.3 Fase II ... 79
6.3.1 Escalabilidade em Função do Número de Clientes ... 82
6.3.2 Modelos de Agregação ... 85
6.3.3 Impacto do *Batch Size* ... 86
6.3.4 *Recall* vs Precisão ... 86
6.4 Validação Estatística ... 86
6.4.1 Normalidade das diferenças (Shapiro–Wilk) ... 87
6.4.2 Teste de Wilcoxon ... 88
6.4.3 Tamanho de Efeito (Correlação de Postos Bisserial) ... 88
6.4.4 Correção de múltiplas comparações (Benjamini–Hochberg) ... 89
6.5 Comparação com o Estado da Arte ... 89
6.6 Considerações finais ... 92
7 CONCLUSÃO ... 93
7.1 Contribuições ... 94
7.2 Trabalhos Futuros ... 95
REFERÊNCIAS ... 96
Apêndice A - Modelo Adaptado ... 102
Apêndice B - Resultados estatísticos ... 115

LISTA DE FIGURAS

Figura 2.1 Série histórica de internações (a) e óbitos (b) por Doença de Alzheimer no Brasil (2008-2024). 18
Figura 2.2 MRI craniana com foco na análise de substância branca. 24
Figura 2.3 Escala MTA mostrando os graus de 0 a 4 cortes coronais MRI-T1 25
Figura 2.4 Representação visual dos graus de atrofia posterior em diferentes cortes de MRI 26
Figura 2.5 Arquitetura da CNN 28
Figura 2.6 Conectividade em redes convolucionais: Padrão, ResNet e DenseNet .. 29
Figura 2.7 Ciclo do aprendizado federado 31

Figura 3.1 Fluxograma da RSL no padrão PRISMA. 36

Figura 4.1 Aprendizado dividido 53
Figura 4.2 Visão geral do modelo *Split Federated Learning* 60

Figura 5.1 Imagens em MRI representativas dos estágios da AD. 61
Figura 5.2 Distribuição de dados na Fase I 63
Figura 5.3 Distribuição de dados na Fase II 64

Figura 6.1 Compilação de matrizes de confusão para diferentes combinações de algoritmos de agregação e arquiteturas de modelo, detalhando o desempenho da classificação para 3 e 5 clientes. 75
Figura 6.2 Compilação das curvas de Acurácia para cenários selecionados da Fase I. 76
Figura 6.3 Curvas de validação do FedAvgM - Fase I. 78
Figura 6.4 Compilação das curvas de Acurácia para cenários selecionados da Fase II. 83
Figura 6.5 Compilação de matrizes de confusão para diferentes combinações de algoritmos de agregação e arquiteturas de modelo, detalhando o desempenho da classificação para 3 e 5 clientes. 84
Figura 6.6 Curvas de validação do FedAvgM - Fase II. 85

LISTA DE TABELAS

Tabela 2.1 Internações e Óbitos por Alzheimer no Brasil (2008-2024) 19
Tabela 2.2 Guia para Avaliação de Manifestações Neuropsiquiátricas na Anamnese 23

Tabela 3.1 Relação entre as questões e os objetivos da RSL 38
Tabela 3.2 Critérios de inclusão (CI) e exclusão (CE) adotados na RSL..... 40
Tabela 3.3 Publicações selecionadas para a Revisão Sistemática da Literatura (RSL). 40
Tabela 3.4 Artigos e suas características. 46

Tabela 4.1 Arquitetura em camadas da ResNet-50. 55
Tabela 4.2 Arquitetura em camadas da DenseNet-169. 56

Tabela 5.1 Configurações dos experimentos 65
Tabela 5.2 Cenários simulados para os testes. 66

Tabela 6.1 Resultados médios (% ± desvio padrão) dos experimentos na Fase I 72
Tabela 6.2 Melhores resultados da Fase I. 78
Tabela 6.3 Fase I — Recall vs Precisão 79
Tabela 6.4 Resultados médios (% ± desvio padrão) dos experimentos na Fase II 80
Tabela 6.5 Melhores resultados da Fase II. 82
Tabela 6.6 Fase II — Recall vs Precisão 86
Tabela 6.7 Resultados do teste Shapiro-Wilk para as cinco métricas de desempenho. Valores de W (estatística do teste) e p-valores correspondentes. 87
Tabela 6.8 Resultados de classificação (média ± desvio padrão) de diferentes configurações de aprendizado federado em termos de quatro métricas. . 90
Tabela 6.9 Resultados da classificação da arquitetura BANet e de outros modelos tradicionais de aprendizado profundo. 91
Tabela 6.10 Comparação de desempenho da arquitetura ResNet-50 aprimorada com ativação Mish. 92

LISTA DE ABREVIATURAS E SIGLAS

ACC	Acurácia
ADNI	Iniciativa de Neuroimagem da Doença de Alzheimer (*Alzheimer's Disease Neuroimaging Initiative*)
AIH	Autorização de Internação Hospitalar
AUC	Área sob a Curva ROC (*Area Under the Curve*)
CN	Cognitivamente Normal
CNN	Redes Neurais Convolucionais (*Convolutional Neural Networks*)
CPU	Unidade Central de Processamento (*Central Processing Unit*)
AD	Doença de Alzheimer
DICOM	*Digital Imaging and Communications in Medicine*
DL	Aprendizado Profundo (*Deep Learning*)
EMT	Atrofia de Estruturas Mesiais Temporais
FL	Aprendizado Federado (*Federated Learning*)
FLOPs	Operações de Ponto Flutuante por segundo (*Floating Point Operations per Second*)
FP32	Pontos Flutuantes de Precisão Simples (*Single-Precision Floating-Point*)
FP64	Pontos Flutuantes de Precisão Dupla (*Double-Precision Floating-Point*)
FNNDSC	*Fetal-Neonatal Neuroimaging and Developmental Science Center*
GPU	Unidade de Processamento Gráfico (*Graphics Processing Unit*)
IA	Inteligência Artificial
IBGE	Instituto Brasileiro de Geografia e Estatística
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos (*Institute of Electrical and Electronics Engineers*)
IR	Razão de Desbalanceamento (*Imbalance Ratio*)
JPEG	*Joint Photographic Experts Group*

LCR	Líquido Cefalorraquidiano
MCI	Comprometimento Cognitivo Leve (*Mild Cognitive Impairment*)
ML	Aprendizado de Máquina (*Machine Learning*)
MLP	Perceptron Multicamadas (*Multi-Layer Perceptron*)
MRI-T1	Imagem de Ressonância Magnética Ponderada em T1
MTA	Escala de Atrofia Temporal Medial
NIA-AA	*National Institute on Aging-Alzheimer's Association*
NIfTI	*Neuroimaging Informatics Technology Initiative*
non-IID	Distribuição de dados não homogênea entre os clientes
OASIS	Série de Estudos de Imagem de Acesso Aberto (*Open Access Series of Imaging Studies*)
PAR	Lobo Parietal
PCA	Atrofia Parietal
PCS	Sulco Cingulado Posterior
PET	Tomografia por Emissão de Pósitrons (*Positron Emission Tomography*)
PNG	*Portable Network Graphics*
POS	Sulco Parieto-Occipital
SEAs	Codificadores Automáticos Empilhados
SFL	*Split Federated Learning*
SGD	Descida de Gradiente Estocástico (*Stochastic Gradient Descent*)
SIH	Sistema de Informações Hospitalares
SL	Aprendizado Dividido (*Split Learning*)
SUS	Sistema Único de Saúde
TC	Tomografia Computadorizada

LISTA DE SÍMBOLOS

$\sum \frac{a}{b}$	Somatório do produtório
w	Vetor de parâmetros (pesos) do modelo global
w^*	Vetor de pesos ótimos que minimiza a perda global
n_k	Número de amostras no cliente k
n	Número total de amostras em todos os clientes ($n = \sum_k n_k$)
K	Número total de clientes participantes
$F_k(w)$	Função de perda média local do cliente k
$f(w)$	Função de perda global (média ponderada das perdas locais)
S_t	Subconjunto de clientes selecionados na rodada t
C	Fração de clientes escolhidos por rodada ($m = C \cdot K$)
E	Número de épocas de treinamento local realizadas por cliente
η	Taxa de aprendizagem (learning rate) usada no SGD local
b	Minilote (minibatch) de tamanho B utilizado no SGD
$\nabla l(b)$	Gradiente da função de perda calculado no minilote b
β	Coeficiente de momentum no algoritmo FedAvgM
v	Velocidade acumulada (momentum) mantida pelo servidor no FedAvgM
Δt	Atualização média ponderada enviada pelos clientes na rodada t
μ	Hiperparâmetro de proximidade introduzido no FedProx
$H(x)$	Função desejada (mapeamento) que a camada residual deve aprender na ResNet
$F(x)$	Função residual aprendida, definida por $F(x) = H(x) - x$
x	Entrada (ativação) que flui através da conexão de atalho em ResNet
f_c	Sub-rede do cliente (camadas iniciais)
w_c	Pesos globais da parte cliente do modelo
$w_c^{t+1,k}$	Pesos do cliente k após a rodada t

f_s	Sub-rede do servidor (camadas finais)
w_s	Pesos globais da parte servidor do modelo
h_k	Tensor de ativações intermediárias (“smashed data”) do cliente k
$\hat{y}$	Predição do modelo
$\mathcal{L}$	Função de perda
$\mathcal{D}_k$	Conjunto de dados local do cliente k
T	Número total de rodadas globais de treinamento
η_g	Taxa de aprendizagem global aplicada ao momentum (FedAvgM)
v_c, v_s	Vetores de momento do cliente e do servidor
Δ_c, Δ_s	Atualizações médias dos pesos do cliente e do servidor
$\nabla_{h_k}\mathcal{L}$	Gradiente da perda em relação às ativações h_k
w_c^k	Matriz de pesos (parâmetros) da sub-rede do cliente k na iteração atual
$w_c^{k,\text{new}}$	Pesos atualizados da sub-rede do cliente k após a aplicação do otimizador
$\mathcal{L}_k$	Função de perda específica computada para o cliente k
$\nabla_{w_c^k}$	Gradiente da função de perda em relação aos pesos do cliente k

1 INTRODUÇÃO

A doença de Alzheimer (AD) é um distúrbio neurodegenerativo irreversível caracterizado por declínio cognitivo progressivo e é a principal causa de demência, representando 70% dos casos em todo o mundo (FARINA et al., 2020; MIHELČIĆ et al., 2017; ALZHEIMER'S ASSOCIATION, 2018). Projeções indicam que a prevalência global de demência triplicará até 2050 (ALBERDI; AZTIRIA; BASARAB, 2016). Os sintomas iniciais surgem como perda de memória e problemas de linguagem devido a neurônios danificados ou destruídos. Os pacientes normalmente vivem com sintomas de Alzheimer durante anos. Com o tempo, os sintomas tendem a aumentar e afetar a capacidade dos indivíduos de realizar atividades cotidianas (ALZHEIMER'S ASSOCIATION, 2018). Como ainda não foi desenvolvida nenhuma cura para a AD, o foco do tratamento atual é reduzir a velocidade de progressão para o estágio mais grave.

Apesar dos avanços recentes nos ensaios clínicos, diagnosticar essa condição está se tornando cada vez mais desafiador. Isso se deve ao aumento do número de pacientes, à possibilidade de erros na interpretação visual de neuroimagens e à presença de padrões e correlações ainda não compreendidos entre diferentes biomarcadores (ALZHEIMER'S ASSOCIATION, 2018). Além disso, a degeneração cerebral associada ao envelhecimento complica ainda mais a identificação precoce da AD (NAEEM et al., 2022). Portanto, o auxílio ao diagnóstico por computador está se tornando fundamental para facilitar a identificação da doença.

Nesse cenário, ferramentas de Inteligência Artificial (IA) baseadas em aprendizado profundo (DL) emergem como aliadas potenciais para auxiliar decisões clínicas. Contudo, esses modelos enfrentam um desafio: a alta demanda de conjuntos de dados de treinamento. Embora muitos centros de imagem possuam vastos bancos de dados, muitas imagens não estão rotuladas, dificultando o aprendizado do modelo (NAEEM et al., 2022). A anotação manual dessas imagens não rotuladas por radiologistas treinados é necessária, tornando a produção de um conjunto de dados de treinamento demorada e um fator limitante no desenvolvimento de soluções de IA para neuroimagens (RAUNIYAR et al., 2022).

1.1 Motivação

O Aprendizado Federado (FL) oferece uma alternativa descentralizada, permitindo que instituições treinem localmente, compartilhem atualizações do modelo de agregação e preservem a privacidade dos dados. Este paradigma é promissor para treinar modelos colaborativamente sem compartilhar dados brutos (MCMAHAN et al., 2016). No entanto, sua adoção prática em ambientes hospitalares heterogêneos esbarra em um desafio operacional crucial: a exigência de que cada cliente (hospital) treine localmente um modelo completo de DL, demandando recursos computacionais (GPUs de alto desempenho) frequentemente indisponíveis em instituições menores ou em regiões com poucos recursos (LI et al., 2020).

Uma forma de otimizar os treinamentos é aplicar o Aprendizado Dividido (SL), com o intuito de distribuir a carga computacional entre clientes e servidor (VEPAKOMMA et al., 2018). Divide-se a rede neural em duas partes: os clientes calculam as camadas iniciais localmente, enquanto um servidor central finaliza as passagens de propagação para frente (*forward*) e retropropagação (*backward*), reduzindo os FLOPs e o uso da memória. Esse modelo oferece duas vantagens: (1) carga computacional significativamente reduzida nos dispositivos dos clientes e (2) privacidade aprimorada, uma vez que nenhuma das partes acessa a arquitetura completa do modelo da outra, tornando-o particularmente adequado para detecção e classificação da AD em hospitais com diferentes capacidades técnicas (THAPA et al., 2022).

1.2 Problema de Pesquisa

A configuração de ambientes federados exige uma análise criteriosa de múltiplos fatores que influenciam diretamente a eficácia e a estabilidade do treinamento, indo além da otimização de hiperparâmetros convencionais de ML. Primeiramente, a escolha da arquitetura da rede neural deve ser centrada nas restrições computacionais dos dispositivos dos clientes. Posteriormente, um desafio central do FL, segundo Kairouz et al. (2021), é o *client drift*, que ocorre quando os modelos locais dos clientes divergem significativamente devido à heterogeneidade estatística dos dados. Para mitigar isso, a escolha do modelo de agregação adequado é fundamental. E, por fim, a otimização envolve um conjunto de hiperparâmetros específicos de FL, como a fração de clientes participantes, a taxa de aprendizado e o número de épocas locais.

1.3 Objetivos

Diante deste contexto, este trabalho tem como objetivo avaliar comparativamente o desempenho de arquiteturas neurais profundas submetidas à técnica de FL para classificação multiclasse dos estágios de AD: Cognitivamente Normal (CN), Comprometimento Cognitivo Leve (MCI) e Paciente com Alzheimer (AD), a partir de imagens de ressonância magnética. Por fim, propõem-se protocolos otimizados de hiperparâmetros e configurações para treinamentos envolvendo AD e FL. Para tal, estrutura-se, nestes objetivos específicos interligados:

- Ajustar um modelo de FL integrado a DL para classificação multiclasse de estágios de AD, utilizando SL para reduzir demandas locais de hardware;
- Destacar heurísticas de configuração de hiperparâmetros (número de clientes, tamanho de lote (*batch*), modelos de agregação) que obtiveram os melhores desempenhos.

Ao integrar avanços em FL, arquiteturas profundas e diagnóstico médico, esta pesquisa visa fornecer: (i) diretrizes técnicas reprodutíveis para implantação de sistemas colaborativos de classificação de AD; (ii) protocolos computacionalmente eficientes adaptados a ambientes heterogêneos, contribuindo para a democratização de ferramentas de precisão em instituições com recursos limitados.

1.4 Organização do texto

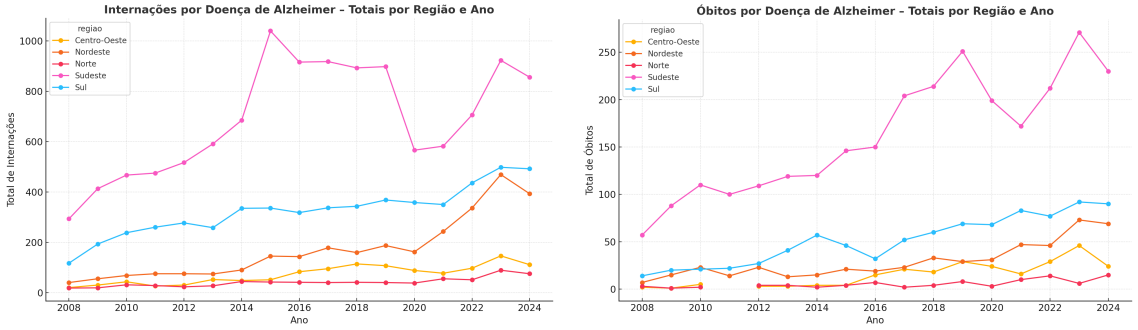
Para além do capítulo introdutório, o trabalho organiza-se em seis capítulos subsequentes. Inicialmente, o Capítulo 2 apresenta o panorama da doença de Alzheimer e os conceitos essenciais de DL, SL e FL. O Capítulo 3 aprofunda a análise do estado da arte por meio de uma revisão sistemática da literatura, que mapeia e detalha os principais trabalhos relacionados à pesquisa. A arquitetura do modelo proposto é detalhada no Capítulo 4. Os procedimentos metodológicos e o plano de trabalho, que abrangem desde o pré-processamento dos dados até a implementação e configuração de hiperparâmetros, são descritos no Capítulo 5. O Capítulo 6 é dedicado à apresentação e à análise crítica dos resultados obtidos nos experimentos, utilizando métricas como acurácia, sensibilidade, precisão, área sob a curva ROC e matriz de confusão, além de uma análise estatística sobre as medidas obtidas. Finalmente, o Capítulo 7 sintetiza as conclusões da pesquisa e discute suas implicações, além de propor caminhos para investigações futuras.

2 CONTEXTO

Neste capítulo, será inicialmente apresentado o panorama epidemiológico da doença de Alzheimer, com ênfase no contexto brasileiro. Em um escopo global, serão detalhados os métodos clínicos e de neuroimagem universalmente adotados para o diagnóstico e a classificação dos estágios da doença. Por fim, será analisada a aplicabilidade de algoritmos de inteligência artificial como ferramentas de auxílio na detecção e classificação da enfermidade.

2.1 Panorama da Doença de Alzheimer no Brasil

Ao longo do período disponível no Sistema de Informações Hospitalares (SIH) do DATASUS (2008 - 2024), foi registrado um total de 22.065 internações em que a AD foi o diagnóstico principal. Associados a essas internações, ocorreram 4.457 óbitos hospitalares e o custo total dessas internações para o sistema público de saúde atingiu R$227.769.273,91. Observa-se na Figura 2.1 uma flutuação no número de internações ao longo dos anos, com um valor inicial de 488 internações em 2008, atingindo um pico de 2.125 em 2023. Os óbitos hospitalares também apresentaram variação, com 83 em 2008 e um pico de 488 em 2023.



(a) Internações por Região e Ano. (b) Óbitos por Região e Ano.

Figura 2.1 – Série histórica de internações (a) e óbitos (b) por Doença de Alzheimer no Brasil (2008-2024).
Fonte: SIH/DATASUS

Com base no levantamento epidemiológico conduzido por Almeida e Pereira (2022) (Tabela 2.1), onde foi analisado o período de 2008 até 2020, a região Sudeste concentrou a maioria absoluta das internações (57,22%) e dos óbitos (65,78%), exibindo a maior taxa de mortalidade (21,34%). Norte e Centro-Oeste apresentaram menores números absolutos e taxas de mortalidade inferiores ao nacional, Sul e Nordeste tiveram internações intermediárias (24,73% e 9,57%, respectivamente), mas o Nordeste apresen-

tou mortalidade (18,33%) ligeiramente próxima da média nacional (18,69%).

Para dar continuidade à análise, foram extraídos do banco de dados do SUS os registros referentes aos anos de 2021-2024 (Tabela 2.1).

A região Sudeste mantém a maior concentração de internações (43,9%) e óbitos (54,6%), com agravamento na taxa de mortalidade, que aumentou de 21,34% (2008-2020) para 28,71% (2021-2024). O Centro-Oeste, apesar de representar apenas 6,2% das internações, apresenta mortalidade elevada (26,7%), sugerindo diagnóstico tardio ou menor disponibilidade de serviços especializados.

O Nordeste mostrou evolução positiva: elevou sua participação nas internações de 9,6% para 20,6%, enquanto reduziu a mortalidade de 18,3% para 16,3%. O Sul manteve-se estável em participação relativa (24,8% para 25,4%), porém sua mortalidade subiu de 14,1% para 19,3%. O Norte permanece com a menor representatividade (3,9%) e mortalidade abaixo da média nacional (17,13% contra 23,27%).

Em termos absolutos, houve redução significativa: de 14.872 para 7.430 internações (-50,0%) e de 2.780 para 1.729 óbitos (-37,8%) entre os períodos analisados. Essa diminuição pode estar associada a mudanças recentes nos protocolos de autorização de internação hospitalar (AIH), que passaram a priorizar atendimento ambulatorial e domiciliar para casos de demência, conforme diretrizes do Ministério da Saúde implementadas a partir de 2020.

Tabela 2.1 – Internações e Óbitos por Alzheimer no Brasil (2008-2024)

Regiões	**Período 2008-2020**			**Período 2021-2024**		
	Intern.	**Óbitos**	**Mort. (%)**	**Intern.**	**Óbitos**	**Mort. (%)**
Norte	425	44	10,35	286	49	17,13
Nordeste	1.424	261	18,33	1.518	248	16,34
Sudeste	8.571	1.829	21,34	3.274	940	28,71
Sul	3.679	519	14,11	1.886	365	19,35
Centro-Oeste	773	127	16,43	466	127	26,70
Total	**14.872**	**2.780**	**18,69**	**7.430**	**1.729**	**23,27**

Fonte: Período 2008-2020 adaptado de (ALMEIDA; PEREIRA, 2022); Período 2021-2024: Autor, 2025

Além disso, o Instituto Brasileiro de Geografia e Estatística (IBGE) projeta que, até 2070, o Brasil terá cerca de 70 milhões de idosos (≥ 60 anos), posicionando-se como o sexto país em número absoluto de pessoas idosas.

Em suma, o panorama epidemiológico evidencia que a AD está assumindo contornos cada vez mais desafiadores no Brasil, com crescimento sustentado de internações e

mortalidade em quase todas as regiões. Dessa forma, a consolidação de ações de diagnóstico precoce torna-se fundamental para enfrentar a progressiva carga clínica e financeira.

2.2 Procedimentos diagnósticos da Doença de Alzheimer

O diagnóstico precoce da AD é essencial para o tratamento eficaz e a manutenção da qualidade de vida do paciente. Sabendo disso, desenvolveram-se diversas metodologias para detectar essa enfermidade. Um dos sinais iniciais aparentes é a perda de memória abrupta, manifestação prevalente em aproximadamente 85% dos casos, outras formas de apresentação incluem alterações na linguagem, nas capacidades visuoespaciais, nas funções executivas e nas habilidades motoras complexas. Por conseguinte, a Academia Brasileira de Neurologia (DEPARTAMENTO CIENTÍFICO DE NEUROLOGIA COGNITIVA E DO ENVELHECIMENTO, 2022), baseada no *National Institute on Aging-Alzheimer 's Association* (NIA-AA), atualizou os critérios diagnósticos, a fim de facilitar a identificação das formas iniciais (SCHILLING et al., 2022).

2.2.1 Estágios clínicos da demência

A AD apresenta um curso clínico progressivo, tipicamente classificado em estágios. Na fase inicial, correspondente à demência leve, o diagnóstico caracteriza-se predominantemente pela exacerbação de sintomas amnésticos. Estes incluem: anomia (comprometimento da capacidade de nomeação), desorientação temporal-espacial, prejuízo da memória operacional (manifestado por dificuldades em tarefas simultâneas) e défices cognitivos diversos (REISBERG et al., 1982). Concomitantemente, observa-se a progressão de sintomas neuropsiquiátricos, tais como apatia, depressão, ansiedade e ausência de consciência dos défices cognitivos (anosognosia), conforme a evolução da doença. O estágio de demência moderada é marcado por uma acentuada perda de autonomia funcional, com o indivíduo tornando-se progressivamente incapaz de realizar atividades instrumentais e básicas da vida diária sem auxílio. A memorização de nomes de familiares e de eventos recentes apresenta dificuldade significativa. Observa-se ainda a intensificação de sintomas preexistentes, como a desorientação e o surgimento ou agravamento de manifestações neuropsiquiátricas, incluindo delírios e alucinações. Na fase grave da demência, instala-se a dependência total para cuidados. Ocorre um compro-

metimento profundo da memória episódica remota, frequentemente restrita a fragmentos autobiográficos. A linguagem torna-se severamente limitada ou ausente. Complicações neurológicas e sistêmicas surgem nas fases mais avançadas, incluindo incontinência urinária e fecal, sinais parkinsonianos, mioclonias e crises epilépticas (PALOP; MUCKE, 2009) (XU et al., 2021). A sobrevida média após o início dos sintomas é estimada em 12 anos (VERMUNT et al., 2019).

2.2.2 Diagnóstico clínico

A análise clínica do indivíduo com suspeita de AD fundamenta-se na avaliação minuciosa dos domínios cognitivos e do grau de comprometimento funcional, conforme estabelecido nos critérios diagnósticos consensuais e nas baterias padronizadas de testagem neuropsicológica. Dado o caráter progressivo da patologia, torna-se imperativo o acompanhamento médico longitudinal desde a emergência das primeiras manifestações clínicas (SPERLING et al., 2011).

A condução de uma anamnese, com ênfase nas alterações cognitivas e neuropsiquiátricas mais prevalentes na AD, é a primeira etapa para aumentar a acurácia diagnóstica. Este procedimento permite não apenas a identificação do subtipo de apresentação inicial e do estágio evolutivo da patologia, mas também a sua diferenciação de outras enfermidades neurodegenerativas. A entrevista clínica, envolvendo o paciente e um familiar ou informante próximo, deve investigar sistematicamente, conforme está na (Tabela 2.2):

1. Manifestações Neuropsiquiátricas: Avaliação da presença de sintomas como depressão, ansiedade, apatia, ideação delirante, alucinações e comportamentos motores aberrantes, desinibidos ou socialmente inadequados.

2. Deficiências Cognitivas: Investigação de disfunções nos seguintes domínios cognitivos, primariamente afetados pela AD:

    - Memória Episódica: Caracterizada por dificuldades no registro e evocação de fatos e datas recentes, listas de compras, compromissos, localização de objetos, bem como pela repetição de perguntas ou comentários.
    - Funções Executivas: Observadas como dificuldades na manutenção da atenção concentrada, tomada de decisões, planejamento de atividades, resolução de problemas cotidianos, realização de compras e manejo de finanças (mesmo

em pequenas quantias). Adicionalmente, podem incluir perda de motivação, iniciativa e julgamento inadequado de situações.

- Habilidades Visuoespaciais e Práxicas: Manifestam-se como dificuldades na orientação espacial (intra e extradomiciliar), no vestir-se, nos cuidados pessoais (como pentear-se ou barbear-se), no uso de objetos comuns e no reconhecimento de fisionomias familiares. Pode ocorrer também perda de destreza em tarefas previamente bem executadas.
- Linguagem: Evidenciada por dificuldades anômicas (encontrar palavras durante a conversação ou para nomear objetos e pessoas), déficits na compreensão de palavras ou frases, dificuldades na articulação de explicações ou em se fazer entender, frequentemente acompanhadas por um vocabulário empobrecido e redução da fluência verbal.

2.2.3 Diagnóstico por imagem

A análise de neuroimagem, como a tomografia computadorizada (TC) ou a ressonância magnética (RM), desempenha um papel fundamental no diagnóstico da AD, atuando em dois eixos complementares: a exclusão de patologias alternativas (hematomas subdurais, tumores ou lesões vasculares) e a identificação de padrões de atrofia cerebral característicos da neurodegeneração (SCHILLING et al., 2022). A RM emerge como modalidade de escolha devido à sua resolução anatômica superior e capacidade de aplicação de sequências avançadas (volumetria hipocampal, avaliação de substância branca e técnicas de difusão), que proporcionam maior *recall* na diferenciação entre AD, demência vascular, encefalopatias priônicas e outras demências degenerativas (NITRINI et al., 2022).

A análise padronizada focaliza a avaliação de estruturas mesiais temporais (EMT), com ênfase no hipocampo e córtex entorrinal. A atrofia hipocampal constitui o marcador mais consolidado nas formas típicas amnésicas, correlacionando-se com os estágios III-VI da escala de Braak e com déficits de memória episódica (SCHILLING et al., 2022). Protocolos quantitativos (escala Medial Temporal Atrophy - MTA) e técnicas de avaliação volumétrica automatizada são recomendados para aumentar a reprodutibilidade diagnóstica. Esta escala avalia a largura da fissura coroideia do corno temporal, e também a altura do hipocampo.

Tabela 2.2 – Guia para Avaliação de Manifestações Neuropsiquiátricas na Anamnese

Manifestações NEUROPSIQUIÁTRICAS	**Aspecto**	**Questionamento**
Sintomas afetivos	Humor	Episódios de tristeza prolongada, choro fácil ou irritabilidade?
	Ansiedade	Preocupação excessiva, agitação psicomotora ou sintomas somáticos?
	Motivação	Redução de interesses, passividade em atividades rotineiras ou perda de iniciativa?
Sintomas Psicóticos	Delírios	Ideias persecutórias (roubo/infidelidade) ou fenômenos de falsos reconhecimentos?
	Alucinações	Percepções visuais/auditivas sem estímulo (presença de pessoas/objetos)?
Comportamento	Desinibição	Comentários sociais inadequados, perda de pudor ou invasão de espaço pessoal?
	Comprometimento Motor	Agitação, deambulação sem propósito ou comportamentos repetitivos?
	Social	Negligência de higiene pessoal, apatia social ou perda de empatia?

Fonte: Adaptado de (CAMOZZATO et al., 2015)

A avaliação estrutural de uma RM craniana deve ser feita de forma sistematizada, pontuando os escores de Leucoaraiose (Fazekas), atrofia hipocampal (MTA) e atrofia parietal (PCA):

- Escala de Fazekas (FAZEKAS et al., 1987): Protocolo utilizado para quantificar o grau de lesões hiperintensas na substância branca, visíveis em MRI-T2, conforme a Figura 2.2.
  - Grau 1: Caracterizado por focos esparsos com diâmetro inferior a 10 mm e/ou

áreas de lesões confluentes menores que 20 mm de diâmetro. Isso indica um nível leve de alteração.

- Grau 2 (Confluência Inicial): Apresenta lesões únicas entre 10 e 20 mm e/ou áreas de lesões agrupadas com mais de 20 mm de diâmetro. Neste estágio, as lesões começam a se tornar mais proeminentes e coalescer.
- Grau 3 (Confluência Severa): Refere-se a amplas áreas de confluência, com lesões únicas que excedem 20 mm de diâmetro. Este é o grau mais grave, indicando uma extensão significativa das alterações na substância branca.

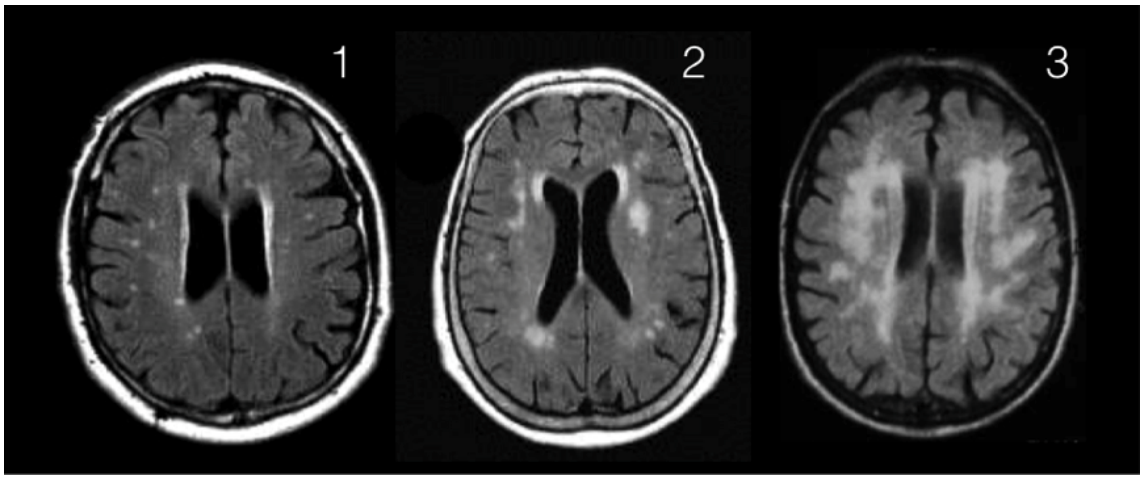


Figura 2.2 – MRI craniana com foco na análise de substância branca.
Fonte: (KIM; MACFALL; PAYNE, 2008)

- Escore de atrofia temporal (MTA) ou escala de Scheltens (Scheltens et al., 1995): Consiste numa análise visual realizada na MRI, especificamente em cortes coronais ponderados em T1 na altura do hipocampo, como mostra a Figura 2.3, sendo útil para quantificar o grau de atrofia do lobo temporal medial, especialmente do hipocampo, que é frequentemente afetado por doenças neurodegenerativas.
  - Grau 0: Ausência de Atrofia: Não há líquido cefalorraquidiano (LCR) visível ao redor do hipocampo, indicando que a estrutura está preservada.
  - Grau 1: Atrofia Muito Leve: A fissura coróide apresenta um leve alargamento, sugerindo um início sutil de atrofia
  - Grau 2: Atrofia Leve: Observa-se um alargamento moderado da fissura coróide. Além disso, há um leve aumento do corno temporal e uma discreta perda no tamanho do hipocampo.

- Grau 3: Atrofia Moderada: Caracteriza-se por um alargamento acentuado da fissura coróide, um aumento moderado do corno temporal e uma perda de tamanho hipocampal também moderada.
- Grau 4: Atrofia Severa: Alargamento da fissura coróide é marcado, o corno temporal está significativamente aumentado, e o hipocampo encontra-se acentuadamente atrofiado, a ponto de perder sua estrutura interna.

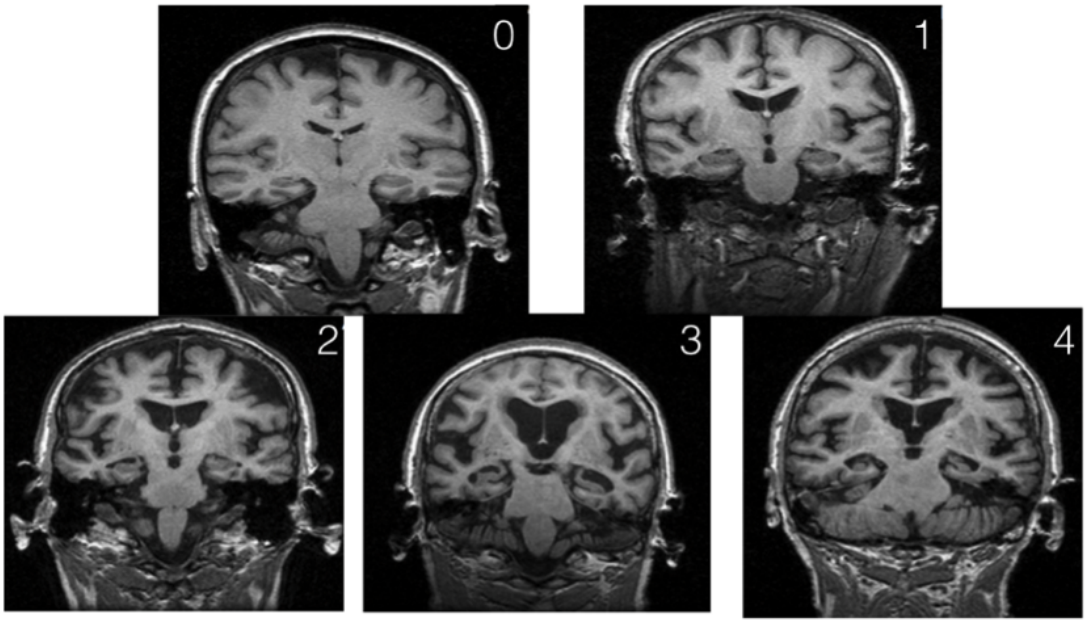


Figura 2.3 – Escala MTA mostrando os graus de 0 a 4 cortes coronais MRI-T1
Fonte: (SCHELTENS et al., 1995)

- O Escore de Koedam (KOEDAM et al., 2012) é uma ferramenta desenvolvida para a avaliação visual da atrofia do córtex parietal em exames de MRI, Figura 2.4. Para a sua aplicação, o cérebro é analisado em três planos:
- Plano Sagital: Avaliam-se o sulco cingulado posterior (PCS), o sulco parieto-occipital (POS) e o giro pré-cuneus (PRE).
- Plano Coronal: Foca-se no PCS e no lobo parietal (PAR).
- Plano Axial: Analisam-se os PCS e os PARs.

  O grau final do escore (de 0 a 3) é determinado pelos achados mais severos observados em qualquer um dos planos:

  1. Sulcos cerebrais fechados e ausência de atrofia giral.

2. Leve alargamento dos sulcos e atrofia discreta dos giros.

3. Alargamento moderado dos sulcos e atrofia giral moderada.

4. Alargamento acentuado dos sulcos e atrofia giral acentuada, com os giros apresentando um aspecto característico de "lâmina de faca".

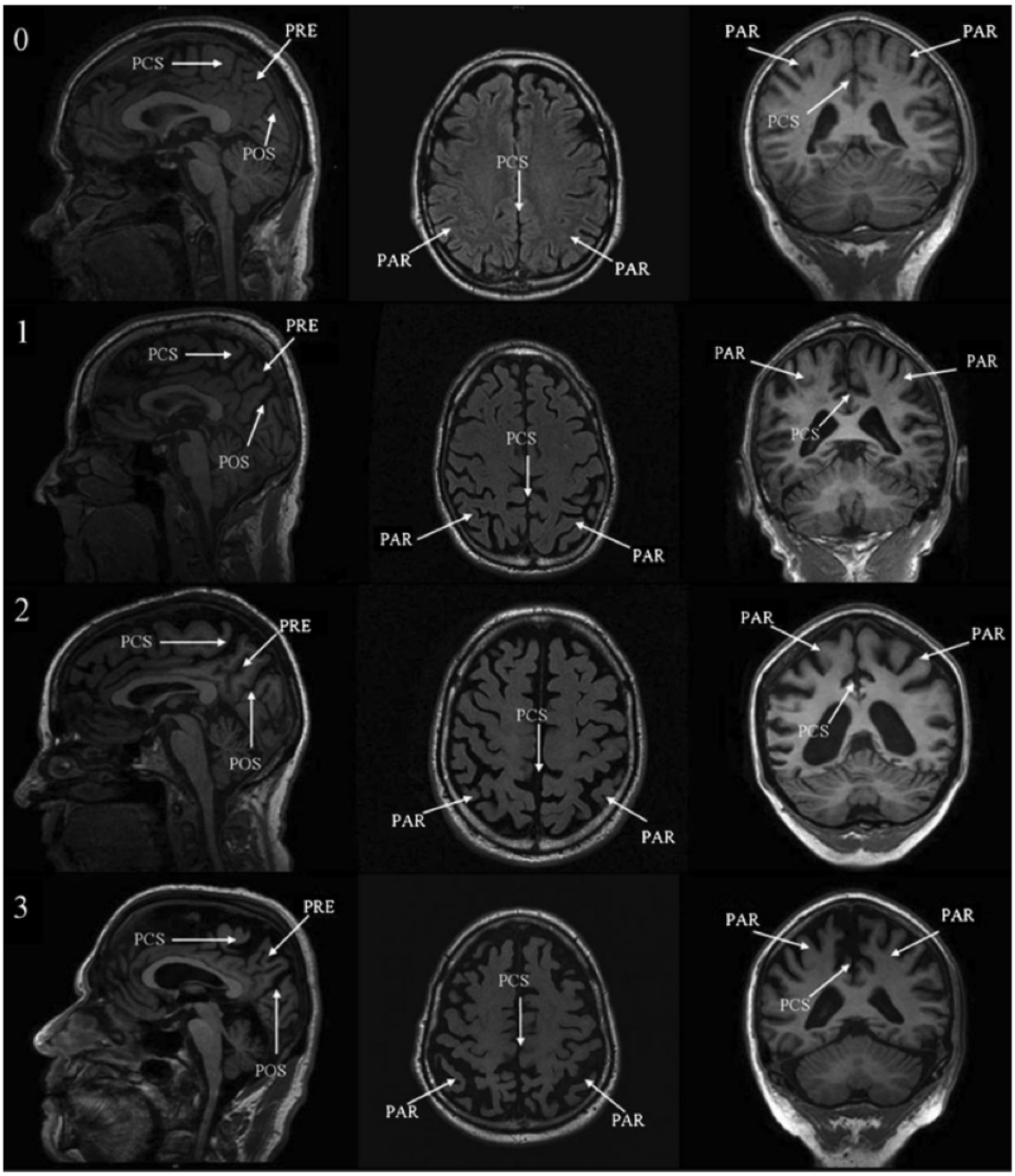


Figura 2.4 – Representação visual dos graus de atrofia posterior em diferentes cortes de MRI
Fonte: (KOEDAM et al., 2012)

2.3 Aprendizado profundo aplicado a processamento de imagens

Apesar do avanço notável nas técnicas de neuroimagem, como a MRI e a Tomografia por emissão de pósitrons (PET), para a identificação de características patológicas da AD, a interpretação visual e a quantificação manual das alterações estruturais e metabólicas ainda podem apresentar limitações quanto à objetividade, reprodutibilidade e escalabilidade. Diante desse cenário, a emergência de técnicas de aprendizado profundo (DL), um subcampo da IA, tem revolucionado o processamento e a análise de dados complexos, incluindo imagens. As redes neurais convolucionais (CNNs), em particular, demonstraram notável capacidade de aprender automaticamente características hierárquicas e padrões complexos diretamente a partir de grandes conjuntos de dados de imagens, superando abordagens tradicionais baseadas em extração manual de características (FUKUSHIMA; MIYAKE, 1982).

No cerne, o impacto transformador do DL reside em sua capacidade de automatizar tarefas complexas de análise de imagens. Isso inclui a detecção precisa de achados patológicos, a avaliação quantitativa da extensão da doença e a caracterização de várias patologias (ZHOU et al., 2021). Essa automação aborda diretamente o crescente "desafio da radiologia", uma questão crítica decorrente do volume cada vez maior de imagens médicas, juntamente com o tempo limitado dos clínicos, o que frequentemente leva a longos tempos de resposta. A descrição do desafio da radiologia como número crescente de imagens e tempo limitado, levando a longos tempos de resposta indica que o DL oferece mais do que apenas ganhos de eficiência (MACEDO; ROCHA, 2024).

Quando os radiologistas humanos são sobrecarregados pelo volume de trabalho e pelas restrições de tempo, seu desempenho cognitivo pode se degradar, aumentando a probabilidade de negligenciar detalhes patológicos sutis, mas críticos. As capacidades de automação do DL, ao lidar com o alto volume, não apenas aumentam a eficiência, mas também fornecem um motor analítico consistente e incansável. Essa consistência e a redução da carga cognitiva humana podem levar a uma melhor precisão e confiabilidade diagnóstica, mitigando efetivamente a fadiga. Portanto, o impacto do DL se estende além da mera eficiência operacional para uma melhoria significativa na qualidade e segurança gerais dos diagnósticos médicos (ZHOU et al., 2021).

2.4 Redes neurais convolucionais

As redes neurais convolucionais (CNNs) são arquiteturas de DL projetadas para processar dados com uma topologia de grade, como imagens. Sua eficácia reside na capacidade de aprender representações, isso significa que as camadas iniciais da rede detectam elementos visuais, como bordas, cores e texturas. À medida que os dados progridem através das camadas intermediárias, essas características de baixo nível são combinadas para formar padrões mais complexos, como formas e partes de objetos. Finalmente, as camadas mais profundas sintetizam essas representações para reconhecer objetos completos e até mesmo combinações de cenas.

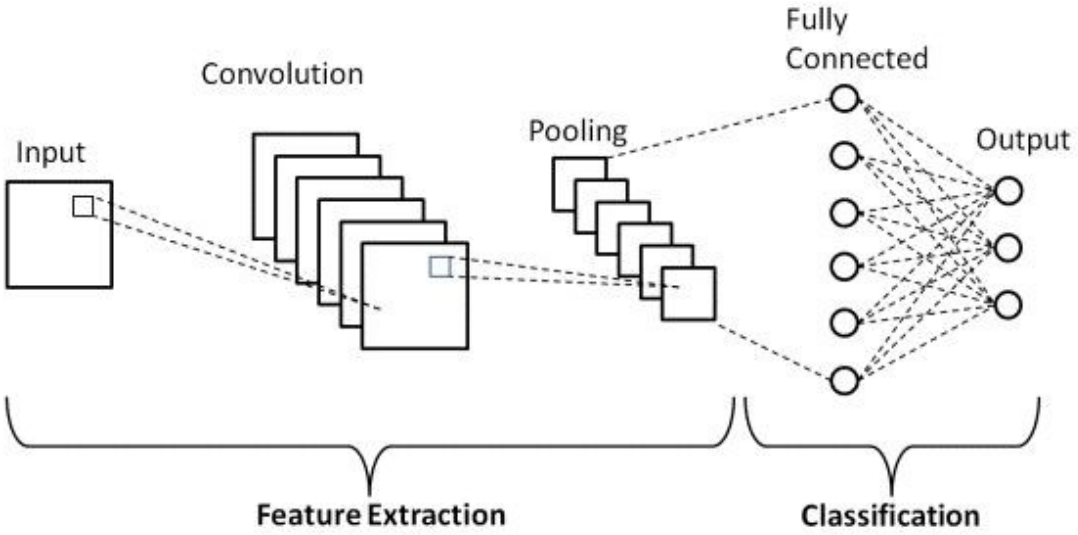


Figura 2.5 – Arquitetura da CNN
Fonte: (MHALANKAR; KORDE; KHATRI, 2024)

Cada camada encontra-se representada na Figura 2.5, onde cada componente é descrito como: Convolucionais (*Convolution*), as quais aplicam-se aos filtros, com o propósito de detectar padrões específicos, construindo recursos em cada camada. *Pooling*, as quais reduzem as dimensões espaciais da representação, aumentando a invariância a pequenas translações e minorando a complexidade computacional e, por fim, a camada totalmente conectada (*Fully Connected*), a qual se refere a uma camada onde cada neurônio recebe entrada de todos os neurônios da camada antecedente. Essa conexão completa entre as camadas possibilita que as redes aprendam relações complexas e eficientes (MHALANKAR; KORDE; KHATRI, 2024).

Arquiteturas avançadas de CNNs, como ResNet (Redes Residuais) e DenseNet (Redes Densas), em que suas conectividades são mostradas na Figura 2.6, têm sido amplamente empregadas e demonstraram desempenho superior em diversas tarefas de

classificação de imagens médicas. A ResNet, por exemplo, introduziu conexões de atalho (*skip connections*) que permitem que as redes sejam muito mais profundas (até 152 camadas) sem degradação do desempenho, facilitando o treinamento e o aprendizado de representações complexas (HE et al., 2016). A DenseNet, por sua vez, adota uma estratégia de concatenação densa, onde cada camada recebe as características de todas as camadas anteriores como entrada. Isso permite uma reutilização de características mais eficiente e um fluxo de informações aprimorado em toda a rede, embora possa resultar em maior consumo de memória e tempo de treinamento devido ao aumento dos canais de entrada e ao uso de muitas convoluções pequenas (HUANG; YANG; LEE, 2021). Ambas, ResNet e DenseNet, são frequentemente utilizadas como estruturas pré-treinadas em cenários distintos para classificação de AD vs CN, bem como para prever a conversão de MCI para AD, demonstrando resultados aprimorados (AGARWAL et al., 2021).

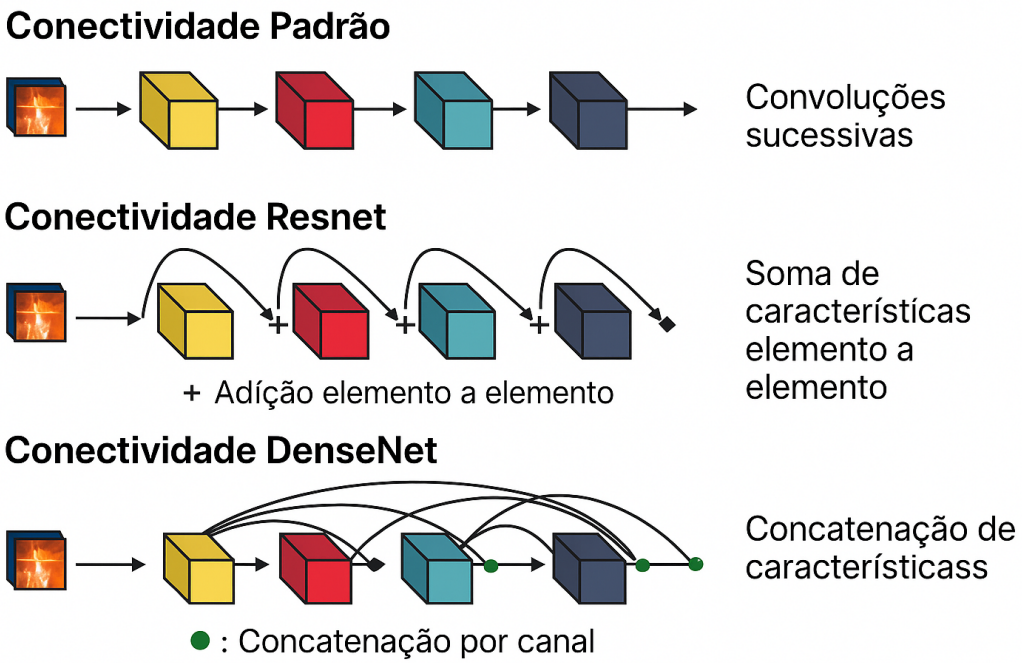


Figura 2.6 – Conectividade em redes convolucionais: Padrão, ResNet e DenseNet
Fonte: Adaptado de (HUANG; YANG; LEE, 2021) e (HE et al., 2016)

A ResNet foi introduzida como solução para o problema de desaparecimento de gradientes — que ocorre quando os sinais de erro retropropagados tornam-se infinitesimalmente pequenos ou excessivamente grandes, dificultando a convergência desde o início do treinamento — e para o problema da degradação do desempenho das camadas mais profundas. A ideia central é a adição de conexões de atalho que permitem que a entrada de uma camada seja adicionada diretamente à saída de uma camada anterior,

ignorando uma ou mais camadas. Matematicamente, em vez de aprender a função $H(x)$, a camada aprende uma função residual $F(x) = H(x) - x$. A saída da camada é então $F(x) + x$. Isso facilita o treinamento de redes mais profundas, já que as camadas que não precisam, necessariamente, aprender naquele instante, podem passar a entrada adiante e, por conseguinte, os gradientes podem fluir mais facilmente através da rede (HE et al., 2016).

Em contrapartida, ao invés de uma única conexão de atalho, a DenseNet implementa uma metodologia de concatenação densa, na qual cada camada recebe como entrada os atributos de todas as camadas precedentes. Consequentemente, a saída de cada camada é concatenada com as entradas das camadas subsequentes, fomentando a máxima reutilização de atributos e um fluxo de informações otimizado ao longo da rede. Tal conectividade densa auxilia na mitigação do problema do gradiente, na propagação de atributos e diminui a quantidade de parâmetros, visto que cada camada necessita aprender apenas um conjunto reduzido de novos mapas de atributos. Não obstante, a concatenação densa pode resultar em um elevado consumo de memória e em um maior tempo de treinamento, em decorrência do aumento dos canais de entrada e da utilização de múltiplas convoluções de pequena dimensão (HUANG; YANG; LEE, 2021).

No levantamento bibliográfico apresentado na seção 3 deste trabalho, detalhou-se a aplicabilidade das redes neurais no contexto da AD e sua eficácia na classificação de imagens.

2.5 Advento do aprendizado federado aplicado à saúde

O FL surge como uma metodologia promissora para a realização de treinamentos de modelos de IA em ambientes de dados distribuídos, oferecendo uma solução eficaz para a proteção da privacidade de informações sensíveis. Em um cenário como o da área da saúde, onde hospitais, clínicas e laboratórios lidam diariamente com dados extremamente confidenciais de pacientes, a privacidade se torna um ponto crítico durante a elaboração de um modelo (GUAN et al., 2024a).

Sendo uma abordagem de Aprendizado de Máquina tradicional (ML) na qual múltiplos clientes colaboram no treinamento de um modelo centralizado, preservando a descentralização de seus dados locais (KAIROUZ et al., 2021). No ML, o modelo fica em um servidor central e os dados são enviados a esse servidor para treinar um modelo global único, que é distribuído aos dispositivos de borda para realização de inferências.

Contudo, essa configuração apresenta duas limitações: (i) privacidade e segurança e (ii) falta de personalização.

O FL redefine o paradigma tradicional ao criar um esquema de treinamento híbrido, combinando as vantagens do modelo global central com a existência de modelos locais descentralizados, tendo assim, a solução para ambas limitações: (i) Os dados dos clientes permanecem em seus dispositivos locais e (ii) modelos locais são treinados com dados específicos, e suas contribuições são agregadas para formar um modelo global. Como

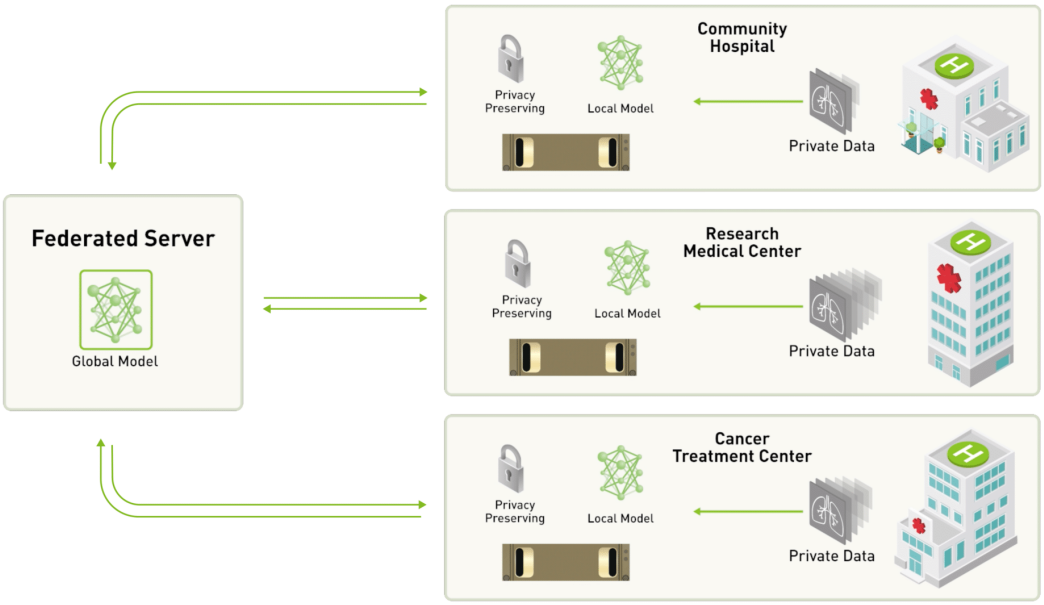


Figura 2.7 – Ciclo do aprendizado federado
Fonte: Adaptado de (HUANG; YANG; LEE, 2021) e (HE; GARCIA, 2009)

ilustrado na Figura 2.7, o processo começa com um modelo global em um servidor central e uma rodada de treinamento que segue estas etapas: um processo de seleção identifica um conjunto de clientes candidatos para participar da rodada; o modelo global é então transmitido aos dispositivos de borda dos clientes selecionados; cada cliente calcula uma atualização local do modelo usando exclusivamente seus dados locais, tipicamente via Gradiente Descendente (BANABILAH et al., 2022); o servidor central coleta essas atualizações e calcula uma atualização agregada do modelo, etapa que pode incorporar técnicas como agregação segura (para privacidade) ou compressão com perdas (para eficiência de comunicação) (KAIROUZ et al., 2021); finalmente, o servidor central atualiza o modelo global utilizando essa atualização agregada. Após múltiplas rodadas, o desempenho do modelo global é validado antes da implantação nos dispositivos dos clientes finais para inferência (WEI et al., 2020).

2.6 Algoritmos de agregação

Em sistemas distribuídos, a agregação é fundamental para gerar um modelo global eficiente a partir das atualizações de modelos locais. O processo consiste na combinação das atualizações recebidas dos diversos clientes pelo servidor central, sem que este acesse os dados brutos.

O principal algoritmo empregado é o Federated Averaging (FedAvg), proposto por McMahan. Neste método, as atualizações dos modelos locais de cada cliente são ponderadas conforme a quantidade de dados amostrais de cada um, e posteriormente, combinadas para gerar uma atualização global. Além disso, o FedAvg pode ser uma extensão do Descida de Gradiente Estocástico (SGD) (MCMAHAN et al., 2016).

O funcionamento pode ser descrito da seguinte forma: sendo o objetivo principal da FL minimizar uma função de perda global, $f(w)$, que é a média ponderada das funções de perdas locais $F_k(w)$ de cada cliente k (MCMAHAN et al., 2016).

Adotando que há K clientes e os dados de treinamento são particionados entre eles, temos que:

- w: Vetor de parâmetros do modelo de ML;
- n_k: Número de amostras de dados do cliente k;
- n: Número total de amostras de dados em todos os clientes, $n = \sum_{k=1}^{K} n_k$;
- $F_k(w)$: A função de perda média no conjunto de dados locais do cliente k. Se $f_i(w)$ é a perda para a amostra de dados i, então $F_k(w) = \frac{1}{n_k} \sum_{i \in P_k} f_i(w)$, onde P_k é o conjunto de índices de dados no cliente k.

Com isso, podemos definir a função de perda global $f(w)$:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \tag{2.1}$$

O objetivo é encontrar os parâmetros ótimos w^* que minimizem esta função:

$$w^* = \arg\min_{w} f(w) \tag{2.2}$$

Sendo FedAvg um algoritmo iterativo que opera em rodadas t de comunicação, podemos dividi-lo em algumas etapas, tendo a primeira como seleção e distribuição:

onde o servidor selecionará um subconjunto S_t aleatório de m clientes, então envia os parâmetros w_t do modelo global atual para cada cliente $k \in S_t$.

A segunda etapa é o treinamento local dos clientes: Cada cliente $k \in S_t$ atualiza os parâmetros do modelo recebido, w_t, usando seu conjunto de dados locais. O cliente executa E épocas de SGD em seus dados. Para uma única atualização local, usando um minibatch b de tamanho B, a regra de atualização é:

$$w \leftarrow w - \eta \nabla l(w; b) \tag{2.3}$$

onde:

- η é a taxa de aprendizagem.
- $\nabla l(w; b)$ é o gradiente da função de perda calculado no minibatch.

Após E épocas de iterações sobre os minibatches de seus dados locais, o cliente k obtém um novo vetor de parâmetros, w_{t+1}^k. Este vetor é então enviado de volta para o servidor.

Por fim, na etapa de Agregação no Servidor, ele aguarda para receber os parâmetros atualizados w_{t+1}^k de todos os clientes no subconjunto S_t. Em seguida, ele agrega esses parâmetros para calcular o novo modelo global, w_{t+1}, através de uma média ponderada:

$$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{\sum_{j \in S_t} n_j} w_{t+1}^k \tag{2.4}$$

Este novo modelo global w_{t+1} é então usado como ponto de partida para a próxima rodada de comunicação $t + 1$, e o processo se repete até que o modelo convirja.

Apesar de o FedAvg ser amplamente reconhecido, demonstra uma queda de performance em distribuições de dados não-IID.. Para solucionar essa questão, Hsu, Qi e Brown (2019) desenvolveram uma estratégia para mitigar essa perda de desempenho, chamada de *Federated Averaging with Server Momentum* (FedAvgM).

Enquanto o FedAvg padrão atualiza os pesos do modelo diretamente com a média ponderada das atualizações dos clientes, o FedAvgM introduz uma variável de velocidade v que acumula um histórico de atualizações. Assim como no FedAvg (Eq. 2.4), o servidor primeiro calcula a atualização média ponderada dos clientes, Δt e a velocidade v é atualizada para incorporar a nova atualização e manter uma fração β de seu valor anterior:

$$v_{t+1} \leftarrow \beta v_t + \Delta t \tag{2.5}$$

Finalmente, o modelo global w é atualizado usando a velocidade v:

$$w_{t+1} \leftarrow w_t + v_{t+1} \tag{2.6}$$

Paralelamente, Li et al. (2020) propuseram o *Federated Proximal Optimization* (FedProx), um framework desenhado para abordar os desafios da heterogeneidade em FL. O método lida com duas vertentes principais: a heterogeneidade de sistemas (variação nas capacidades de hardware como CPU, memória, conexão de rede e bateria) e a heterogeneidade estatística (dados non-IID). A principal inovação do FedProx é a introdução de um termo de regularização proximal na função de perda local, que limita a divergência dos atualizações locais em relação ao modelo global, garantindo convergência mais estável mesmo com dispositivos instáveis (*stragglers*).

O FedProx é apresentado como uma generalização e parametrização do FedAvg, fazendo pequenas modificações no método, sendo a principal a adição de um termo de proximidade à função de perda local de cada cliente. Em vez de apenas minimizar sua função de perda local $F_k(w)$, cada cliente k selecionado em uma rodada t trabalha para minimizar aproximadamente um novo objetivo, h_k:

$$h_k(w; w_t) = F_k(w) + \frac{\mu}{2}\|w - w_t\|^2 \tag{2.7}$$

Onde:

- w_t são os pesos do modelo global no início da rodada.
- μ é um hiperparâmetro que controla o peso do termo de proximidade.

O termo de proximidade arrasta a solução local para mais perto do modelo global inicial w_t.

2.7 Considerações finais

Este capítulo apresentou conceitos fundamentais que servem como alicerce para este trabalho. Foram abordados os temas de *Deep Learning* e *Federated Learning*, bem como sua aplicação conjunta no processamento de imagens. O objetivo foi proporcionar ao leitor uma compreensão abrangente de suas definições, características e aplicações. No próximo capítulo, serão explorados trabalhos relacionados que exemplificam o uso desses modelos.

3 TRABALHOS RELACIONADOS

Há diversos estudos dedicados ao avanço na classificação da AD. A maioria dessas pesquisas fundamenta-se no uso de redes neurais. Portanto, neste contexto, será fornecida uma visão abrangente dessa abordagem ao elaborar uma revisão sistemática da literatura. Posteriormente, serão realizadas comparações entre algumas implementações utilizando a técnica de DL e FL.

3.1 Revisão Sistemática da Literatura

Visando preencher lacunas na literatura sobre a aplicação de técnicas de FL no diagnóstico da AD, foi conduzida uma revisão sistemática da literatura (RSL) a partir da análise de uma série de publicações científicas relacionadas ao tema. A revisão concentrou-se na investigação de modelos de FL aplicados ao processamento e classificação de imagens médicas em sistemas distribuídos, especialmente em cenários com dados heterogêneos (non-IID).

A estrutura desta RSL contempla a definição da metodologia, a formulação de questões de pesquisa e a descrição da estratégia de busca, além da apresentação dos resultados e da análise comparativa entre os modelos propostos nos estudos selecionados. O desenvolvimento desta revisão seguiu as diretrizes metodológicas propostas por Kitchenham e Charters (2007), amplamente reconhecidas na área de engenharia de software e ciência de dados aplicada.

A revisão foi conduzida em conformidade com um protocolo baseado em mapeamento sistemático, estruturado em três fases principais: (1) planejamento, (2) condução e (3) apresentação dos resultados. A fase de planejamento abrange: (i) a identificação da necessidade da revisão, (ii) a definição de questões de pesquisa específicas e (iii) a elaboração de um protocolo de revisão.

A partir da formulação das perguntas de pesquisa e da definição das *strings* de busca, foram realizadas pesquisas nas bases bibliográficas detalhadas na Seção 3.1.3. Inicialmente, foram recuperados 328 registros relacionados ao tema. Após a remoção de duplicatas e a aplicação dos critérios de inclusão e exclusão — que consideraram, entre outros fatores, a presença de experimentos quantitativos com dados (públicos, como ADNI e OASIS, ou clínicos), o uso de arquiteturas baseadas em DL e o foco específico na AD —, foram selecionadas 23 publicações que compuseram o conjunto final da análise. A Figura 3.1 ilustra o fluxo seguido na construção da RSL. Esses estudos forneceram as

informações necessárias para responder às questões de pesquisa previamente definidas e possibilitaram uma análise crítica das abordagens mais relevantes em termos de redes neurais, métodos de agregação e configurações experimentais.

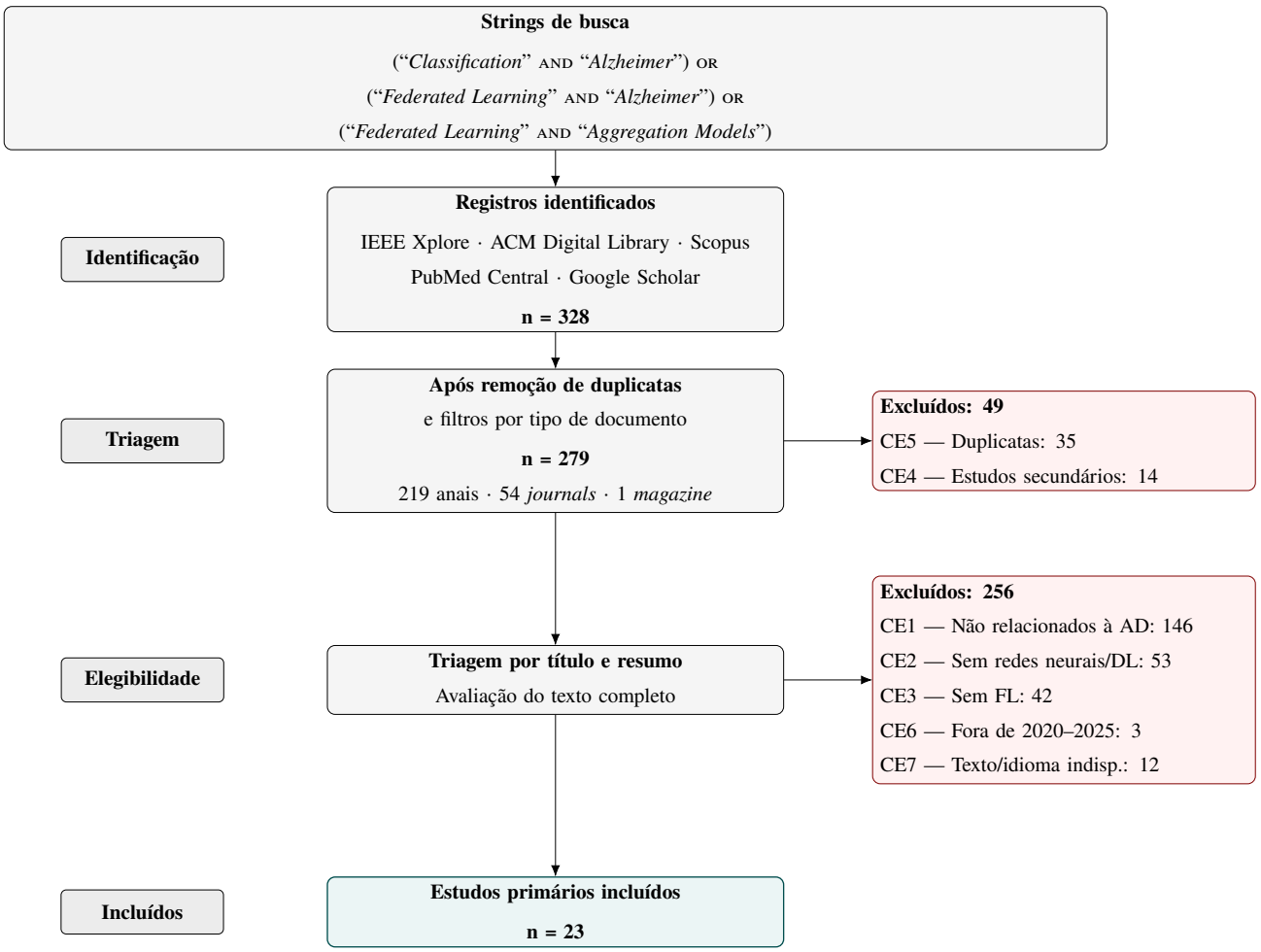


Figura 3.1 – Fluxograma da RSL no padrão PRISMA.

Fonte: Autor, 2025

3.1.1 Objetivos

A implementação do FL em ambientes clínicos tem se mostrado promissora para a construção de modelos descentralizados, sem a necessidade de compartilhamento de dados sensíveis. Dentre as aplicações, os estudos com imagens de ressonância magnética com o foco na AD destacam-se pela relevância clínica e pelos desafios associados à heterogeneidade dos dados.

Esta revisão sistemática tem como objetivo principal a identificação dos desafios técnicos recorrentes, das arquiteturas de redes neurais e dos modelos de agregação mais utilizados entre servidor e clientes em contextos de FL aplicados à AD. Além disso, busca-

se promover uma análise comparativa que permita compreender as decisões metodológicas predominantes e sintetizar recomendações:

- Avaliar a eficiência de diferentes arquiteturas de redes neurais (ex.: ResNet, DenseNet, MLP, etc..) no contexto DA em ambientes federados;
- Analisar comparativamente a eficácia dos algoritmos de agregação em cenários da AD;
- Mapear as métricas de avaliação empregadas nos estudos, com foco em indicadores de desempenho.

3.1.2 Questões

Para preencher a lacuna de conhecimento sobre a existência e os limites de tal abordagem, foram elaboradas perguntas com o intuito de investigar aspectos técnicos e metodológicos fundamentais para a aplicação de FL em contexto da AD. A Tabela 3.1 relaciona os objetivos com a questão de pesquisa.

Q1: Quais arquiteturas de redes neurais são mais utilizadas para a tarefa de classificação de imagens médicas no contexto da AD em ambientes de FL?

Q2: Qual é o impacto da escolha de diferentes métodos de agregação no desempenho de modelos de FL em cenários com dados heterogêneos?

Q3: Quais estratégias de configuração experimental são mais comuns no treinamento de modelos de FL aplicados à classificação de imagens médicas?

Q1 identifica:

- As arquiteturas de redes neurais convolucionais mais recorrentes (como ResNet, DenseNet, EfficientNet, MLP, entre outras) e suas respectivas metodologias de implementação nos estudos analisados.

Q2 identifica:

- Os principais algoritmos de agregação utilizados (como FedAvg, FedProx, FedSGD, FedCM, entre outros) e sua aplicação em estudos voltados à detecção da AD.

- A influência dessas abordagens no desempenho dos modelos quando submetidos à variação estatística e estrutural típica dos dados clínicos distribuídos.

Q3 identifica:

- Parâmetros recorrentes nos experimentos, como número de clientes, tamanho do *batch*, taxa de aprendizado e número de épocas.

Tabela 3.1 – Relação entre as questões e os objetivos da RSL

Objetivos da RSL	**Questão de Pesquisa Correspondente**	**Justificativa**
1. Identificar como as arquiteturas de redes neurais (ResNet, DenseNet, MLP, etc.) são eficientes em *Federated Learning* para a classificação de imagens médicas.	Q1	A questão busca identificar quais arquiteturas são mais comuns, enquanto o objetivo visa avaliar a eficiência dessas mesmas arquiteturas.
2. Analisar comparativamente a eficácia dos algoritmos de agregação em cenários da AD.	Q2	A questão investiga o impacto dos métodos de agregação, e o objetivo se propõe a comparar a eficácia desses métodos para entender tal impacto. Ambos focam no mesmo desafio central: o desempenho em dados heterogêneos.
3. Mapear as métricas de avaliação empregadas nos estudos, com foco em indicadores de desempenho.	Q3	A análise das métricas é um componente essencial para entender as estratégias de configuração experimental.

Fonte: Autor, 2025

3.1.3 Estratégia

O processo de revisão foi conduzido com base nas diretrizes de Kitchenham e Charters (2007) e envolveu a definição de palavras-chave e o uso dos operadores booleanos *AND* e *OR* para recuperar publicações relevantes. Foram utilizadas as seguintes palavras-chave:

(“*Classification AND Alzheimer*") OR
(“*Federated learning*” AND “*Alzheimer*”) OR
(“*Federated learning*” AND “*Aggregation Models*")

As *strings* de busca foram adaptadas à sintaxe de cada motor e aplicadas em cinco bases bibliográficas: IEEE Xplore, ACM Digital Library, Scopus, PubMed Central (PMC) e Google Scholar. A seleção priorizou bases que indexam de forma abrangente as principais editoras da área — entre elas IEEE, ACM, Springer e Elsevier — de modo que essas editoras são alcançadas *por meio* dos índices consultados, e não pesquisadas isoladamente. Inicialmente, foram recuperados 328 registros, utilizando combinações de palavras-chave como “*Federated Learning*”, “*Alzheimer's Disease*” e “*Classification*”, ajustadas conforme os requisitos de cada motor de busca.

3.1.4 Critérios de inclusão e exclusão

Como critério de filtragem temporal, foram considerados artigos publicados entre 2020 e 2025, refletindo a produção científica recente no campo. Os critérios de inclusão (CI) e de exclusão (CE) adotados na triagem estão na Tabela 3.2.

Dos 328 registros inicialmente recuperados, a remoção de duplicatas e a triagem por tipo de documento reduziram o conjunto a 279 estudos - 219 provenientes de anais de conferência, 54 de *journals* e 1 de *magazine*. Em seguida, as etapas de triagem por título e resumo e de avaliação do texto completo aplicaram os critérios de exclusão, resultando nos 23 estudos primários incluídos.

Tabela 3.2 – Critérios de inclusão (CI) e exclusão (CE) adotados na RSL.

ID	Descrição
Critérios de inclusão	
CI1	Estudos primários que aplicam FL (isoladamente ou combinado a SL) à classificação ou detecção da AD.
CI2	Estudos que reportam experimentos quantitativos com dados de imagem ou biomarcadores.
CI3	Estudos publicados entre 2020 e 2025.
CI4	Estudos com texto completo disponível, em inglês ou português.
Critérios de exclusão	
CE1	Estudos não relacionados à AD.
CE2	Estudos que não empregam redes neurais ou aprendizado profundo.
CE3	Estudos que não empregam aprendizado federado ou distribuído.
CE4	Estudos secundários (revisões, *surveys*, editoriais) ou sem avaliação experimental.
CE5	Registros duplicados.
CE6	Estudos fora do recorte temporal (2020–2025).
CE7	Estudos com texto completo indisponível ou em idioma fora do escopo.

Fonte: Autor, 2025

Tabela 3.3 – Publicações selecionadas para a Revisão Sistemática da Literatura (RSL).

Ano	Autores	Título	Arquiteturas	Modelo FL
2025	BASNIN et al.	An Evolutionary Federated Learning Approach to Diagnose Alzheimer's Disease Under Uncertainty.	CNN	FedAvg & FedProx
2025	SONG et al.	Harmonized Swarm Learning With Guided Optimization for Multi-Center sMRI Classification of Alzheimer's Disease.	CNN	Swarm (HSGO)
2024	LAKHAN et al.	FDCNN-AS: Federated deep convolutional neural network Alzheimer detection schemes for different age groups.	Deep CNN	FedAvg

(*continua na próxima página*)

Tabela 3.3 – – (continuação)

Ano	Autores	Título	Arquiteturas	Modelo FL
2024	KAPOOR et al.	FedCL-Ensemble Learning: A Framework of Federated Continual Learning with Ensemble Transfer Learning Enhanced for Alzheimer's MRI Classifications.	ResNet, Inception Ensemble	FedAvg
2024	CASTRO; IMPEDOVO; PIRLO	A Federated Learning System with Biometric Medical Image Authentication for Alzheimer's Diagnosis.	CNN	FedAvg
2024	RAMESH et al.	A Federated Learning Paradigm for Early Alzheimer's Detection using MRI Imaging.	Xception, GhostNet	FedAvg
2024	IMBOCCIOLI; CIALONE; FERRETTI	Decentralization of Learning and Trust in the Healthcare: Blockchain-driven Federated Learning for Alzheimer's MRI Image Classification.	CNN	FedAvg (Blockchain)
2024	SRINIVASU et al.	Enhancing medical image classification via federated learning and pre-trained model.	EfficientNet	FedAvg
2024	ROSHAN; ANITHA; JULIET	Federated Learning for Alzheimer's Diagnosis: Leveraging MRI Data.	ResNet, VGG, Inception	FedAvg

(*continua na próxima página*)

Tabela 3.3 – – (continuação)

Ano	Autores	Título	Arquiteturas	Modelo FL
2024	TRIVEDI; JAIN; AGARWAL	Identifying and Categorizing Alzheimer's Disease with Lightweight Federated Learning Using Identically Distributed Images.	ResNet, VGG	FedAvg
2024	NIMESHIKA; SUBITHA	Enhancing Alzheimer's disease classification through split federated learning and GANs for imbalanced datasets.	ResNet (SplitFL)	SplitFed
2024	OUYANG et al.	ADMarker: A Multi-Modal Federated Learning System for Monitoring Digital Biomarkers of Alzheimer's Disease.	MLP, CNN (multimodal)	FedAvg
2024	SAHID et al.	Enhancing Parallelism in Cross-silo Federated Learning for Brain Disease Classification.	CNN	FedAvg
2024	LEI et al.	Hybrid federated learning with brain-region attention network for multi-center Alzheimer's disease detection.	ResNet	FedAvg
2023	CASELLA; FONIO	Architecture-Based FedAvg for Vertical Federated Learning.	ResNet, MLP, CNN	FedAvg
2023	KHALIL et al.	A Federated Learning Model Based on Hardware Acceleration for the Early Detection of Alzheimer's Disease.	MLP	FedAvg

(*continua na próxima página*)

Tabela 3.3 – – (continuação)

Ano	Autores	Título	Arquiteturas	Modelo FL
2023	ELSERSY et al.	Federated Learning Model for Early Detection of Dementia Using Blood Biosamples.	MLP	FedAvg
2023	GHOSH; KRISHNAMOORTHY	DifEvoDenseFed: Differential Evolution Optimization Based DenseNet Federated Model in Alzheimer's Detection.	DenseNet	FedAvg (evo)
2023	EOM et al.	Federated Learning in Prediction of Dementia Stage: An Experimental Study.	VGG	FedAvg
2023	SRINIDHI; VICTOR	A Federated Deep Learning Method for Chronic Disease Diagnosis.	ResNet	FedAvg
2023	WEI et al.	FedCPC: An Effective Federated Contrastive Learning Method for Privacy Preserving Early-Stage Alzheimer's Speech Detection.	CNN, LSTM	FedAvg
2021	HUANG; YANG; LEE	Federated Learning via Conditional Mutual Learning for Alzheimer's Disease Classification on T1w MRI.	3D CNN (VGG-like)	FedCM
2021	QU; BALACHANDAR; RUBIN	An experimental study of data heterogeneity in federated learning methods for medical imaging.	ResNet	FedAvg vs FedSGD

Fonte: Autor, 2025

3.1.5 Resultados

Este tópico apresenta, de forma estruturada, os achados extraídos dos 23 estudos primários, respondendo às três questões de pesquisa (Q1, Q2 e Q3) relacionadas à aplicação do FL à AD. A Tabela 3.4 apresenta todas as publicações e suas características.

Inicialmente, constatou-se que as CNNs dominam a literatura, sobretudo variantes da ResNet. No trabalho HFL-BANet, (LEI et al., 2024) implementaram uma ResNet com mecanismo de atenção, obtendo acurácia média de 85,7% para a tarefa de classificação (AD vs. CN) em cinco centros clínicos. Modelo alternativo, criado por Ghosh e Krishnamoorthy (2023), integra uma DenseNet a um algoritmo de evolução diferencial que ajusta hiperparâmetros e pondera os filtros convolucionais antes da etapa de agregação, isto resultou numa acurácia de 94%. Adicionalmente, Huang, Yang e Lee (2021) demonstraram que 3D-CNNs oferecem melhor desempenho em imagens de ressonância magnética, reforçando sua pertinência quando o alvo são volumes inteiros de ressonância magnética.

Como a designação "CNN" aparece de forma recorrente na Tabela 3.4, convém detalhar a natureza dessas arquiteturas. É possível distinguir quatro grupos. O primeiro reúne estudos que adotam *backbones* consagrados — como ResNet, VGG, DenseNet, Inception, Xception, GhostNet e EfficientNet —, frequentemente por meio de aprendizado por transferência (KAPOOR et al., 2024; ROSHAN; ANITHA; JULIET, 2024; TRIVEDI; JAIN; AGARWAL, 2024; RAMESH et al., 2024; SRINIVASU et al., 2024; GHOSH; KRISHNAMOORTHY, 2023). O segundo emprega CNNs volumétricas (3D), que processam volumes inteiros de MRI e preservam a informação interfatias (HUANG; YANG; LEE, 2021). O terceiro é composto por CNNs customizadas, descritas pelos autores apenas como "CNN", sem um *backbone* nomeado (BASNIN et al., 2025; CASTRO; IMPEDOVO; PIRLO, 2024; IMBOCCIOLI; CIALONE; FERRETTI, 2024; SAHID et al., 2024; SONG et al., 2025); nesses casos, o rótulo genérico reflete a ausência de uma arquitetura padronizada no estudo de origem, e não uma simplificação desta revisão. O quarto grupo abrange arquiteturas híbridas e multimodais, que combinam CNN com módulos recorrentes para dados sequenciais, como a detecção de AD a partir de fala (CNN + LSTM) (WEI et al., 2023), ou que associam CNN e MLP em cenários multimodais (OUYANG et al., 2024). Essa estratificação evidencia que a predominância de CNNs na literatura não implica homogeneidade arquitetural: coexistem desde redes rasas customizadas até *backbones* profundos pré-treinados e modelos volumétricos.

No que concerne aos métodos de agregação, FedAvg permanece referência em 83% dos estudos; contudo, abordagens que integram regularização têm mostrado ganhos consistentes. Basnin et al. (2025) compararam três agregadores e verificaram que FedProx, com seu termo proximal, pode aumentar o desempenho e reduzir a variabilidade entre clientes. Uma alternativa criada por Imboccioli, Cialone e Ferretti (2024) foi utilizar *blockchain*, com o intuito de coordenar a agregação no servidor e se mostrou condizente com os resultados comparativos a outros modelos sem o uso dessa técnica.

Quanto aos parâmetros experimentais, verifica-se convergência em *batch size* de 32–64 e ciclos locais curtos ($\leq$ 4 épocas), estratégia adotada por Lei et al. (2024), Basnin et al. (2025) e Ouyang et al. (2024) para controlar a divergência dos modelos locais. O otimizador Adam é predominante, evidenciando a busca por estabilidade dos gradientes em conjuntos de dados limitados por cliente. A tabela 3.4 compila todas as características dos artigos analisados.

Esta revisão sistemática forneceu uma visão geral comparativa do estado da arte. A partir dela, foram identificadas lacunas de pesquisa que motivam e posicionam o presente trabalho, discutidas em detalhe na Seção 3.1.6.

Tabela 3.4 – Artigos e suas características.

Ano	Autores	Rede Neural	Modelo FL	Dataset	Métrica	Clientes	Batch	Otimizador
2025	Basnin et al. (2025)	CNN	FedAvg & FedProx	ADNI	ACC	6	32	Adam
2025	Song et al. (2025)	CNN	Swarm (HSGO)	ADNI + Local	AUC, ACC	10	32	Adam
2024	Lakhan et al. (2024)	Deep CNN	FedAvg	ADNI (age-strat.)	ACC	4	32	Adam
2024	Kapoor et al. (2024)	ResNet, Inception Ensemble	FedAvg	ADNI	ACC	5	32	Adam
2024	Castro, Impedovo e Pirlo (2024)	CNN	FedAvg	OASIS	ACC	3	32	Adam
2024	Ramesh et al. (2024)	Xception + GhostNet	FedAvg	ADNI	ACC	5	64	SGD
2024	Imboccioli, Cialone e Ferretti (2024)	CNN	FedAvg (Blockchain)	ADNI	AUC, ACC	3	32	Adam

(*continua na próxima página*)

Tabela 3.4 – – (continuação)

Ano	Autores	Rede Neural	Modelo FL	Dataset	Métrica	Clientes	Batch	Otimizador
2024	Srinivasu et al. (2024)	EfficientNet	FedAvg	Private MRI	ACC	4	32	Adam
2024	Roshan, Anitha e Juliet (2024)	ResNet, VGG, Inception	FedAvg	ADNI	ACC	5	16	Adam
2024	Trivedi, Jain e Agarwal (2024)	ResNet, VGG	FedAvg	IID synthetic	ACC	2	128	SGD
2024	Nimeshika e Subitha (2024)	ResNet (Split-FL)	SplitFed	ADNI	ACC	4	32	Adam
2024	Ouyang et al. (2024)	MLP, CNN (multimodal)	FedAvg	Wearables + MRI	ACC	6	32	Adam
2024	Sahid et al. (2024)	CNN	FedAvg	Private MRI	ACC	8	64	Adam
2024	Lei et al. (2024)	ResNet + Attention	FedAvg	ADNI + OASIS	AUC, ACC	5	32	Adam
2023	Casella e Fonio (2023)	ResNet, MLP, CNN	FedAvg (vertical)	Multi-center	ACC	3	16	SGD

(*continua na próxima página*)

Tabela 3.4 – – (continuação)

Ano	Autores	Rede Neural	Modelo FL	Dataset	Métrica	Clientes	Batch	Otimizador
2023	Khalil et al. (2023)	MLP	FedAvg	Blood biosamples	ACC	5	32	Adam
2023	Elsersy et al. (2023)	MLP	FedAvg	Blood biosamples	ACC	4	32	Adam
2023	Ghosh e Krishnamoorthy (2023)	DenseNet	FedAvg (evo)	ADNI	ACC	4	32	Adam
2023	Eom et al. (2023)	VGG	FedAvg	ADNI	ACC	5	16	Adam
2023	Srinidhi e Victor (2023)	ResNet	FedAvg	ADNI	ACC	4	32	Adam
2023	Wei et al. (2023)	CNN, LSTM	FedAvg	Speech corpus	ACC, F1-score	6	32	Adam
2021	Huang, Yang e Lee (2021)	3D CNN (VGG-like)	FedCM	ADNI	AUC, ACC	4	16	Adam

(*continua na próxima página*)

Tabela 3.4 – – (continuação)

Ano	Autores	Rede Neural	Modelo FL	Dataset	Métrica	Clientes	Batch	Otimizador
2021	Qu, Balachandar e Rubin (2021)	ResNet	FedAvg vs FedSGD	ADNI	ACC, F1-score	-	16	SGD

3.1.6 Lacunas de Pesquisa e Oportunidades

A análise dos 23 estudos primários, sintetizada na Tabela 3.4, revela um conjunto de lacunas recorrentes na intersecção entre FL, aprendizado profundo e classificação da AD. Tais lacunas são discutidas a seguir e relacionadas às contribuições desta dissertação.

3.1.6.1 Ausência de diretrizes padronizadas para o treinamento.

A maioria dos estudos seleciona a arquitetura neural e o algoritmo de agregação de maneira empírica, sem justificativa sistemática. O FedAvg figura como opção predominante (presente em cerca de 83% dos trabalhos analisados), frequentemente por ser o agregador de referência, e não por uma comparação controlada com alternativas. Essa ausência de diretrizes dificulta a reprodutibilidade e a transferência de resultados entre estudos. Esta dissertação aborda diretamente a lacuna ao comparar, de forma controlada, três agregadores (FedAvg, FedAvgM e FedProx) combinados a duas arquiteturas (ResNet-50 e DenseNet-169) sob diferentes números de clientes.

3.1.6.2 Comparação sistemática de agregadores sob heterogeneidade.

Poucos estudos variam o algoritmo de agregação e quantificam seu efeito sobre a estabilidade do treinamento em cenários non-IID. Em particular, o comportamento de agregadores baseados em *momentum*, como o FedAvgM, permanece pouco investigado nesse contexto.

3.1.6.3 Escassez de avaliações de arquiteturas híbridas e do efeito do volume de dados.

A grande maioria dos trabalhos adota FL "puro", sendo rara a integração com SL e escassa a análise sistemática do impacto do volume de dados sobre a escolha arquitetural. Esta dissertação investiga um arranjo híbrido FL+SL e avalia, em duas fases com volumes distintos de dados, como o aumento da amostra altera a sensibilidade à arquitetura escolhida.

3.1.6.4 Heterogeneidade de protocolos e métricas.

Observa-se ampla variação no número de clientes, no *batch size* e nas métricas reportadas (com predomínio da acurácia isolada), o que compromete a comparabilidade

entre estudos, sobretudo em cenários com classes desbalanceadas. Este trabalho adota um protocolo experimental explícito e reporta, além da acurácia, métricas de *recall* e precisão, mais sensíveis ao desbalanceamento típico dos dados clínicos.

3.1.6.5 Integração multimodal e garantias de privacidade.

Apenas poucos estudos exploram a fusão multimodal em FL (OUYANG et al., 2024) ou mecanismos avançados de privacidade, como a agregação segura (MITROVSKA et al., 2024). Embora fora do escopo principal desta dissertação, esses aspectos constituem direções relevantes para trabalhos futuros.

3.2 Considerações finais

Neste capítulo, uma análise aprofundada do estado da arte foi realizada através de uma revisão sistemática da literatura. A partir desta revisão, obteve-se um panorama dos estudos que abordam a interseção entre a AD, processamento de imagens médicas e métodos distribuídos. Uma lacuna proeminente identificada é a falta de diretrizes padronizadas para o treinamento de modelos de DL (especificamente CNNs) em ambientes federados, resultando em uma seleção frequentemente empírica de algoritmos de agregação e arquiteturas neurais. Diante disso, o presente trabalho propõe-se a investigar e avaliar diferentes configurações para o treinamento distribuído, integrando o FL e o SL na classificação de estágios da AD em imagens de MRI. O modelo proposto será descrita no capítulo seguinte.

4 MODELO

Este capítulo descreve a organização e a arquitetura do modelo desenvolvido para o treinamento das redes neurais. O foco principal é detalhar como a estrutura foi projetada para lidar com a classificação dos estágios da AD a partir de imagens médicas, equilibrando a necessidade de alto desempenho computacional com a garantia de privacidade dos dados dos pacientes. Nas seções a seguir, serão apresentados a visão geral da topologia adotada, os papéis e responsabilidades do cliente e do servidor central, bem como o fluxo completo de comunicação, treinamento e agregação dos parâmetros.A implementação de toda a arquitetura utilizou a linguagem Python, apoiada por bibliotecas especializadas como PyTorch[1], Torchvision e Scikit-learn[2]. O detalhamento em código da adaptação deste modelo, evidenciando a lógica de separação entre cliente e servidor, encontra-se disponibilizado na íntegra no Apêndice A.

4.1 Visão Geral da Arquitetura Proposta

Devido à complexidade das CNNs utilizadas no processamento de imagens médicas, exigir que cada cliente treine um modelo completo localmente demandaria recursos computacionais de alto custo, frequentemente indisponíveis em instituições hospitalares menores.

Para contornar essa limitação e viabilizar o treinamento, o modelo de Aprendizado Federado Dividido (SFL - *Split Federated Learning*), proposto por Thapa et al. (2022), introduz a figura de um *fed server* (servidor federado), que atua como o nó central responsável por orquestrar a agregação dos parâmetros dos modelos locais. Nessa arquitetura, a rede neural é dividida em um ponto específico denominado *cut-layer* (camada de corte). Durante o treinamento, cada cliente executa o *forward pass* (propagação para frente) — processo em que os dados de entrada atravessam as camadas iniciais da rede — apenas até atingir essa camada de corte. Com isso, o cliente preserva a privacidade de seus dados brutos e retém apenas os pesos da sua respectiva sub-rede, enviando as ativações intermediárias ao servidor, que processa as camadas finais até a etapa de classificação, conforme ilustrado na Figura 4.1.

[1] https://pytorch.org/
[2] https://scikit-learn.org/

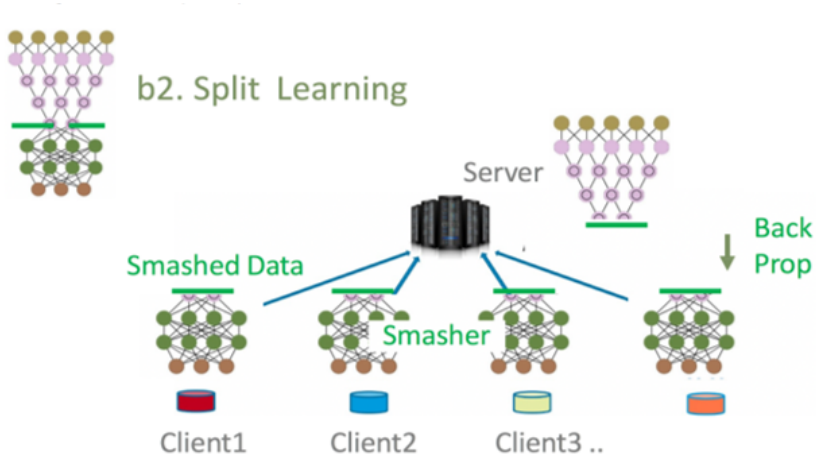


Figura 4.1 – Aprendizado dividido

Fonte: (VEPAKOMMA et al., 2018)

De forma macro, o ciclo de treinamento colaborativo ocorre seguindo os seguintes passos:

1. Inicialização e Posse de Dados: Cada cliente possui seu próprio conjunto de dados locais de imagens de ressonância magnética particionadas de forma não homogênea (non-IID), garantindo que os dados nunca saiam de suas origens;

2. Processamento Local: O cliente inicia o treinamento processando as imagens através das camadas iniciais da rede neural, extraindo as características visuais mais básicas. O resultado desse processamento é um mapa de características intermediário;

3. Comunicação Intermediária: O cliente envia este mapa de características ao servidor. Como os dados brutos não são transmitidos, a privacidade do paciente é preservada;

4. Processamento Central: O servidor recebe os dados intermediários, passa-os pelas camadas finais da rede neural, realiza a classificação da doença e calcula a margem de erro. Em seguida, o servidor calcula os gradientes (direções de ajuste) e os envia de volta ao cliente;

5. Atualização e Agregação: Após múltiplas iterações desse processo, os clientes enviam os pesos atualizados de suas sub-redes locais para o servidor. O servidor, então, atua como um orquestrador, agregando todos os pesos recebidos utilizando algoritmos específicos para criar um modelo global mais inteligente e abrangente. Esse ciclo se repete por diversas rodadas de comunicação até que o modelo global alcance a convergência.

4.2 Cliente

O objetivo do cliente no modelo[3] é viabilizar treinamento colaborativo garantindo que os dados brutos permaneçam no dispositivo local. A lógica do cliente gerencia o acesso aos dados, a execução das rodadas de treinamento local e os pesos atualizados ao servidor.

4.2.1 Particionamento de Dados

Para os experimentos reportados, adotou-se uma estratégia de particionamento non-IID baseada em pacientes entre os N clientes. A distribuição dos pacientes foi realizada de forma exclusiva entre clientes utilizando estratégia *round-robin* por classe, garantindo três propriedades fundamentais:

1. Exclusividade de pacientes: Nenhum paciente aparece em múltiplos clientes, eliminando vazamento de dados entre conjuntos de treino e validação;

2. Diversidade de classes: Cada cliente recebe amostras de todas as três classes (AD, CN, MCI) sempre que possível, evitando especialização excessiva em uma única classe;

3. Ausência de clientes vazios: Caso o particionamento inicial resulte em clientes sem dados, realiza-se redistribuição automática para garantir que todos os clientes participantes tenham amostras suficientes para treinamento. Essa abordagem produz uma distribuição non-IID, a qual será mais detalhada na Seção 5.2.

4.2.2 Arquitetura do Cliente

A sub-rede do cliente (f_c), parametrizada por w_c, atua como extrator de características. Neste trabalho, dois pontos de corte foram explorados em diferentes arquiteturas de *backbone*:

[3]Disponível em: <https://github.com/Luanmantegazine/FedAlzheimer/blob/main/SlitFedAlzheimer.ipynb>

4.2.2.1 ResNet-50

O ponto de corte da rede foi definido após o segundo grande agrupamento de camadas da arquitetura padrão, estruturalmente denominada `layer2`, conforme detalhado na Tabela 4.1. A sub-rede do cliente inclui:

- Camadas iniciais congeladas: A primeira camada de convolução(`conv1`) e sua respectiva camada de normalização em lote (`bn1` - *Batch Normalization*, técnica utilizada para estabilizar e acelerar o treinamento padronizando as ativações) permanecem com pesos fixos durante todo o treinamento. Essa abordagem reduz o custo computacional e estabiliza a extração de características de baixo nível;
- Blocos residuais treináveis: Os agrupamento subsequentes, `layer1` (composto por 3 blocos residuais) e `layer2` (composto por 3 blocos residuais), continuam com seus pesos durante o treinamento local;
- Saída intermediária: O processamento no cliente resulta em um tensor(estrutura de dados matricial multidimensional) de dimensão $28 \times 28 \times 512$. Esses valores representam, respectivamente, as dimensões espaciais (largura e altura) e o número de canais de características extraídos, compondo os dados que serão transmitidos ao servidor.

Tabela 4.1 – Arquitetura em camadas da ResNet-50.

Tipo de Camada	**Formato de Saída**	**Kernel**	**Stride**	**Filtros/Canais**
Camada de entrada	(224, 224, 3)	–	–	3 (RGB/cinza expandido)
Conv1	(112, 112, 64)	7×7	2	64
BatchNorm + ReLU	(112, 112, 64)	–	–	64
Max pooling	(56, 56, 64)	3×3	2	64
Bloco residual 1 (layer1, 3)	(56, 56, 256)	3×3	1	$64 \rightarrow 256$
Bloco residual 2 (layer2, 4)	(28, 28, 512)	3×3	2	$128 \rightarrow 512$
Divisão cliente-servidor (mapa de características enviado ao servidor)				
Bloco residual 3 (layer3, 6)	(14, 14, 1024)	3×3	2	$256 \rightarrow 1024$
Bloco residual 4 (layer4, 3)	(7, 7, 2048)	3×3	2	$512 \rightarrow 2048$
Average pooling	(1, 1, 2048)	7×7	1	2048
Camada totalmente conectada (FC)	(1, 3)	–	–	3 (CN, MCI, AD)
Ativação Softmax	(1, 3)	–	–	Distribuição de probabilidade

4.2.2.2 DenseNet-169

Nesta arquitetura, o corte para a divisão do modelo é estabelecido após o segundo bloco denso, referenciado como `denseblock2`, conforme a Tabela 4.2. A configuração inclui:

- Camadas iniciais congeladas: De forma análoga à ResNet, a camada de convolução inicial (`conv0`) e sua respectiva camada de normalização (`norm0`) ppermanecem fixas para conferir consistência, com o intuito de garantir a estabilização inicial da rede;

- *Progressive unfreezing*: O `denseblock2` (12 camadas densas interconectadas) permanece congelado nas etapas iniciais do treinamento, sendo liberado para atualização de pesos apenas a partir da época 10. Essa estratégia permite que o modelo primeiro ajuste as camadas mais profundas antes de refinar a extração de características intermediárias;

- Blocos treináveis: O primeiro bloco denso (`denseblock1`, 6 camadas), a cada camada de transição (`transition1`, responsável por reduzir a dimensionalidade dos dados entre os blocos) e, posteriormente, o `denseblock2` contribuem ativamente para a representação das características aprendidas.

Esses pontos de corte, de ambas as redes, foram escolhidos considerando três critérios: (i) equilibrar a carga computacional entre cliente e servidor; (ii) preservar a capacidade de extração de características relevantes no cliente; e (iii) manter fluxo estável de gradientes durante a retropropagação distribuída.

Tabela 4.2 – Arquitetura em camadas da DenseNet-169.

Tipo de Camada	**Formato de Saída**	**Kernel**	**Stride**	**Canais**
Camada de entrada	(224, 224, 3)	–	–	3 (RGB/cinza expandido)
Conv0 (congelada)	(112, 112, 64)	7×7	2	64
Norm0 + ReLU0 (congeladas)	(112, 112, 64)	–	–	64
Pool0 (max pooling)	(56, 56, 64)	3×3	2	64
DenseBlock1 (6 camadas)	(56, 56, 256)	3×3	1	$64 \rightarrow 256$
Transition1	(28, 28, 128)	1×1	2	128
DenseBlock2 (12 camadas)	(28, 28, 512)	3×3	1	$128 \rightarrow 512$
Divisão cliente-servidor (mapa de características enviado ao servidor)				
Transition2	(14, 14, 256)	1×1	2	256
DenseBlock3 (32 camadas)	(14, 14, 1024)	3×3	1	$256 \rightarrow 1024$
Transition3	(7, 7, 512)	1×1	2	512
DenseBlock4 (32 camadas)	(7, 7, 1664)	3×3	1	$512 \rightarrow 1664$
Norm5 + ReLU	(7, 7, 1664)	–	–	1664
Average pooling (adaptativo)	(1, 1, 1664)	–	–	1664
Camada totalmente conectada (FC)	(1, 3)	–	–	3 (CN, MCI, AD)
Ativação Softmax	(1, 3)	–	–	Distribuição de probabilidade

4.2.3 Treinamento Local

Cada cliente executa épocas locais antes de comunicar os pesos atualizados ao servidor. O procedimento de treinamento local segue o fluxo:

1. Forward Pass Local e Backward Pass: Para cada *batch* de imagens $\{x_i, y_i\}_{i=1}^{B}$, o cliente computa:

$$h_k = f_c(x; w_c^k) \tag{4.1}$$

onde h_k representa o tensor intermediário (*smashed data*). Este tensor é então transmitido ao servidor para conclusão do *forward pass* e cálculo da perda. Após receber o gradiente $\nabla_{h_k}\mathcal{L}$ do servidor, o cliente completa a retropropagação local.

2. Otimização e Atualização de Pesos: A atualização dos parâmetros utiliza o otimizador Adam (algoritmo popular que ajusta dinamicamente a taxa de aprendizado para cada peso individualmente) com taxas de aprendizado diferenciadas. Essa diferenciação permite que ajuste conservador nas camadas intermediárias e uma adaptação mais rápida nas camadas mais profundas.

$$w_c^{k,\text{new}} = \text{Adam}(w_c^k, \nabla_{w_c^k}\mathcal{L}_k, \eta) \tag{4.2}$$

onde η é definido de acordo com a profundidade das camadas: (i) $\eta_{\text{baixa}} = 10^{-5}$ para camadas que passaram por *progressive unfreezing* (e.g., `denseblock2` na DenseNet-169 ou `layer2` na ResNet-50) e (ii) $\eta_{\text{alta}} = 10^{-4}$ para demais camadas treináveis.

4.2.4 Balanceamento em Batch

Para mitigar o desbalanceamento de classes durante o treinamento, implementa-se o `BalancedBatchSampler`. Este componente garante que todo lote processado possua a mesma quantidade de imagens de cada diagnóstico. Para isso, ele identifica a classe majoritária (MCI) e aplica *oversampling* com reposição nas classes minoritárias (AD e CN).

4.2.5 Comunicação com o Servidor

Ao final das épocas locais, o cliente envia ao servidor:

- Pesos atualizados da sub-rede do cliente: $w_c^{t+1,k}$;
- Pesos atualizados da sub-rede do servidor mantida localmente: $w_s^{t+1,k}$ (cada cliente mantém uma cópia local da sub-rede do servidor para completar o *forward pass* durante o treinamento);
- Número de amostras de treinamento: n_k (utilizado para ponderação na agregação).

4.3 Servidor

O servidor atua como orquestrador e hospeda a sub-rede, que ocorre em três etapas principais: (i) conclusão do *forward pass* e cálculo da perda; (ii) retropropagação e envio de gradientes; e (iii) agregação dos modelos locais.

4.3.1 Forward Pass e Cálculo da Perda

Ao receber o tensor intermediário h_k (*smashed data*) do cliente k, o servidor completa o *forward pass* através de sua sub-rede $f_s(h_k; w_s^k)$, produzindo as predições finais $\hat{y}_k$. A divergência entre a previsão da rede e o diagnóstico real (função de perda) é calculada utilizando *Cross-Entropy* (Entropia Cruzada, função padrão para problemas de classificação multiclasses). Adicionalmente, aplica-se o *label smoothing*(suavização de rótulos) no valor de 0.05. Essa técnica distribui uma pequena margem de incerteza na rede, que melhora a capacidade de generalização do modelo para dados novos:

$$\mathcal{L}(\hat{y}_k, y_k) = -\sum_{c=1}^{C} \alpha_c \cdot \tilde{y}_{k,c} \log(\hat{y}_{k,c}) \tag{4.3}$$

onde α_c são os pesos de classe calculados inversamente proporcionais à frequência, $\tilde{y}_{k,c}$ representa os rótulos suavizados, e C correspondem às classes AD, CN e MCI. Essa estratégia mitiga o desbalanceamento entre classes, Seção (5.4).

4.3.2 Retropropagação e Gradientes

Após calcular o erro, o servidor inicia a retropropagação, aplicando o *gradient clipping* (norma máxima de 1.0) para estabilidade. O gradiente $\nabla_{h_k}\mathcal{L}$ é então transmitido de volta ao cliente k para completar a retropropagação em sua sub-rede local. O servidor atualiza seus próprios parâmetros utilizando o otimizador Adam com taxa de aprendizado $\eta = 10^{-4}$, *weight decay* de 10^{-4} e agendamento *Cosine Annealing* com $\eta_{\min} = 10^{-6}$.

4.3.3 Agregação Federada

Ao final de cada rodada de comunicação t, o servidor coleta os pesos atualizados do cliente ($w_c^{t+1,k}$) e do servidor ($w_s^{t+1,k}$) de cada cliente participante $k \in \mathcal{S}_t$ e realiza as agregações separadamente:

- FedAvg (média ponderada): Equação 2.4;
- FedAvgM (momento no servidor): Equação 2.5, com $\beta \in (0, 1)$;
- FedProx (regularização proximal no cliente): Equação 2.7, com $\mu = 1$.

4.3.4 Avaliação Global

O servidor também é responsável pela avaliação periódica do modelo global. A cada rodada, métricas de validação (acurácia, precisão, *recall*, F1-score e AUC-ROC) são calculadas nos conjuntos de teste locais de cada cliente participante e agregadas para formar estatísticas globais. Implementa-se *Early Stopping*, monitorando a perda de validação global e salvando o melhor modelo durante o treinamento.

A divisão resulta em aproximadamente 40% dos parâmetros treináveis alocados no lado do cliente e 60% no servidor, balanceando carga computacional e privacidade dos dados. Na Figura 4.2 está representada de forma simplificada a orquestração do modelo.

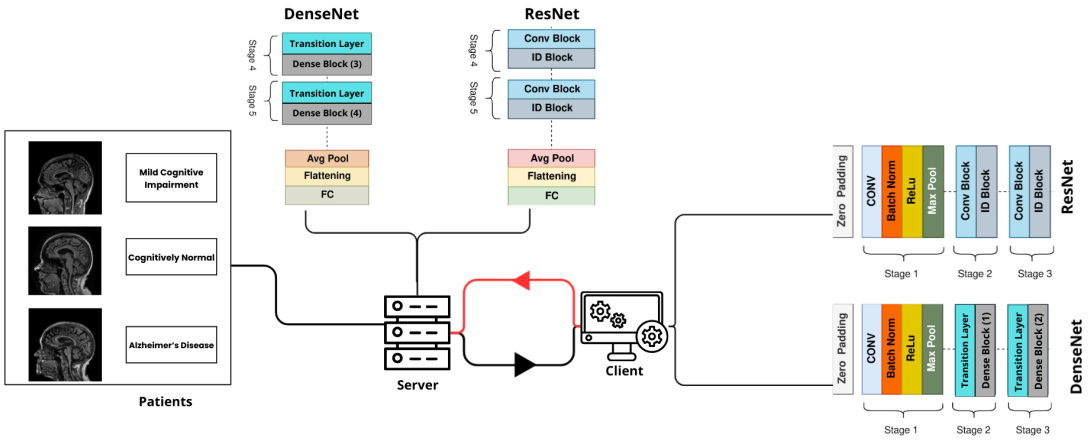


Figura 4.2 – Visão geral do modelo *Split Federated Learning*

4.4 Considerações finais

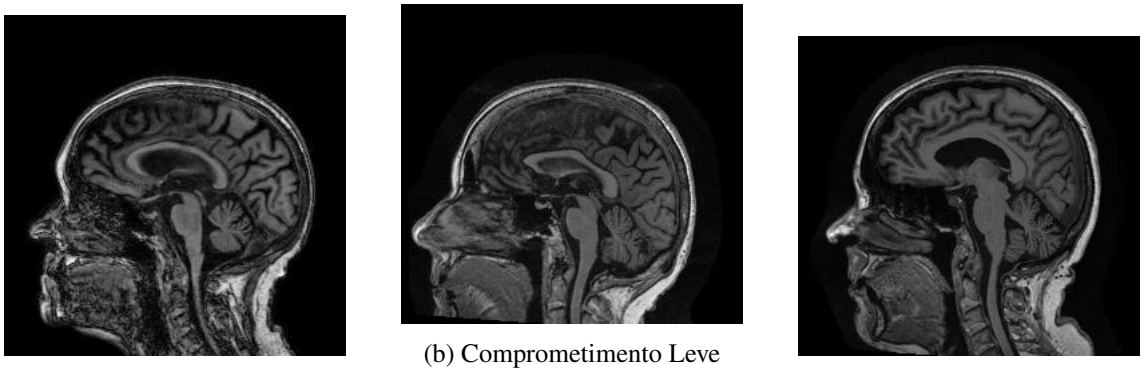
Este capítulo apresentou as características do modelo de *Federated Learning*, que foi utilizado para a realização dos testes, e introduziu etapas de funcionamento do mesmo. O modelo foi adaptado para avaliar os cenários estipulados e investigar o impacto das escolhas de diferentes arquiteturas neurais em uma arquitetura distribuída e federada. No capítulo seguinte será apresentada a metodologia utilizada para desenvolver o modelo adaptado para produzir os experimentos.

5 METODOLOGIA

Este capítulo descreve o delineamento experimental adotado nesta pesquisa, cujo objetivo é validar protocolos de configuração para o modelo adaptado. A metodologia foi concebida para permitir uma avaliação sistemática da performance de distintos algoritmos de agregação e arquiteturas neurais. Para isso, os cenários experimentais foram estruturados para investigar o impacto de um conjunto de variáveis, a saber: combinações de hiperparâmetros; conjuntos de dados de imagem; técnicas de pré-processamento e balanceamento; e distintas configurações de arquiteturas convolucionais. Cada um desses elementos é pormenorizado nas seções subsequentes deste capítulo.

5.1 Dados utilizados

Para o treinamento e avaliação do modelo, foram utilizadas imagens de MRI-T1. Os dados foram adquiridos do banco de dados ADNI (ALZHEIMER'S DISEASE NEUROIMAGING INITIATIVE, 2024). A iniciativa ADNI, lançada em 2003, tem como objetivo principal disponibilizar o uso de MRI, PET, biomarcadores e avaliações neuropsicológicas que podem auxiliar na mensuração da progressão do MCI e da AD em estágio inicial. O treinamento supervisionado foi dividido em duas fases. Na fase I, contempla um conjunto de 548 pacientes, sendo a amostra dividida em três classes: 183 CN, 267 com MCI e 99 com AD. Já na fase II, a amostra foi expandida para um grupo de 1.287 pacientes, dividiu-se em: 448 CN, 587 com MCI e 252 com AD. Esse tipo de estratégia, visou estudar o comportamento do algoritmo perante a escalabilidade de dados.



(a) Doença de Alzheimer (AD) (b) Comprometimento Leve (MCI) (c) Estágio Normal (CN)

Figura 5.1 – Imagens em MRI representativas dos estágios da AD.

As Figuras 5.1 mostram as imagens usadas nos treinamentos. Na Figura 5.1a, observa-se atrofia pronunciada do hipocampo e do córtex temporal medial, sulcos corti-

cais acentuadamente alargados e aumento ventricular posterior, achados que se alinham com a doença de Alzheimer; a Figura 5.1b demonstra atrofia moderada do hipocampo e do giro parahipocampal, sulcos levemente acentuados e dilatação ventricular sutil, características sugestivas de comprometimento cognitivo leve; enquanto a Figura 5.1c exibe anatomia girocortical e sulcal preservada, contornos hipocampais bem demarcados e dimensões ventriculares dentro dos limites esperados para a idade, consistente com um perfil cognitivamente normal.

5.2 Pré-processamento de dados

Inicialmente, as imagens foram extraídas no formato NIfTI (*Neuroimaging Informatics Technology Initiative*), comumente utilizado em estudos de neuroimagem, por reunir em um único contêiner, o volume 3-D/4-D de voxels e um cabeçalho de 352 bytes que descreve dimensões, espaçamento físico e orientação anatômica (ELHADAD; JAMJOOM; ABULKASIM, 2024). Tratando-se de um treinamento supervisionado, onde as imagens já estão categorizadas e visando a diminuição do custo computacional, foi aplicada uma transformação utilizando o *framework* `med2image`, que é uma utilidade de linha de comando, distribuída sob licença MIT, criada pelo *Fetal-Neonatal Neuroimaging & Developmental Science Center* (FNNDSC) para converter arquivos de imagem médica, atualmente DICOM (*Digital Imaging and Communications in Medicine*) e NIfTI em formatos 2-D usualmente PNG e JPEG. A ferramenta encontra-se na versão 2.6.6 e é implementada inteiramente em Python (FETAL-NEONATAL NEUROIMAGING & DEVELOPMENTAL SCIENCE CENTER, 2022).

Antes da execução do *framework*, os arquivos brutos necessitavam de organização em pastas com suas classificações. Para automatizar essa etapa inicial, desenvolveu-se um *script* em Python. Diante disso, as imagens NIfTI foram renomeadas com o ID do paciente, facilitando a identificação para futura conversão para a transformação. Como última parte de pré-processamento, foi instalado o pacote `med2image`.

Por fim, as imagens foram redimensionadas para a entrada de 224×224, convertidas para tons de cinza com 3 canais, e normalizadas por canal com média 0,5 e desvio padrão 0,5. Foram aplicadas as seguintes transformações:

- `RandomHorizontalFlip (p=0.5)`;
- `RandomRotation` (até 15°);
- `RandomAffine` (com *translate* e *scale*);
- `RandomNoise`;
- `Normalize([0.5]*3, [0.5]*3)`.

5.3 Particionamento dos dados

Visando evitar o vazamento de dados entre os clientes, implementou-se uma divisão estratificada, com base nos IDs dos pacientes. Inicialmente, o conjunto completo de imagens foi particionado em subconjuntos de treino e teste utilizando o método `StratifiedGroupKFold` ($n = 5$ *folds*). As classes diagnósticas foram utilizadas como variável de estratificação, assegurando que as proporções de cada classe fossem aproximadamente preservadas entre treino e teste. Simultaneamente, foi imposto um agrupamento por paciente (groups = `patient_ds`), de modo que todas as imagens pertencentes a um dado paciente permanecessem no mesmo subconjunto (treino ou teste), prevenindo vazamento de informação entre os conjuntos. Na prática, foi utilizado apenas o primeiro split gerado pelo `StratifiedGroupKFold`, resultando em uma divisão próxima de 80% das imagens para treino e 20% para teste.

Foi realizada uma verificação de sanidade para garantir a ausência de vazamentos de pacientes entre treino e teste, assegurando que o conjunto de IDs de pacientes no treino e no teste fosse estritamente disjunto.

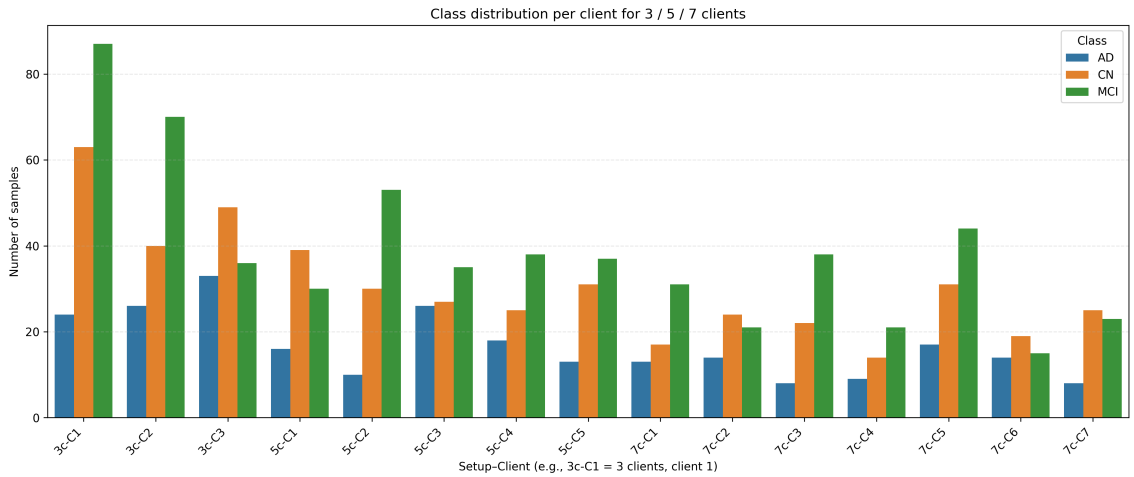


Figura 5.2 – Distribuição de dados na Fase I

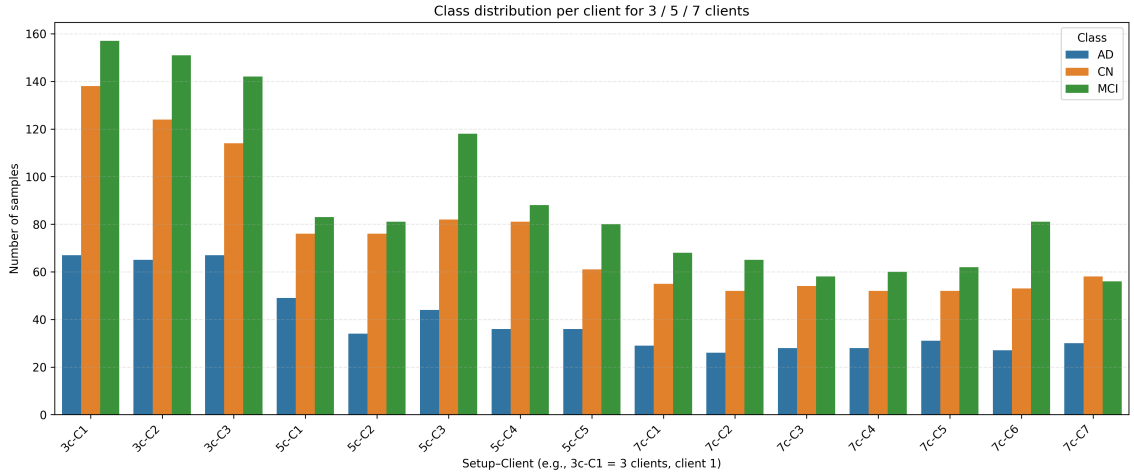


Figura 5.3 – Distribuição de dados na Fase II

As Figuras 5.3 e 5.2 ilustram como terminou esse processo de distribuição de dados entre clientes.

5.4 Balanceamento dos dados

Para mitigar o viés nos resultados, três técnicas complementares foram aplicadas, visando garantir representatividade adequada de todas as classes durante o treinamento.

Primeiramente, calculou-se a taxa de desbalanceamento para cada classe. Segundo He e Garcia (2009), deve-se tirar a razão entre o número de amostras da classe majoritária e o número de amostras da classe minoritária. Tomando a classe MCI como majoritária, a taxa de balanceamento resultou em $IR_1 \cong 2,70$ para a fase I e $IR_2 \cong 2,32$ para a fase II.

Resultados indicaram um desequilíbrio considerável; a classe AD apresenta uma quantidade muito inferior de elementos, o que pode resultar em valores não representativos para métricas como *recall* e *precision*. Para solucionar esse possível problema, utilizou-se uma abordagem tripla de balanceamento:

(1) Pesos de Classe: Calculou-se pesos balanceados inversamente proporcionais à frequência de cada classe utilizando `compute_class_weight` do scikit-learn. Esses pesos foram aplicados na função de perda `CrossEntropyLoss`, atribuindo maior penalização a erros nas classes minoritárias (AD e CN). Adicionalmente, aplicou-se *label smoothing* de 0.05 para regularização.

(2) `BalancedBatchSampler`: Implementou-se um amostrador personalizado que garante representação equilibrada de todas as classes em cada *batch* durante o treinamento. O *sampler* replica amostras de classes minoritárias (*oversampling*) até igualar o tamanho

da classe majoritária, distribuindo uniformemente n_{per_class} amostras de cada classe por textitbatch, onde $n_{per_class} = \lfloor batch_size/num_classes \rfloor$.

(3) Particionamento Balanceado por Paciente: Na distribuição dos dados entre clientes federados, utilizou-se estratégia *round-robin* que aloca pacientes de forma equilibrada por classe, garantindo diversidade em cada cliente e evitando clientes vazios. Essa abordagem preserva a exclusividade de pacientes (sem *data leakage* entre treino e validação) enquanto mantém distribuição balanceada.

Ao final desse processo triplo, o modelo é treinado com *batches* equilibrados, função de perda ponderada e distribuição balanceada entre clientes, mitigando efetivamente o desbalanceamento original dos dados.

5.5 Protocolos experimentais

Para este trabalho, testes foram conduzidos em diferentes cenários simulados. Com o intuito de alcançar os objetivos, foi definida uma base de hiperparâmetros invariáveis, detalhada na Tabela 5.1. O ambiente distribuído federado simulou abordagens com 3, 5 e

Tabela 5.1 – Configurações dos experimentos

Rede Neural	**Modelo Agregador**	**Clientes**	**Épocas**	**Batch**
DenseNet-169 / ResNet-50	FedAvg / FedAvgM / FedProx	3, 5, 7	80	32 / 64

Fonte: Autor, 2025

7 clientes e um servidor centralizado, os dados foram divididos de maneira non-IID entre os clientes, garantindo que cada um recebesse uma porção não balanceada, dividiu-se esses dados entre 80% para treino e 20% para validação. O treinamento foi realizado em *batches* de tamanho 32 e 64, utilizando o otimizador Adam, enquanto o servidor agrega os pesos locais usando FedAvgM (*momentum* $\beta = 0.9$), FedProx (regularização $\mu = 0.1$) e FedAvg. A Tabela 5.2 mostra todos os cenários simulados, estes foram reproduzidos por três variando o valor do SEED em 12, 123 e 1234, isso com a finalidade de se obter uma amostragem relevante para extração da validação estatística.

Tabela 5.2 – Cenários simulados para os testes.

Cenários	Rede Neural	Modelo Agregador	Clientes	Tamanho da batch
1	ResNet-50	FedAvg	3	32
2	ResNet-50	FedAvg	5	32
3	ResNet-50	FedAvg	7	32
4	ResNet-50	FedAvg	3	64
5	ResNet-50	FedAvg	5	64
6	ResNet-50	FedAvg	7	64
7	ResNet-50	FedAvgM	3	32
8	ResNet-50	FedAvgM	5	32
9	ResNet-50	FedAvgM	7	32
10	ResNet-50	FedAvgM	3	64
11	ResNet-50	FedAvgM	5	64
12	ResNet-50	FedAvgM	7	64
13	ResNet-50	FedProx	3	32
14	ResNet-50	FedProx	5	32
15	ResNet-50	FedProx	7	32
16	ResNet-50	FedProx	3	64
17	ResNet-50	FedProx	5	64
18	ResNet-50	FedProx	7	64
19	DenseNet-169	FedAvg	3	32
20	DenseNet-169	FedAvg	5	32
21	DenseNet-169	FedAvg	7	32
22	DenseNet-169	FedAvg	3	64
23	DenseNet-169	FedAvg	5	64
24	DenseNet-169	FedAvg	7	64
25	DenseNet-169	FedAvgM	3	32
26	DenseNet-169	FedAvgM	5	32
27	DenseNet-169	FedAvgM	7	32
28	DenseNet-169	FedAvgM	3	64
29	DenseNet-169	FedAvgM	5	64

(*continua na próxima página*)

Tabela 5.2 – – (continuação)

Cenários	Rede Neural	Modelo Agregador	Clientes	Tamanho da batch
30	DenseNet-169	FedAvgM	7	64
31	DenseNet-169	FedProx	3	32
32	DenseNet-169	FedProx	5	32
33	DenseNet-169	FedProx	7	32
34	DenseNet-169	FedProx	3	64
35	DenseNet-169	FedProx	5	64
36	DenseNet-169	FedProx	7	64

Fonte: Autor, 2025

5.6 Limitações

A principal limitação deste estudo é o tamanho relativamente pequeno do conjunto amostral geral. Numa configuração distribuída, esta restrição pode levar à inflação de métricas que complicam a interpretação das curvas de treinamento, revelando possíveis padrões de *overfitting* e flutuações no desempenho de validação.

Outra limitação significativa é que todos os experimentos foram executados em um único nó usando uma GPU T4 no Google Colab, simulando um ambiente federado sem restrições reais de rede. Como resultado, métricas de comunicação entre clientes e servidor - incluindo latência, uso de largura de banda e número de rodadas de comunicação necessárias para convergência - não foram medidas.

Uma limitação metodológica reside no conjunto restrito de algoritmos de otimização federada considerados. Embora este estudo tenha focado em FedAvg, FedAvgM e FedProx, outras abordagens promissoras como SCAFFOLD, FedNova e FedOpt não foram implementadas devido à restrições computacionais. Estes algoritmos abordam mais efetivamente o *client drift* e o viés do lado do servidor, e deveriam ser incluídos em comparações futuras para fornecer uma análise mais abrangente do comportamento de otimização sob dados médicos non-IID.

Adicionalmente, embora o FL ofereça uma camada inerente de privacidade ao manter os dados brutos de ressonância magnética restritos aos dispositivos locais, este trabalho não incorporou técnicas avançadas de defesa contra ataques de inferência (como

a tentativa de reconstrução de imagens a partir da interceptação de gradientes). Especificamente, dois mecanismos robustos foram deliberadamente desativados:

- Privacidade Diferencial via DP-SGD (*Differentially Private Stochastic Gradient Descent*): Técnica que injeta ruído matemático intencional nos gradientes antes de enviá-los ao servidor, garantindo que a contribuição de um paciente individual não possa ser memorizada pela rede;
- Agregação Segura (*Secure Aggregation*): Protocolo criptográfico que garante que o servidor central tenha acesso apenas à soma (ou média) das atualizações dos clientes, impedindo a visualização dos pesos enviados por um hospital específico.

5.7 Considerações finais

Este capítulo detalhou a metodologia e os parâmetros de configuração necessários para a reprodução dos experimentos aplicados à classificação da AD. Descreveu-se desde o pré-processamento das imagens de MRI-T1 e o balanceamento das classes (AD, MCI e CN), até a implementação do particionamento de dados non-IID entre os nós da rede. Foram estruturados protocolos experimentais focados em avaliar o treinamento das redes ResNet-50 e DenseNet-169 sob a influência de diferentes algoritmos de agregação federada (FedAvg, FedAvgM e FedProx) em federações de 3, 5 e 7 clientes. A análise estatística detalhada desses cenários e a avaliação de seus respectivos resultados de classificação serão apresentadas no próximo capítulo.

6 RESULTADOS E AVALIAÇÕES

Este capítulo apresenta e discute os resultados obtidos nos 216 experimentos conduzidos para avaliar o desempenho de diferentes combinações de arquiteturas de redes neurais e algoritmos de agregação federada na tarefa de classificação dos estágios da AD.

6.1 Visão geral

A investigação foi estruturada em duas fases experimentais complementares. A Fase I utilizou um conjunto reduzido de 548 pacientes, distribuídos entre as classes Cognitivamente Normal (CN), Comprometimento Cognitivo Leve (MCI) e Doença de Alzheimer (AD), com o objetivo de caracterizar o comportamento dos modelos sob condições de escassez de dados. A Fase II ampliou o conjunto amostral para 1.287 pacientes, permitindo avaliar a escalabilidade dos modelos e verificar se as tendências identificadas na fase anterior se mantinham ou se invertiam com maior volume de informação. Os cenários experimentais combinaram duas arquiteturas (ResNet-50 e DenseNet-169), três algoritmos de agregação federada (FedAvg, FedAvgM e FedProx), três configurações de número de clientes (3, 5 e 7) e dois tamanhos de *batch* (32 e 64), totalizando 36 cenários por fase. A análise dos resultados é organizada em torno de quatro eixos interpretativos:

(i) escalabilidade em função do número de clientes federados;

(ii) comparação entre os algoritmos de agregação;

(iii) impacto do tamanho do *batch* na estabilidade do treinamento;

(iv) equilíbrio entre precisão e sensibilidade, com ênfase nas implicações clínicas para o diagnóstico da DA.

Cada fase foi executada três vezes com valores de *seed* distintos ($\{12,\ 123,\ 1234\}$), de modo que os resultados apresentados nas Tabelas 6.1 e 6.4 correspondem a médias e respectivos desvios-padrão. As curvas de validação, estão exibidas nas Figuras 6.2 e 6.4 para as Fases I e II, respectivamente. Para uma análise qualitativa dos erros, as Figuras 6.1 e 6.5 apresentam as matrizes de confusão para combinações representativas de cenários, algoritmos e arquiteturas, detalhando a distribuição das predições entre as classes após a distribuição *non-IID* dos dados entre os clientes.

A avaliação de desempenho dos modelos foi conduzida por meio de cinco métricas complementares. Dado o caráter triclasse do problema e o desbalanceamento inerente ao conjunto de dados (razão de desbalanceamento IR $\approx 2{,}70$ na Fase I e IR $\approx 2{,}32$ na Fase II), todas as métricas são agregadas pela média macro, que atribui peso igual a cada classe independentemente de sua frequência, evitando que a classe majoritária (MCI) domine artificialmente os resultados.

As definições são formuladas em termos da matriz de confusão para a classe c, onde TP_c, FP_c e FN_c denotam, respectivamente, verdadeiros positivos, falsos positivos e falsos negativos associados à classe c, e N corresponde ao número total de amostras no conjunto de teste, com $C = \{\mathrm{AD}, \mathrm{MCI}, \mathrm{CN}\}$.

- Acurácia (*Accuracy* — ACC): Mede a proporção global de predições corretas em relação ao total de amostras avaliadas, independentemente da classe:

$$\mathrm{ACC} = \frac{\sum_{c \in C} \mathrm{TP}_c}{N} \tag{6.1}$$

- Precisão (*Precision*): Quantifica a fração de predições positivas para uma dada classe que são de fato corretas. Sua forma por classe é:

$$\mathrm{Precision}_c = \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FP}_c} \tag{6.2}$$

  e o valor reportado nas tabelas corresponde à média macro sobre as três classes:

$$\mathrm{Precision}_{\mathrm{macro}} = \frac{1}{|C|} \sum_{c \in C} \mathrm{Precision}_c \tag{6.3}$$

- Sensibilidade (*Recall*): Mede a capacidade do modelo de identificar corretamente todas as amostras pertencentes a uma classe:

$$\mathrm{Recall}_c = \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FN}_c} \tag{6.4}$$

  com agregação:

$$\mathrm{Recall}_{\mathrm{macro}} = \frac{1}{|C|} \sum_{c \in C} \mathrm{Recall}_c \tag{6.5}$$

- *F1-score*: É a média harmônica entre precisão e sensibilidade, oferecendo síntese

equilibrada das duas métricas. Sua forma por classe é:

$$F_1^{(c)} = 2 \cdot \frac{\text{Preciso}_c \times \text{Recall}_c}{\text{Preciso}_c + \text{Recall}_c} \tag{6.6}$$

e o valor reportado é:

$$F_{1,\,\text{macro}} = \frac{1}{|C|} \sum_{c \in C} F_1^{(c)} \tag{6.7}$$

- Área Sob a Curva ROC (*AUC-ROC*): Quantifica a capacidade discriminativa geral do classificador ao longo de todos os limiares de decisão possíveis. Para o caso triclasse, adota-se a formulação *one-vs-rest* com agregação macro:

$$\text{AUC}_{\text{macro}} = \frac{1}{|C|} \sum_{c \in C} \int_0^1 \text{TPR}_c\left(\text{FPR}_c^{-1}(t)\right) dt \tag{6.8}$$

  onde TPR_c e FPR_c denotam, respectivamente, a taxa de verdadeiros positivos e a taxa de falsos positivos para a classe c em função do limiar de classificação.

Todas as métricas foram calculadas com a biblioteca Scikit-learn, o código de avaliação está disponível no repositório público.[1]

6.2 Fase I

A primeira fase concentrou-se na validação preliminar com um conjunto de dados limitado. O DenseNet-169 alcançou a maior acurácia (86,97% ± 1,14%), F1-score (82,05% ± 3,30%) e AUC (96,58% ± 0,43%) nesta etapa, mantendo um desempenho eficaz em federações maiores com cinco clientes. O ResNet-50 mostrou-se competitivo, atingindo valores comparáveis de acurácia (85,75% ± 0,53%), F1-score (77,09% ± 1,42%) e AUC (94,29% ± 0,23%); No entanto, suas métricas declinaram ao escalonar o modelo. A análise nesta fase centrou-se na escalabilidade da rede, modelos de agregação, tamanho da *batch* e precisão versus sensibilidade.

[1] <https://github.com/Luanmantegazine/FedAlzheimer/blob/main/validation_stats.py>

Tabela 6.1 – Resultados médios (% ± desvio padrão) dos experimentos na Fase I

Arquitetura	FL	Batch	Clientes	ACC (%)	Precisão (%)	Recall (%)	F1 (%)	AUC(%)
DenseNet-169	FedAvg	32	3	83,44 ± 4,26	77,85 ± 4,57	79,41 ± 6,34	77,07 ± 5,11	95,33 ± 1,46
DenseNet-169	FedAvg	32	5	79,61 ± 4,63	75,87 ± 4,39	78,35 ± 4,71	74,59 ± 4,61	95,54 ± 3,87
DenseNet-169	FedAvg	32	7	80,22 ± 5,07	75,44 ± 3,63	74,66 ± 2,34	73,35 ± 4,19	94,81 ± 1,68
DenseNet-169	FedAvg	64	3	86,97 ± 1,13	82,58 ± 1,91	82,03 ± 5,31	82,05 ± 3,50	96,58 ± 0,43
DenseNet-169	FedAvg	64	5	82,98 ± 7,31	78,86 ± 6,83	77,07 ± 7,71	77,31 ± 7,66	91,55 ± 4,56
DenseNet-169	FedAvg	64	7	82,15 ± 6,58	79,96 ± 9,32	74,70 ± 3,43	76,27 ± 5,81	93,03 ± 3,67
DenseNet-169	FedProx	32	3	85,32 ± 2,34	80,12 ± 2,65	81,01 ± 4,51	79,29 ± 0,84	94,95 ± 1,44
DenseNet-169	FedProx	32	5	80,16 ± 5,09	73,62 ± 7,32	74,64 ± 5,23	73,24 ± 6,03	91,98 ± 3,83
DenseNet-169	FedProx	32	7	80,50 ± 5,41	75,26 ± 6,35	76,95 ± 4,55	74,85 ± 5,72	94,21 ± 1,63
DenseNet-169	FedProx	64	3	80,77 ± 4,49	75,01 ± 5,40	80,13 ± 2,54	76,15 ± 4,67	93,26 ± 3,63
DenseNet-169	FedProx	64	5	84,99 ± 1,62	78,52 ± 4,18	78,84 ± 3,08	77,89 ± 3,37	94,98 ± 0,28
DenseNet-169	FedProx	64	7	79,04 ± 5,61	76,01 ± 8,25	72,21 ± 3,46	71,90 ± 5,88	93,48 ± 1,58
DenseNet-169	FedAvgM	32	3	75,67 ± 3,45	71,23 ± 2,89	71,56 ± 4,12	69,89 ± 3,45	86,45 ± 1,12
DenseNet-169	FedAvgM	32	5	71,45 ± 5,12	66,34 ± 3,45	68,67 ± 4,78	64,56 ± 3,12	84,78 ± 3,12
DenseNet-169	FedAvgM	32	7	78,45 ± 4,23	74,12 ± 5,45	73,78 ± 4,89	73,67 ± 5,45	86,45 ± 2,12
DenseNet-169	FedAvgM	64	3	78,23 ± 1,45	74,89 ± 1,67	74,12 ± 4,56	74,23 ± 3,12	87,23 ± 0,56
DenseNet-169	FedAvgM	64	5	74,56 ± 5,45	71,23 ± 3,89	69,45 ± 4,78	69,67 ± 6,45	83,45 ± 3,78
DenseNet-169	FedAvgM	64	7	74,23 ± 5,89	72,45 ± 7,45	68,89 ± 3,12	68,45 ± 4,89	84,56 ± 3,89

(*continua na próxima página*)

Tabela 6.1 – – (continuação)

Arquitetura	FL	Batch	Clientes	ACC (%)	Precisão (%)	Recall (%)	F1 (%)	AUC(%)
ResNet-50	FedAvg	32	3	85,75 ± 0,53	78,29 ± 2,25	77,07 ± 0,66	77,09 ± 1,42	94,29 ± 0,23
ResNet-50	FedAvg	32	5	80,91 ± 1,58	71,41 ± 3,60	74,83 ± 0,40	72,07 ± 2,73	91,19 ± 0,36
ResNet-50	FedAvg	32	7	47,74 ± 3,38	36,59 ± 3,39	43,75 ± 2,12	36,38 ± 3,46	63,36 ± 3,09
ResNet-50	FedAvg	64	3	78,73 ± 4,92	72,10 ± 5,28	71,22 ± 2,05	71,00 ± 3,47	88,45 ± 2,43
ResNet-50	FedAvg	64	5	75,92 ± 7,32	69,29 ± 5,52	70,22 ± 3,83	69,10 ± 5,40	89,10 ± 2,17
ResNet-50	FedAvg	64	7	68,71 ± 14,75	54,97 ± 17,41	58,09 ± 16,06	55,01 ± 15,51	77,49 ± 17,32
ResNet-50	FedProx	32	3	78,48 ± 4,49	72,81 ± 1,12	75,24 ± 0,49	72,73 ± 0,92	90,46 ± 2,31
ResNet-50	FedProx	32	5	73,94 ± 10,46	69,67 ± 9,00	70,42 ± 7,17	68,63 ± 7,99	88,16 ± 9,24
ResNet-50	FedProx	32	7	41,21 ± 31,82	35,37 ± 29,38	45,96 ± 20,61	34,95 ± 30,56	62,83 ± 19,63
ResNet-50	FedProx	64	3	80,17 ± 8,39	76,30 ± 8,48	77,45 ± 8,18	76,42 ± 8,28	89,40 ± 7,16
ResNet-50	FedProx	64	5	77,63 ± 8,91	70,90 ± 12,31	69,55 ± 17,21	69,38 ± 15,13	90,40 ± 7,15
ResNet-50	FedProx	64	7	66,64 ± 5,34	46,15 ± 21,86	48,93 ± 15,07	47,07 ± 18,91	66,91 ± 23,43
ResNet-50	FedAvgM	32	3	77,25 ± 1,15	70,14 ± 1,89	69,08 ± 1,12	69,32 ± 1,24	85,41 ± 0,45
ResNet-50	FedAvgM	32	5	72,63 ± 1,42	70,28 ± 2,85	69,54 ± 1,32	74,12 ± 2,18	82,67 ± 0,58
ResNet-50	FedAvgM	32	7	72,45 ± 0,85	65,67 ± 1,45	73,89 ± 1,23	89,76 ± 1,05	82,34 ± 0,67
ResNet-50	FedAvgM	64	3	80,92 ± 3,67	74,89 ± 4,12	73,45 ± 2,89	73,56 ± 3,12	90,23 ± 1,87
ResNet-50	FedAvgM	64	5	78,34 ± 5,89	71,84 ± 4,67	72,89 ± 3,45	71,45 ± 4,78	90,87 ± 1,92
ResNet-50	FedAvgM	64	7	71,23 ± 12,45	58,34 ± 15,23	61,45 ± 14,12	58,23 ± 13,89	80,12 ± 15,67

6.2.1 Escalabilidade em Função do Número de Clientes

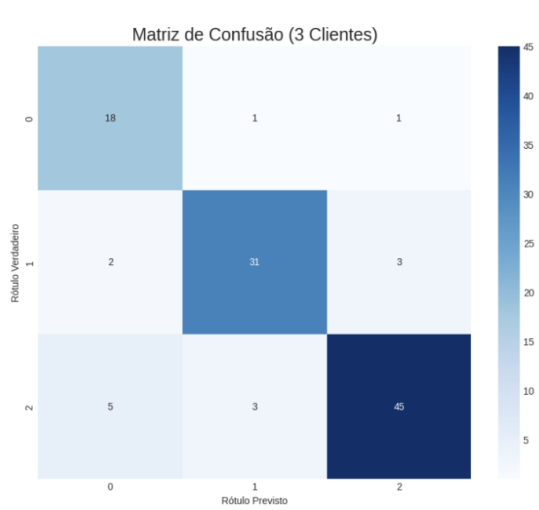
A análise de escalabilidade examinou como o desempenho das arquiteturas foi afetado pelo aumento do número de clientes na federação. Os resultados demonstraram que ambas as redes exibiram maior estabilidade ao operar com três clientes, enquanto aumentos no número de participantes foram acompanhados por um crescimento do ruído nas métricas de avaliação.

De modo geral, o DenseNet-169 manteve um desempenho superior em comparação com o ResNet-50 à medida que o número de clientes aumentava. Por exemplo, em experimentos com 7 clientes, a arquitetura residual exibiu um declínio acentuado de desempenho, culminando em um colapso. Por outro lado, o DenseNet-169, embora tenha apresentado degradação de desempenho em todas as métricas avaliativas, demonstrou maior confiabilidade em federações maiores.

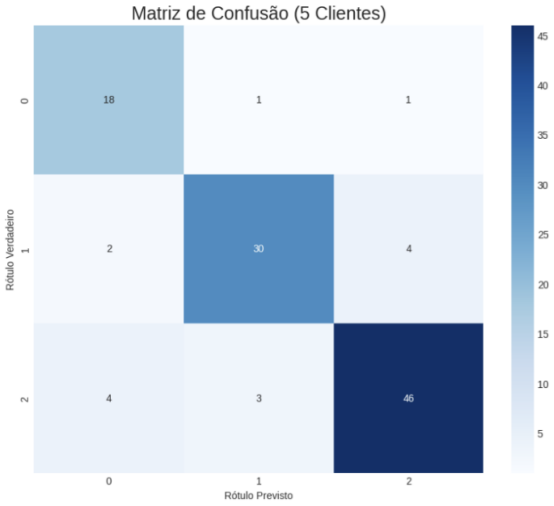
No entanto, as curvas de validação, Figura 6.2, revelaram que a instabilidade no treinamento do DenseNet-169 aumentou com o FedProx conforme o número de clientes crescia. Esta observação está alinhada com Li et al. (2019), onde um termo proximal alto mitiga o *client drift*, mas aumenta o custo computacional e a latência, amplificando assim a variabilidade da convergência.

Coletivamente, com base na Figura 6.1, a análise de ambas as matrizes de confusão e curvas de aprendizagem aponta para uma conclusão central: o impacto da configuração do modelo torna-se mais pronunciado em federações maiores. Este achado é consistente com a pesquisa de Wallach, Siler e Deng (2025), que observou uma degradação significativa da acurácia com o aumento do número de clientes. Os autores atribuem esse declínio não a algoritmos ou arquiteturas específicas, mas ao desafio mais fundamental da fragmentação dos dados. Conforme clientes adicionais ingressam na federação, cada um possui uma porção menor do total de dados, forçando os modelos locais a treinarem com informações limitadas e, consequentemente, degradando a qualidade do modelo agregado final. Isso explica por que as limitações de cada abordagem tornaram-se cada vez mais evidentes em nossos experimentos à medida que a escala da federação aumentou.

Matrizes de confusão para a ResNet (Fase I)

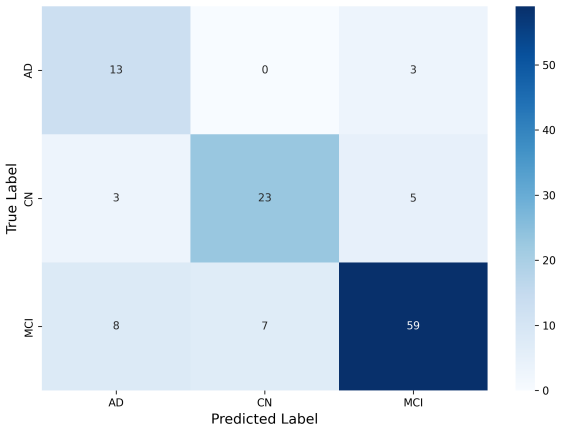


(a) 3 Clientes.

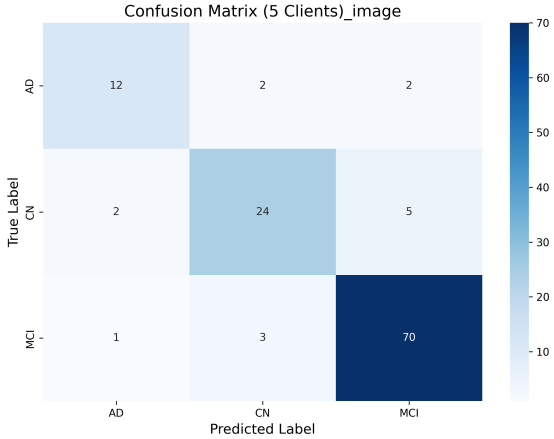


(b) 5 Clientes.

Matrizes de confusão para a DenseNet (Fase I)



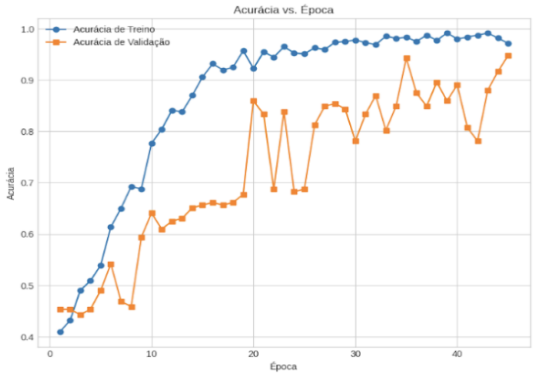
(c) 3 Clientes.



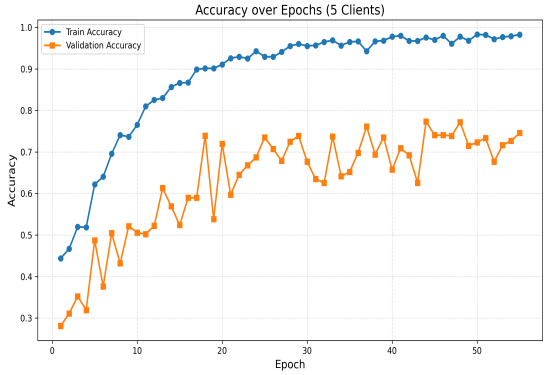
(d) 5 Clientes.

Figura 6.1 – Compilação de matrizes de confusão para diferentes combinações de algoritmos de agregação e arquiteturas de modelo, detalhando o desempenho da classificação para 3 e 5 clientes.

Cenário 21 & Cenário 14

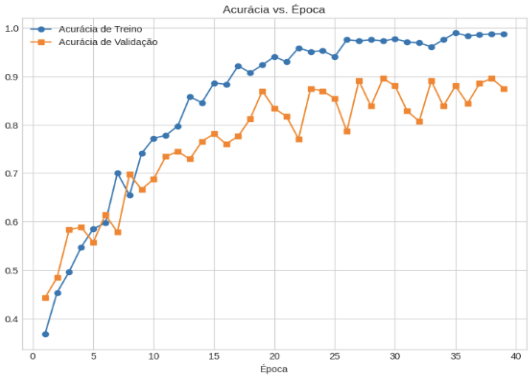


(a) Acurácia - 7 Clientes

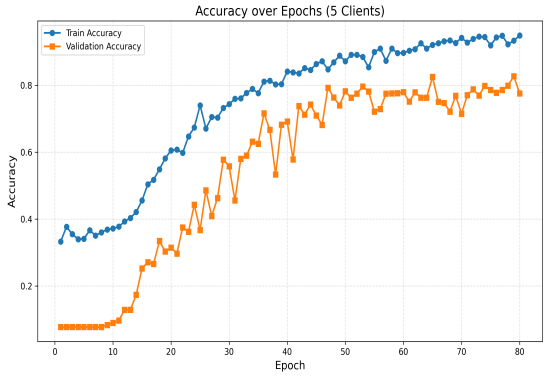


(b) Acurácia - 5 Clientes

Cenário 19 & Cenário 35

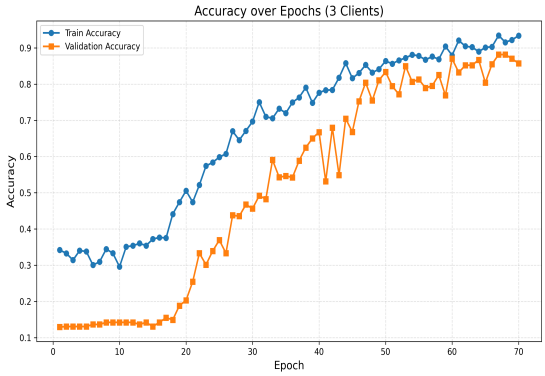


(c) Acurácia - 3 clientes

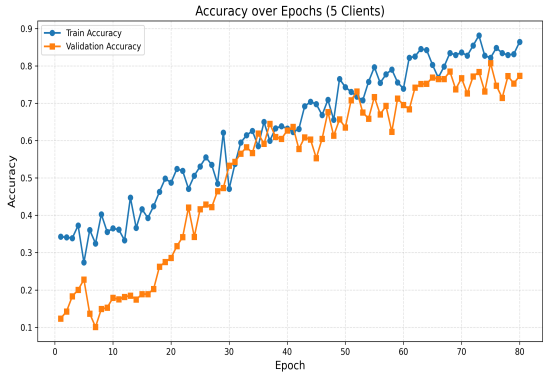


(d) Acurácia - 5 clientes

Cenário 22 & Cenário 32



(e) Acurácia - 3 clientes



(f) Acurácia - 5 clientes

Figura 6.2 – Compilação das curvas de Acurácia para cenários selecionados da Fase I.

6.2.2 Modelos de Agregação

Ao analisar os resultados para cada algoritmo de agregação, constatou-se que, nesta fase, FedProx e FedAvg se destacaram em relação ao FedAvgM, particularmente quando combinados com a DenseNet-169.

Com 3 clientes, o FedAvg alcançou a melhor acurácia geral (86,97% ± 1,14%), enquanto o FedProx atingiu 85,32% ± 2,34%, além disso, esse modelo obteve resultados mais próximos, independentemente da arquitetura conectada. Esse resultado está alinhado com Li et al. (2020), que mostram que a adição de um termo proximal pode melhorar a estabilidade do treinamento em cenários com maior heterogeneidade e mais clientes, ainda que o FedAvg permaneça competitivo.

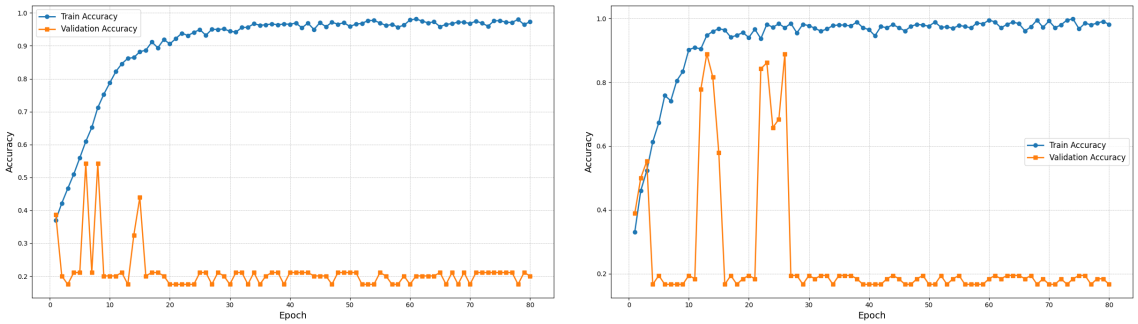
O FedAvg, algoritmo predominante nas pesquisas analisadas, obteve o melhor desempenho e se destacou por sua notável estabilidade, mostrando-se adaptável a diferentes configurações de redes e hiperparâmetros. Essa característica é amplamente reconhecida na literatura, com Kairouz et al. (2021) posicionando o FedAvg como o benchmark padrão da área, a partir do qual inúmeras variantes são propostas, justamente por sua simplicidade e comportamento previsível.

Em contraste, o FedAvgM apresentou desempenho instável em todos os cenários. A acurácia da DenseNet-169, por exemplo, caiu para 75,67% (3 clientes, *batch size*=32) com FedAvgM, em comparação a 83,44% com FedAvg na mesma configuração, evidenciando a perda de desempenho introduzida pelo momento no servidor, ilustrada na Figura 6.3

A nossa hipótese para explicar essa instabilidade é um acúmulo excessivo de momento no servidor. Essa vulnerabilidade pode ser explicada pela própria mecânica do FedAvgM, equação 2.5, em que, para $\nu_t = 1$, a direção do modelo global (w_{t+1}) é determinada exclusivamente pelo vetor *momentum* acumulado (ν_{t+1}), ignorando a contribuição da atualização global da rodada atual (Δ_t). Esse comportamento é consistente com achados recentes: Alrawahna et al. (2025) demonstram empiricamente que valores elevados de momentum (e.g., $\mu = 0{,}9$) causam instabilidade e oscilações sob forte heterogeneidade de dados; e Li et al. (2025) identificam, em cenários *non-IID* com distribuição desbalanceada, que o *momentum* induz uma *distorção direcional global*, amplificando gradientes ruidosos em vez de suavizá-los.

Consequentemente, após várias rodadas, o vetor de momento se torna excessivamente grande, criando uma forte inércia. Uma única atualização de cliente que seja ruidosa é então amplificada desproporcionalmente por esse momento acumulado, pois

não há um componente da atualização rápida para contrabalançá-la, provocando a queda catastrófica de performance observada por Cheng et al. (2024).



(a) ResNet-50. (b) DenseNet-169.

Figura 6.3 – Curvas de validação do FedAvgM - Fase I.

6.2.3 Impacto do *Batch Size*

A escolha do *batch size* pode não ser um fator decisivo no geral. Por exemplo, com FedAvg e 3 clientes, a DenseNet-169 obteve 83,44% ± 4,32% de acurácia com *batch size*=32 e 86,97% ± 1,13% com *batch size*=64. Apesar do ganho de cerca de 3.5 pontos percentuais ao aumentar o *batch size*, as duas configurações mantiveram desempenho competitivo, indicando que o tamanho da *batch* não foi o fator determinante nesta fase, conforme sumarizado na Tabela 6.2.

Tabela 6.2 – Melhores resultados da Fase I.

FL	Batch	Rede	Clientes	Acurácia	F1-score	AUC
FedAvg	64	DenseNet-169	3	86,97% ± 1,14%	82,05% ± 3,30%	96,58% ± 0,43%
FedAvg	32	ResNet-50	3	85,75% ± 0,53%	77,09% ± 1,42%	94,29% ± 0,23%
FedProx	32	DenseNet-169	3	85,32% ± 2,34%	79,29% ± 0,84%	94,95% ± 1,44%
FedProx	64	DenseNet-169	5	84,99% ± 1,62%	77,89% ± 3,37%	94,98% ± 0,28%
FedAvg	32	DenseNet-169	3	83,44% ± 4,26%	77,07% ± 5,11%	95,33% ± 1,46%

6.2.4 *Recall* vs Precisão

Nesta seção, descreve-se a qualidade dos resultados sob duas óticas: o equilíbrio entre precisão e o *recall*, e a dinâmica de convergência pelas curvas de aprendizado. Conforme ilustrado na Tabela 6.3, a DenseNet-169 equilibrou eficazmente a precisão e a sensibilidade. Por exemplo, com FedAvg e batch size = 64, obteve 82,58% ± 1,91% de precisão e 82,03% ± 5,31% de sensibilidade, tornando-a adequada para aplicações médicas onde os falsos negativos são críticos. Em contraste, a ResNet-50 exibiu uma variabilidade maior, com a precisão caindo para 78,29% ± 2,25% (FedAvg, batch size = 32), enquanto a sensibilidade permaneceu em 77,07% ± 0,66%. Esse desequilíbrio destaca os desafios da arquitetura residual em manter um desempenho consistente em todas as métricas.

Tabela 6.3 – Fase I — Recall vs Precisão

Rede	FL	Batch	Clientes	Precisão	Recall
DenseNet-169	FedAvg	64	3	82,58% ± 1,91%	82,03% ± 5,31%
DenseNet-169	FedProx	32	3	80,12% ± 2,65%	81,01% ± 4,51%
DenseNet-169	FedProx	64	5	78,52% ± 4,18%	78,84% ± 3,08%
ResNet-50	FedAvg	32	3	78,29% ± 2,25%	77,07% ± 0,66%
DenseNet-169	FedAvg	32	3	77,85% ± 4,57%	79,41% ± 6,34%

6.3 Fase II

Após a conclusão do treinamento com o conjunto de dados ampliado, os resultados foram organizados em torno de aspectos analíticos fundamentais. A Tabela 6.4 mostra todos os resultados de desempenho para DenseNet-169 e ResNet-50 durante a Fase II. Para a ResNet-50, o melhor resultado foi alcançado com 3 clientes, *batch size* = 32 e o agregador FedProx, atingindo uma acurácia de 99,26% ± 0,73%, F1-score de 99.10% ± 0.96% e AUC de 99,96% ± 0,06%. Essa abordagem permite, portanto, a comparação com os resultados da Fase I, validando as tendências observadas e destacando as mudanças introduzidas pelo aumento do volume de dados.

Tabela 6.4 – Resultados médios (% ± desvio padrão) dos experimentos na Fase II

Arquitetura	FL	Batch	Clientes	ACC (%)	Precisão (%)	Recall (%)	F1 (%)	AUC (%)
ResNet-50	FedAvg	32	3	98,38 ± 0,79	98,21 ± 1,11	98,23 ± 1,21	98,19 ± 0,97	99,94 ± 0,06
ResNet-50	FedAvg	32	5	98,15 ± 1,44	97,63 ± 2,11	98,07 ± 1,77	97,84 ± 1,92	99,45 ± 0,85
ResNet-50	FedAvg	32	7	96,14 ± 1,06	95,76 ± 1,48	95,64 ± 1,47	95,63 ± 1,38	99,75 ± 0,20
ResNet-50	FedAvg	64	3	98,61 ± 1,09	98,11 ± 1,26	98,28 ± 1,68	98,36 ± 1,45	99,67 ± 0,54
ResNet-50	FedAvg	64	5	96,59 ± 1,04	95,88 ± 1,81	96,51 ± 1,26	96,42 ± 1,71	99,49 ± 0,56
ResNet-50	FedAvg	64	7	93,63 ± 0,87	94,21 ± 1,73	92,51 ± 1,11	93,23 ± 1,21	98,88 ± 0,62
ResNet-50	FedProx	32	3	99,26 ± 0,73	99,35 ± 0,57	98,88 ± 1,36	99,10 ± 0,96	99,96 ± 0,06
ResNet-50	FedProx	32	5	97,89 ± 1,18	97,58 ± 1,52	97,63 ± 1,53	97,60 ± 1,53	99,68 ± 0,50
ResNet-50	FedProx	32	7	97,89 ± 0,71	97,60 ± 0,40	97,61 ± 1,68	97,58 ± 1,03	99,61 ± 0,50
ResNet-50	FedProx	64	3	96,50 ± 1,37	96,17 ± 1,39	95,33 ± 1,40	95,70 ± 1,37	99,14 ± 0,66
ResNet-50	FedProx	64	5	89,25 ± 6,39	88,32 ± 7,37	89,80 ± 4,97	88,41 ± 7,24	97,46 ± 3,00
ResNet-50	FedProx	64	7	88,89 ± 1,17	88,54 ± 1,24	88,13 ± 1,80	88,29 ± 1,49	97,42 ± 0,99
ResNet-50	FedAvgM	32	3	96,85 ± 0,65	95,67 ± 0,92	96,78 ± 1,05	98,72 ± 0,85	97,96 ± 0,04
ResNet-50	FedAvgM	32	5	96,42 ± 1,23	95,98 ± 1,89	96,34 ± 1,56	98,12 ± 1,67	97,67 ± 0,72
ResNet-50	FedAvgM	32	7	96,89 ± 0,92	95,45 ± 1,32	96,23 ± 1,29	96,28 ± 1,21	97,82 ± 0,18
ResNet-50	FedAvgM	64	3	96,95 ± 0,94	95,56 ± 1,12	96,67 ± 1,45	98,72 ± 1,28	97,78 ± 0,42
ResNet-50	FedAvgM	64	5	96,23 ± 0,89	95,45 ± 1,67	96,12 ± 1,12	96,67 ± 1,24	97,62 ± 0,34
ResNet-50	FedAvgM	64	7	96,56 ± 0,76	95,23 ± 1,38	96,67 ± 1,24	94,23 ± 1,18	97,12 ± 0,52

(*continua na próxima página*)

Tabela 6.4 – – (continuação)

Arquitetura	FL	Batch	Clientes	ACC (%)	Precisão (%)	Recall (%)	F1 (%)	AUC (%)
DenseNet-169	FedAvg	32	3	98,90 ± 1,25	99,05 ± 0,83	98,68 ± 1,87	98,85 ± 1,38	99,67 ± 0,57
DenseNet-169	FedAvg	32	5	98,15 ± 1,44	97,85 ± 1,73	97,99 ± 1,96	97,91 ± 1,85	99,88 ± 0,16
DenseNet-169	FedAvg	32	7	98,27 ± 1,01	98,47 ± 0,48	97,97 ± 1,58	98,19 ± 1,06	99,90 ± 0,12
DenseNet-169	FedAvg	64	3	98,04 ± 2,11	98,27 ± 1,47	97,72 ± 2,75	97,96 ± 2,15	99,32 ± 1,14
DenseNet-169	FedAvg	64	5	98,00 ± 0,74	98,10 ± 0,66	97,88 ± 0,98	97,98 ± 0,81	99,89 ± 0,10
DenseNet-169	FedAvg	64	7	97,77 ± 1,27	97,71 ± 0,98	97,46 ± 1,99	97,56 ± 1,49	99,71 ± 0,42
DenseNet-169	FedProx	32	3	98,54 ± 1,45	98,86 ± 1,77	97,61 ± 2,09	98,70 ± 1,95	99,50 ± 0,82
DenseNet-169	FedProx	32	5	98,64 ± 1,09	98,67 ± 1,08	98,27 ± 1,61	98,45 ± 1,34	99,64 ± 0,61
DenseNet-169	FedProx	32	7	97,56 ± 1,07	97,54 ± 1,49	96,87 ± 1,38	97,17 ± 1,34	99,16 ± 0,92
DenseNet-169	FedProx	64	3	98,78 ± 1,45	98,83 ± 1,29	98,42 ± 1,99	98,61 ± 1,65	99,86 ± 0,24
DenseNet-169	FedProx	64	5	98,15 ± 1,28	98,12 ± 1,10	97,87 ± 1,78	97,97 ± 1,42	99,94 ± 0,08
DenseNet-169	FedProx	64	7	96,26 ± 1,61	95,52 ± 2,16	96,50 ± 1,56	95,95 ± 1,78	99,27 ± 1,16
DenseNet-169	FedAvgM	32	3	96,45 ± 0,68	93,56 ± 0,78	93,23 ± 1,89	96,89 ± 1,48	98,67 ± 0,68
DenseNet-169	FedAvgM	32	5	96,34 ± 0,61	93,67 ± 0,76	93,89 ± 1,42	96,12 ± 1,08	98,97 ± 0,05
DenseNet-169	FedAvgM	32	7	96,12 ± 0,92	93,89 ± 1,32	93,78 ± 1,29	96,67 ± 1,28	98,75 ± 0,58
DenseNet-169	FedAvgM	64	3	96,78 ± 1,78	93,23 ± 1,21	93,89 ± 3,45	96,45 ± 2,78	98,45 ± 0,87
DenseNet-169	FedAvgM	64	5	96,89 ± 1,48	93,12 ± 1,28	93,23 ± 3,12	96,56 ± 2,18	98,82 ± 0,28
DenseNet-169	FedAvgM	64	7	96,45 ± 1,76	93,34 ± 2,12	93,23 ± 2,45	96,23 ± 2,18	98,45 ± 0,64

6.3.1 Escalabilidade em Função do Número de Clientes

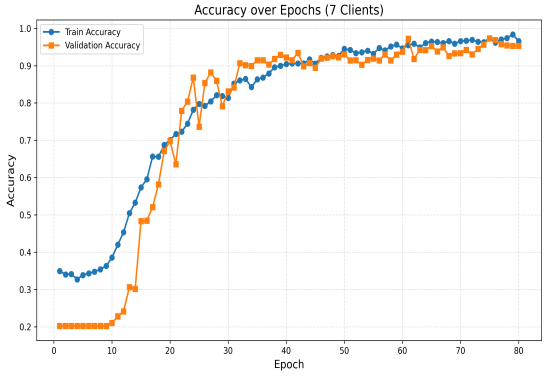
Os resultados da Fase II revelaram uma clara inversão da tendência observada na Fase I. Diferentemente da fase inicial, as arquiteturas alcançaram desempenho superior em federações maiores. Por exemplo, no cenário de 7 clientes, a acurácia da DenseNet consistentemente excedeu 98,27% ± 1,01%, enquanto a ResNet-50 alcançou 97,89% ± 0,71% com 7 clientes. A análise das curvas de validação nas Figuras 6.4 indica que o treinamento com volumes maiores de clientes atingiu maior estabilidade devido ao aumento do volume de dados. Consequentemente, as matrizes de confusão demonstraram um aprendizado do modelo mais preciso, Figuras 6.5.

Esse comportamento sugere que ambas as arquiteturas se beneficiam mais de conjuntos de dados maiores e mais diversificados. Essas descobertas estão alinhadas com estudos anteriores (HSU; QI; BROWN, 2019), que destacam a capacidade da DenseNet e da ResNet de reutilizar características de forma eficiente, mostrando-se particularmente vantajosas quando as federações operam com amostras de dados maiores. Esse comportamento pode estar relacionado à maior capacidade de extração de padrões da DenseNet quando submetida a um volume de dados mais amplo, confirmando a hipótese de que redes mais densamente conectadas se beneficiam de federações com maior diversidade de amostras. Assim, enquanto a fase I havia apontado a ResNet como mais estável à medida que o número de clientes crescia, a fase II sugere que esse cenário se inverte quando há mais instâncias disponíveis. A Tabela 6.5 compila os melhores desempenhos obtidos nesta etapa, evidenciando essa evolução.

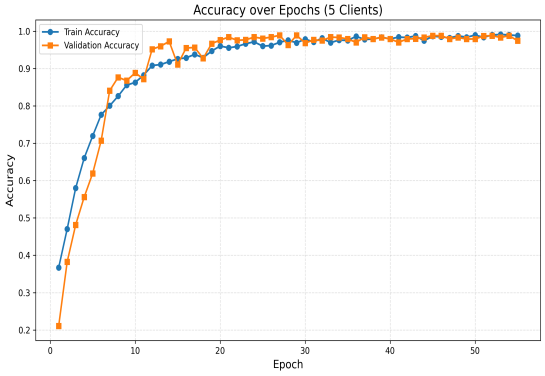
Tabela 6.5 – Melhores resultados da Fase II.

FL	Batch	Rede	Clientes	Acurácia	F1-score	AUC
FedProx	32	ResNet-50	3	99,26% ± 0,73%	99,10% ± 0,96%	99,96% ± 0,06%
FedAvg	32	DenseNet-169	3	98,90% ± 1,25%	98,85% ± 1,36%	99,67% ± 0,57%
FedProx	64	DenseNet-169	5	98,78% ± 1,45%	98,61% ± 1,65%	99,86% ± 0,25%
FedProx	32	DenseNet-169	5	98,64% ± 1,09%	98,45% ± 1,34%	99,64% ± 0,61%
FedProx	64	ResNet-50	3	98,61% ± 1,09%	98,36% ± 1,45%	99,67% ± 0,54%

Cenário 21 & Cenário 33

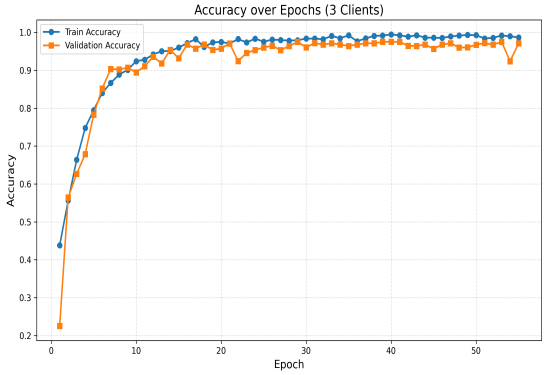


(a) Acurácia - 7 Clientes

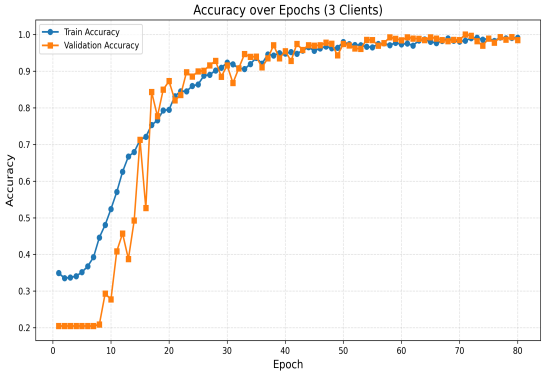


(b) Acurácia - 5 Clientes

Cenário 13 & Cenário 1

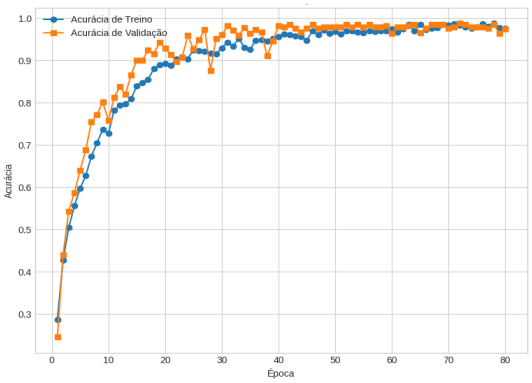


(c) Acurácia - 3 Clientes

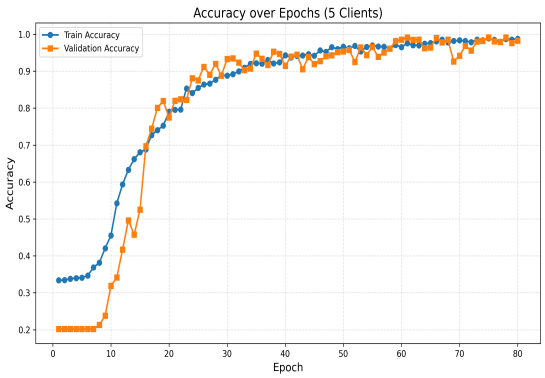


(d) Acurácia - 3 Clientes

Cenário 4 & Cenário 14

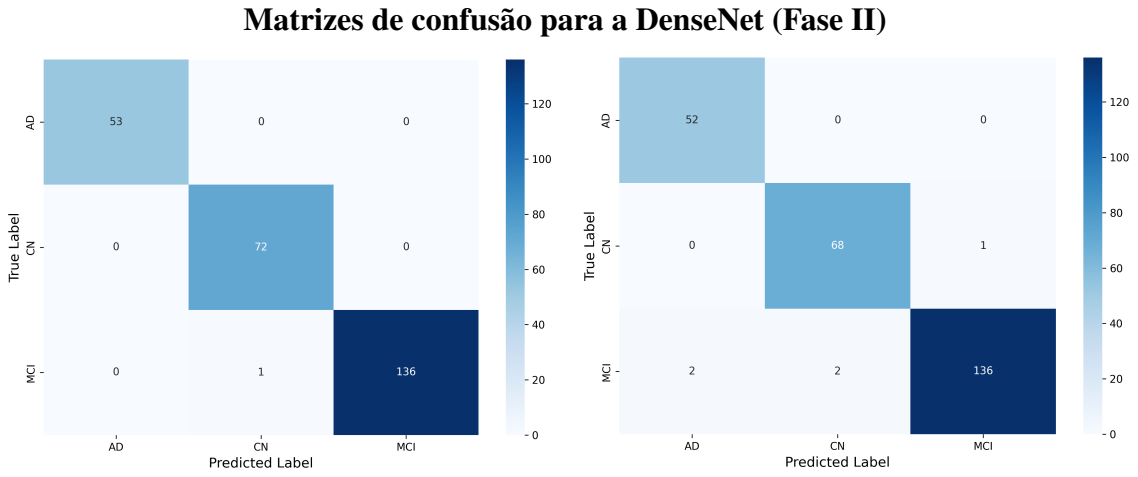


(e) Acurácia - 3 Clientes

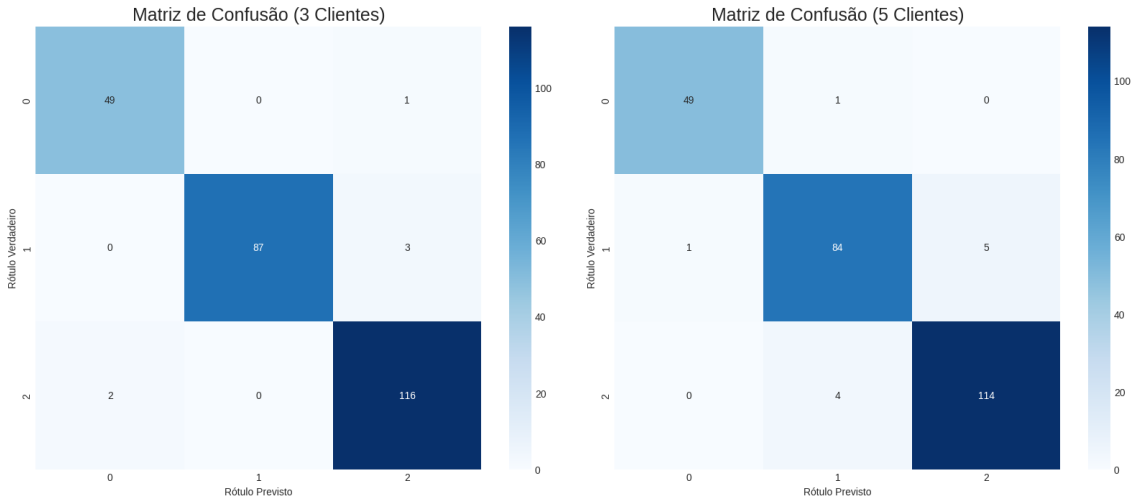


(f) Acurácia - 5 Clientes

Figura 6.4 – Compilação das curvas de Acurácia para cenários selecionados da Fase II.



(a) 3 Clientes. (b) 5 Clientes.



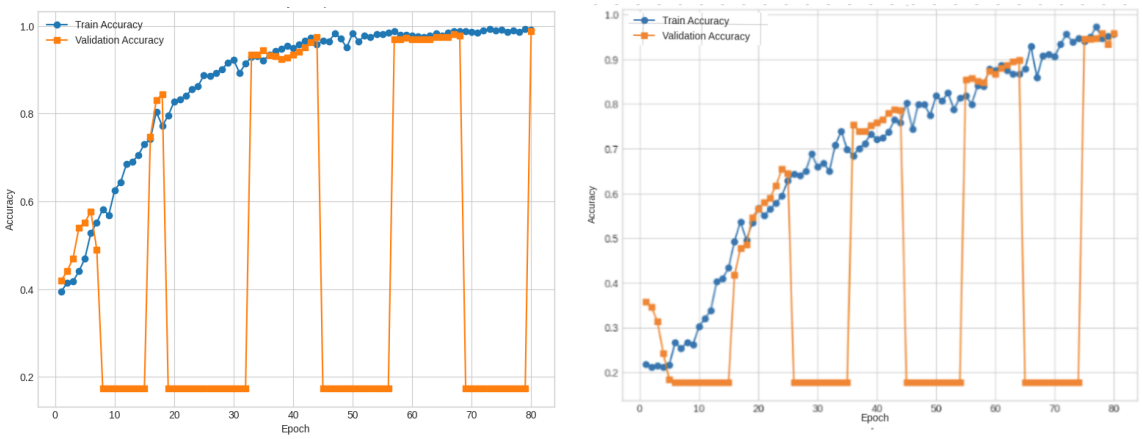
(c) 3 Clientes. (d) 5 Clientes.

Figura 6.5 – Compilação de matrizes de confusão para diferentes combinações de algoritmos de agregação e arquiteturas de modelo, detalhando o desempenho da classificação para 3 e 5 clientes.

6.3.2 Modelos de Agregação

O desempenho dos algoritmos de agregação reforçou tendências já observadas na etapa (i). O FedProx manteve-se com um bom rendimento, mas não foi consistentemente superior. Na DenseNet com 3 clientes e *batch size* = 64, por exemplo, FedProx obteve 98,78% de acurácia e 98,42% de *recall*, superando FedAvg em todas as métricas. Já na ResNet, os resultados foram mais equilibrados: tanto FedAvg quanto FedProx atingiram valores próximos.

Em contraste, o FedAvgM voltou a apresentar instabilidade estrutural, apesar dos valores altos. Observou-se colapso recorrente nas métricas de validação, Figuras 6.6. Esse comportamento confirma a hipótese teórica de que o acúmulo excessivo de momento no servidor amplifica ruídos locais, inviabilizando a convergência estável. Mesmo na amostra ampliada, a DenseNet mostrou ganhos pontuais com FedAvgM, mas insuficientes para caracterizá-lo como alternativa sólida.



(a) ResNet-50. (b) DenseNet-169.

Figura 6.6 – Curvas de validação do FedAvgM - Fase II.

Tabela 6.6 – Fase II — Recall vs Precisão

Rede	FL	Batch	Clientes.	Precisão	Recall
ResNet-50	FedProx	32	3	99,35% ± 0,57%	98,88% ± 1,36%
DenseNet-169	FedAvg	32	3	99,05% ± 0,83%	98,68% ± 1,87%
DenseNet-169	FedProx	64	3	98,83% ± 1,29%	98,42% ± 1,99%
DenseNet-169	FedProx	32	5	98,67% ± 1,08%	98,27% ± 1,61%
ResNet-50	FedAvg	64	3	98,11% ± 1,26%	98,28% ± 1,68%

6.3.3 Impacto do *Batch Size*

O FedProx manteve um forte desempenho, alcançando os melhores resultados sem ser consistentemente dominante em todos os cenários. Para a DenseNet-169 com 3 clientes e um *batch size* = 32, o FedProx atingiu 98,54% ± 1,45% de acurácia e 98,7% ± 1,68% de F1-score. Por outro lado, para a ResNet-50 com 7 clientes e um *batch size* = 64, o FedProx registrou os piores resultados entre os dois algoritmos de agregação estudados, com 88,8% ± 1,17% de acurácia e 88,2% ± 1,49% de F1-score. Em contraste, ao usar o FedAvg, o treinamento demonstrou um desempenho mais equilibrado em todos os cenários analisados.

6.3.4 *Recall* vs Precisão

Com base na Tabela 6.6, constatou-se que federações com um número menor de clientes exibiram maior estabilidade operacional e desempenho superior. Essa descoberta sugere que a complexidade e a variabilidade introduzidas por um grande número de clientes podem afetar negativamente o desempenho do modelo, indicando uma necessidade potencial de estratégias de otimização específicas em ambientes de maior escala.

6.4 Validação Estatística

Para verificar se as melhorias observadas na Fase II não ocorreram por acaso, realizou-se uma comparação estatística entre as duas fases de treinamento[2], cujos relatórios e saídas detalhadas dos testes podem ser consultados no Apêndice B. Como ambas as fases utilizaram as mesmas 24 configurações de modelo, cada caso foi comparado diretamente

[2]Disponível em: <https://github.com/Luanmantegazine/FedAlzheimer/blob/main/validation_stats.py>

com sua contraparte da outra fase. Essa abordagem pareada permitiu-nos focar no efeito do uso de um conjunto de dados maior, mantendo os demais fatores constantes.

Analisaram-se cinco indicadores principais de desempenho: acurácia, precisão, *recall*, F1-score e AUC. Neste contexto, μ representa o desempenho médio dos resultados. Para cada métrica, testou-se se os resultados de ambas as fases eram estatisticamente diferentes. O delineamento inferencial adotou um nível de significância de 5% ($\alpha = 0.05$). Para embasar a escolha adequada dos testes de hipótese, avaliou-se primeiramente o atendimento aos pressupostos paramétricos. Em seguida, aplicou-se o teste de inferência apropriado, avaliou-se a magnitude do ganho por meio de uma métrica de tamanho de efeito e, por fim, empregou-se a correção de Benjamini–Hochberg (HAYNES, 2013) para controlar a inflação de falsos positivos decorrente das múltiplas comparações..

6.4.1 Normalidade das diferenças (Shapiro–Wilk)

O teste de Shapiro-Wilk (SHAPIRO; WILK, 1965) tem como objetivo avaliar se a distribuição dos dados se aproxima de uma distribuição normal, ou seja, atestar que a normalidade dos resultados está coerente. Assim, para cada métrica M e linha i, foi calculada a diferença pareada:

$$d_i = M_{Fase_{II},i} - M_{Fase_{I},i} \tag{6.9}$$

Sendo d_i a média das diferenças entre cada métrica. Ao aplicar o teste, formularam-se duas hipóteses: (H_0) em que as diferenças d_i são normalmente distribuídos e (H_1) onde as diferenças não seguem uma distribuição normal. Os resultados obtidos estão sumarizados na Tabela 6.7:

Métrica	W	*p-values*
Acurácia (Acurácia)	0,533	$1,20 \times 10^{-7}$
Precisão (Precisão)	0,677	$5,07 \times 10^{-6}$
Sensibilidade (*Recall*)	0,708	$1,28 \times 10^{-5}$
F1-score (F1)	0,617	$9,56 \times 10^{-7}$
AUC	0,658	$2,95 \times 10^{-6}$

Tabela 6.7 – Resultados do teste Shapiro-Wilk para as cinco métricas de desempenho. Valores de W (estatística do teste) e p-valores correspondentes.

Conforme exposto na tabela 6.7, os valores de *p-values* foram inferiores a 0.05; sendo assim, a hipótese H_o pode ser rejeitada. Podemos afirmar que os resultados não obedecem

a distribuição normal. A partir desse achado, atesta-se a violação da premissa básica para o uso de testes paramétricos, indicando que a tomada de decisão inferencial deve ser guiada por um teste não paramétrico.

6.4.2 Teste de Wilcoxon

Para comparar o desempenho pareado sem assumir normalidade, aplicou-se o Teste de Wilcoxon (REY; NEUHäUSER, 2011). O teste define a hipótese nula (H_0) de que a mediana das diferenças entre as fases é igual a zero, contra a alternativa (H_1) de que a mediana difere de zero, indicando superioridade de uma das abordagens. O procedimento metodológico envolve o ranqueamento do módulo das diferenças $|d_i|$ e o somatório dos postos associados a diferenças positivas e negativas. A estatística de teste (W) é definida como a menor entre essas duas somas.

No escopo dos 24 experimentos pareados, observou-se que todas as diferenças foram estritamente positivas a favor da Fase II. Como não houve nenhum cenário em que a Fase I superasse a Fase II, a soma dos postos do lado negativo foi zero, produzindo a estatística empírica de $W = 0$ para as cinco métricas analisadas.

Em um teste de Wilcoxon, a obtenção de $W = 0$ representa o resultado mais extremo e significativo possível. Esse achado refuta categoricamente a hipótese nula (H_0) (com *p-values* < 0.001), confirmando estatisticamente que a ampliação do volume de dados em um ambiente federado heterogêneo resultou em ganhos consistentes de classificação em todos os cenários.

6.4.3 Tamanho de Efeito (Correlação de Postos Bisserial)

Para quantificar a magnitude prática dessa superioridade detectada pelo teste de Wilcoxon, utilizou-se a Correlação de Postos Bisserial (r_{rb}) (KERBY, 2014), uma métrica de tamanho de efeito não paramétrica adequada para dados ordinais ou que violam a normalidade. A métrica varia de 0 a 1, onde valores próximos a 1 indicam um efeito prático massivo. Ela é calculada a partir da estatística W:

$$r_{rb} = 1 - \frac{4W}{N(N+1)} \tag{6.10}$$

Dado que o desempenho do modelo na Fase II superou o da Fase I em todas as amostras pareadas ($W = 0$ e $N = 24$), a correlação resultante para todas as cinco métricas foi $r_{rb} = 1,0$. Este tamanho de efeito perfeito corrobora que a evolução das métricas na Fase II não é apenas uma anomalia estatística, mas uma melhoria de magnitude máxima no comportamento preditivo das redes neurais convolucionais avaliadas.

6.4.4 Correção de múltiplas comparações (Benjamini–Hochberg)

Ao conduzir 5 testes de Wilcoxon simultâneos (um para cada métrica), faz-se necessário controlar a Taxa de Falsas Descobertas (False Discovery Rate - FDR) para evitar a inflação da probabilidade de Erro do Tipo I. Utilizou-se o procedimento de Benjamini-Hochberg (BH). O algoritmo segue os seguintes passos:

1. Ordene os *m p-values* obtidos dos testes em ordem crescente: $p_{(1)} \leq p_{(2)} \leq \ldots \leq p_{(m)}$.
2. Para cada valor de k, calcula-se um p-ordenado:

$$p^*(k) = \frac{m}{k} p(k) \tag{6.11}$$

3. Identifique o maior *p-value* ordenados $p_{(k)}$ que satisfaça $p_{(i)} \leq \frac{i \cdot q}{m}$.
4. Declare como estatisticamente significativos todos os testes com *p-values*$\leq p_{(k)}$.

Devido à consistência absoluta dos ganhos ($W = 0$), os *p-values* ordenados permaneceram todos estruturalmente na ordem de $< 0,001$, alocando-se rigorosamente abaixo dos limiares punitivos do procedimento de BH. Portanto, garante-se matematicamente que as cinco constatações de superioridade da Fase II são descobertas estatisticamente eficazes e não representam falsos positivos isolados.

6.5 Comparação com o Estado da Arte

Por fim, após a validação estatística dos resultados locais, faz-se necessário contextualizar os achados desta pesquisa frente à literatura científica atual. Para tanto, elencaram-se estudos recentes que abordaram o mesmo escopo de pesquisa (classificação da AD por meio de neuroimagens) utilizando técnicas de aprendizado profundo e/ou federado.

No estudo conduzido por Guan et al. (2024b), realizou-se uma bateria de testes em um ambiente descentralizado utilizando a base de dados ADNI. Os autores subdividiram o *dataset* em dois subconjuntos focados em classificação binária: ADNI-1 (199 para AD e 229 para CN) e ADNI-2 (159 para AD e 201 para CN). Avaliou-se o desempenho a partir de métricas como acurácia, sensibilidade, especificidade e AUC. Os algoritmos de agregação utilizados foram o FedAvg, o FedSGD (onde os gradientes de cada cliente são agregados para calcular um modelo global) e o FedProx ($\mu = 0.1$).

Tabela 6.8 – Resultados de classificação (média ± desvio padrão) de diferentes configurações de aprendizado federado em termos de quatro métricas.

Cliente	**Método**	**ACC**	**SEN**	**SPE**	**AUC**
ADNI-1	FedAvg	0.860 ± 0.028	0.783 ± 0.025	0.901 ± 0.049	0.897 ± 0.013
	FedSGD	0.823 ± 0.030	0.752 ± 0.047	0.882 ± 0.011	0.880 ± 0.034
	FedProx	0.858 ± 0.031	0.815 ± 0.077	0.896 ± 0.034	0.900 ± 0.044
ADNI-2	FedAvg	0.842 ± 0.021	0.823 ± 0.018	0.864 ± 0.038	0.907 ± 0.017
	FedSGD	0.844 ± 0.039	0.819 ± 0.066	0.871 ± 0.056	0.908 ± 0.040
	FedProx	0.856 ± 0.045	0.845 ± 0.072	0.861 ± 0.047	0.908 ± 0.036

Fonte: Adaptado de (GUAN et al., 2024b)

Ao comparar a Tabela 6.8 com os resultados do modelo proposto nesta dissertação, nota-se uma superioridade em todas as métricas, mesmo com GUAN et al. utilizando cenários inerentemente mais simples (divisão de apenas duas classes e federação com dois clientes). Este desempenho atesta a eficiência das redes neurais escolhidas, com tanto a ResNet quanto a DenseNet obtendo resultados altamente competitivos em um problema multiclasse mais complexo.

Outro estudo relevante no estado da arte foi elaborado por Lei et al. (2024), que propuseram uma rede neural adaptada para ambientes federados intitulada BANet. Os autores realizaram treinamentos com dados volumétricos em 3D oriundos de cinco conjuntos de dados diferentes, totalizando 2.644 amostras. A classificação também foi conduzida de forma binária, utilizando o agregador FedAvg.

Examinando os resultados da Tabela 6.9, constata-se que o modelo proposto nesta dissertação obteve ganhos sobre todas as redes neurais avaliadas pelos autores. Contudo, é importante ressaltar as diferenças estruturais das abordagens. Ao utilizar dados volumétricos (3D), o modelo de LEI et al. aprende a estrutura completa do cérebro, analisando a atrofia do hipocampo em profundidade. Além disso, a arquitetura BANet força a rede a focar em Regiões de Interesse (ROIs) predefinidas e clinicamente importantes. Portanto, embora o modelo aqui proposto alcance métricas absolutas superiores no desafio da clas-

Tabela 6.9 – Resultados da classificação da arquitetura BANet e de outros modelos tradicionais de aprendizado profundo.

Modelo		**ACC(%)**	**SEN(%)**	**SPE(%)**	**AUC(%)**
AD vs. NC	ResNet18	88.80	88.28	90.32	94.26
	ResNet50	89.39	84.72	94.75	91.64
	ResNext50	86.83	86.15	90.02	94.72
	DenseNet121	87.98	88.01	87.30	90.31
	SENet50	89.61	91.04	89.95	95.77
	3DAN	89.15	88.21	89.70	91.05
	BANet	90.20	89.86	92.45	96.23
MCI vs. NC	ResNet18	65.78	68.41	65.90	68.69
	ResNet50	65.20	65.32	67.77	73.60
	ResNext50	60.64	62.03	58.64	75.02
	DenseNet121	64.30	66.95	67.48	70.96
	SENet50	63.84	64.48	65.31	77.54
	3DAN	64.80	62.88	66.44	72.82
	BANet	66.55	66.67	70.64	75.93
AD vs. MCI	ResNet18	70.83	72.66	69.43	79.60
	ResNet50	71.20	70.61	71.88	75.77
	ResNext50	69.31	68.13	70.19	78.51
	DenseNet121	71.05	73.69	70.30	80.10
	SENet50	70.74	70.63	71.77	79.74
	3DAN	70.88	72.56	69.34	79.88
	BANet	72.13	70.56	73.50	81.58

Fonte: Adaptado de (LEI et al., 2024)

sificação multiclasse, a abordagem geométrica orientada a ROIs do BANet mantém uma alta representatividade patológica para o diagnóstico.

Por último, buscou-se uma comparação com modelos centralizados, com a finalidade de corroborar a eficácia da implementação do FL na área da saúde. Sun et al. (2021) elaboraram um modelo aprimorado de ResNet-50 para o diagnóstico de AD, classificando os pacientes nas mesmas três classes abordadas neste estudo (AD, MCI e CN) em um conjunto total de 515 amostras. O objetivo da pesquisa centralizada foi otimizar a arquitetura substituindo a função de ativação ReLU pela Mish, aprimorando a invariância espacial do modelo juntamente com a adição de um mecanismo de atenção.

A referida abordagem centralizada e otimizada alcançou uma acurácia de 97,1%. Esse valor encontra-se ligeiramente abaixo do pico de 99,26% atingido pela melhor configuração do modelo descentralizado proposto neste trabalho. Cabe ressaltar que tal superioridade demonstra uma das principais vantagens operacionais do FL: a capacidade de agregar e extrair conhecimento de um volume consideravelmente maior de dados (como os 1.287 pacientes da Fase II) sem comprometer a privacidade das informações locais.

Tabela 6.10 – Comparação de desempenho da arquitetura ResNet-50 aprimorada com ativação Mish.

Modelo	**ACC**	**SPE$_{macro}$**	**SEN$_{macro}$**	**F1$_{macro}$**
ResNet50 baseline	0.913 ± 0.035	0.871 ± 0.033	0.887 ± 0.032	0.879
ResNet50 + Mish	0.932 ± 0.032	0.893 ± 0.030	0.924 ± 0.028	0.908
STN + ResNet50 + Mish	0.951 ± 0.021	0.940 ± 0.023	0.939 ± 0.021	0.939
Proposed method	0.971 ± 0.016	0.955 ± 0.015	0.953 ± 0.018	0.954

Fonte: Adaptado de (SUN et al., 2021)

Assim, ao se comparar com o *baseline* da ResNet centralizada, a arquitetura federada mostra-se não apenas competitiva, mas altamente recomendada, provando ser mais eficaz quando submetida a redes colaborativas de maior escala.

6.6 Considerações finais

Neste capítulo, foram detalhados os experimentos, as configurações utilizadas e os resultados alcançados. A análise subsequente evidenciou que o modelo adaptado obteve desempenho amplamente satisfatório e consistente, superando diversos trabalhos recentes no estado da arte. Tais resultados atestam o alcance dos objetivos propostos, demonstrando a viabilidade, a segurança e o alto poder preditivo da implementação de redes neurais convolucionais em ecossistemas de Aprendizado Federado para o apoio ao diagnóstico médico.

7 CONCLUSÃO

O objetivo deste trabalho foi avaliar o desempenho de diferentes arquiteturas de redes neurais e algoritmos de agregação em um ambiente híbrido *Federated Learning + Split Learning*, verificando de que forma o número de clientes, *batch size* e a escolha do otimizador afetam a precisão e a estabilidade durante o treinamento distribuído de imagens de MRI para classificação dos estágios da Doença de Alzheimer.

Para validar essa hipótese, foram conduzidos 216 experimentos no Google Colab (GPU NVIDIA Tesla T4) usando o framework PyTorch. Os dados do ADNI foram divididos em duas amostras e distribuídos entre 3, 5 e 7 clientes, avaliando duas CNNs (ResNet-50 e DenseNet-169) combinadas com três algoritmos de agregação: FedAvg, FedAvgM e FedProx, e dois tamanhos de *batch size*. Em cada cenário foram registradas métricas de acurácia, AUC, precisão, *recall*, F1-score e dispersão de desempenho por cliente. Os resultados revelaram que:

- FedProx demonstrou desempenho superior na Fase II, especialmente com *batch size*=32, mas FedAvg foi mais eficaz na Fase I com dados limitados.

- FedAvgM sofre de uma anomalia de acumulação de *momentum*. Nossos experimentos fornecem evidência empírica de que o FedAvgM pode amplificar, em vez de mitigar, a divergência em cenários non-IID. Atualizações ruidosas são desproporcionalmente amplificadas no servidor, causando degradação substancial no desempenho de validação.

- Com dados limitados (Fase I), a DenseNet-169 demonstrou maior robustez, mas a ResNet-50 alcançou desempenho superior quando o volume de dados aumentou (Fase II), especialmente em federações menores (3 clientes).

- FedAvg mantém-se como a opção mais estável, confirmando sua eficiência em diversos cenários, embora geralmente com acurácia global inferior às configurações otimizadas com FedProx.

Assim, este estudo fornece evidências consistentes de que o uso de um termo proximal moderado pode mitigar o *client-drift*, acelerar a convergência e elevar a acurácia global em cenários de dados heterogêneos, contribuindo para a adoção segura e eficiente de *Federated Learning* na área de saúde.

7.1 Contribuições

Este trabalho contribuiu para o avanço do estado da arte ao realizar uma revisão sistemática da literatura (RSL) sobre o uso de *Federated Learning* (FL) aplicado à detecção e classificação da AD, identificando lacunas importantes na avaliação integrada dessas abordagens em cenários médicos heterogêneos.

A partir da definição dos objetivos, questões de pesquisa e estratégias metodológicas da RSL, foram selecionados 23 artigos, nos quais foi possível mapear modelos de FL aplicados à AD, identificar as métricas mais utilizadas e as arquiteturas neurais predominantes nesse contexto. A análise da RSL revelou dois eixos centrais de investigação: (i) a otimização de hiperparâmetros em ambientes federados e (ii) estratégias para mitigar o viés de treinamento diante da heterogeneidade dos dados.

Este estudo dialoga com ambos os eixos, buscando responder à questão de pesquisa proposta e alcançar os objetivos estabelecidos. As principais contribuições técnicas e metodológicas deste trabalho são:

- A avaliação do impacto da heterogeneidade dos dados e do ajuste de hiperparâmetros em ambientes federados voltados à detecção da AD;
- O desenvolvimento de uma metodologia de seleção e distribuição de dados que reduz a sobrecarga de comunicação entre os dispositivos participantes;
- A proposição de um mecanismo genérico, independente do conjunto de dados, para adequar a entrada de elementos ao processamento federado de forma eficiente.

Adicionalmente, os resultados metodológicos e as avaliações empíricas desenvolvidos ao longo desta pesquisa foram validados pela comunidade científica e resultaram nas seguintes publicações derivadas deste trabalho:

- O artigo intitulado *Supervised split federated learning for Alzheimer's disease classification: an evaluation of deep neural networks*, publicado no periódico internacional *Neural Computing and Applications* (Springer) (MANTEGAZINE; GEYER, 2026);
- O artigo *Performance evaluation of neural networks in federated learning for classification of Alzheimer's disease stages*, apresentado no Congresso da Sociedade Brasileira de Computação (CSBC) e publicado nos anais do Seminário Integrado de Software e Hardware (SEMISH) (MANTEGAZINE et al., 2025).

7.2 Trabalhos Futuros

A partir da análise do escopo integral dos resultados obtidos nesta pesquisa, abrem-se diversas frentes de investigação capazes de consolidar e expandir os achados documentados. Diante disso, delineiam-se as seguintes direções principais para trabalhos futuros:

- Investigar variantes do modelo de agregação FedAvgM (tais como FedGM e FedNAG), avaliando de que forma diferentes combinações dos hiperparâmetros ν, β e η influenciam a estabilidade e a convergência do treinamento distribuído;
- Monitorar sistematicamente a norma do vetor de momento (ν) a cada rodada de comunicação, visando estabelecer correlações diretas entre os picos de magnitude de ν e as eventuais quedas de desempenho observadas no modelo global;
- Incorporar subtipos específicos de Comprometimento Cognitivo Leve (MCI), diferenciando empiricamente os estágios precoce (eMCI) e tardio (lMCI), com o intuito de elevar a complexidade da tarefa de classificação e ampliar sua relevância clínica e diagnóstica;
- Analisar o comportamento preditivo do ambiente federado utilizando arquiteturas neurais híbridas, que integrem simultaneamente conexões residuais e densas, com o propósito de maximizar a extração e a reutilização de características espaciais superficiais e profundas.

REFERÊNCIAS

AGARWAL, D. et al. Transfer Learning for Alzheimer's Disease through Neuroimaging Biomarkers: A Systematic Review. **Sensors**, v. 21, n. 21, p. 7259, 2021. Disponível em: <https://doi.org/10.3390/s21217259>.

ALBERDI, A.; AZTIRIA, A.; BASARAB, A. On the early diagnosis of Alzheimer's disease from multimodal signals: A survey. In: **2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)**. [s.n.], 2016. p. 3463–3468. Disponível em: <https://doi.org/10.1109/SMC.2016.7844769>.

ALMEIDA, C. R. de; PEREIRA, A. B. C. N. G. Análise do panorama epidemiológico brasileiro da doença de Alzheimer de 2008 a outubro de 2020. **Revista de Saúde**, v. 13, n. 1, p. 54–60, 2022. Disponível em: <https://doi.org/10.21727/rs.v13i1.2829>.

ALRAWAHNA, A. et al. Federated learning for critical electrical infrastructure — handling data heterogeneity for predictive maintenance. **Frontiers in Artificial Intelligence**, 2025. Disponível em: <https://doi.org/10.3389/frai.2025.1697175>.

ALZHEIMER'S ASSOCIATION. 2018 Alzheimer's disease facts and figures. **Alzheimer's & Dementia**, v. 14, n. 3, p. 367–429, 2018. Disponível em: <https://doi.org/10.1016/j.jalz.2018.02.001>.

ALZHEIMER'S DISEASE NEUROIMAGING INITIATIVE. **ADNI dataset**. 2024. <http://adni.loni.usc.edu>. Accessed on: 2024.

BANABILAH, S. et al. Federated learning review: Fundamentals, enabling technologies, and future applications. **Information Processing & Management**, v. 59, n. 6, p. 103061, 2022. Disponível em: <https://doi.org/10.1016/j.ipm.2022.103061>.

BASNIN, N. et al. An Evolutionary Federated Learning Approach to Diagnose Alzheimer's Disease Under Uncertainty. **Diagnostics**, v. 15, n. 1, p. 80, 2025. Disponível em: <https://doi.org/10.3390/diagnostics15010080>.

CAMOZZATO, A. L. et al. Validity of the Brazilian version of the Neuropsychiatric Inventory Questionnaire (NPI-Q). **Arquivos de Neuro-psiquiatria**, v. 73, n. 1, p. 41–45, 2015. Disponível em: <https://doi.org/10.1590/0004-282X20140177>.

CASELLA, B.; FONIO, S. Architecture-Based FedAvg for Vertical Federated Learning. In: **2023 IEEE/ACM 16th International Conference on Utility and Cloud Computing (UCC)**. [s.n.], 2023. p. 360–365. Disponível em: <https://doi.org/10.1109/UCC58833.2023.00062>.

CASTRO, F.; IMPEDOVO, D.; PIRLO, G. A Federated Learning System with Biometric Medical Image Authentication for Alzheimer's Diagnosis. In: **Proceedings of the 13th International Conference on Pattern Recognition Applications and Methods - ICPRAM**. [s.n.], 2024. p. 951–960. Disponível em: <https://doi.org/10.5220/0012399900003654>.

CHENG, Z. et al. Momentum Benefits Non-IID Federated Learning: Simply and Provably. In: **Proceedings of the International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2024.

DEPARTAMENTO CIENTÍFICO DE NEUROLOGIA COGNITIVA E DO ENVELHECIMENTO. Diagnóstico da doença de Alzheimer: recomendações da Academia Brasileira de Neurologia. **Dementia & Neuropsychologia**, v. 16, n. 3 Suppl 1, p. 25–39, 2022. Disponível em: <https://doi.org/10.1590/1980-5764-DN-2022-S102PT>.

ELHADAD, A.; JAMJOOM, M.; ABULKASIM, H. Reduction of NIFTI files storage and compression time using a novel 3D-CNN and Huffman encoding based lossless approach. **Scientific Reports**, v. 14, n. 1, p. 5168, 2024. Disponível em: <https://doi.org/10.1038/s41598-024-54820-4>.

ELSERSY, M. et al. Federated Learning Model for Early Detection of Dementia Using Blood Biosamples. In: **2023 International Conference on Artificial Intelligence and Big Data (AIBThings)**. [s.n.], 2023. p. 1–6. Disponível em: <https://doi.org/10.1109/AIBThings58966.2023.10202167>.

EOM, B. et al. Federated Learning in Prediction of Dementia Stage: An Experimental Study. In: **2023 International Conference on ICT Convergence (ICTC)**. [s.n.], 2023. p. 1785–1788. Disponível em: <https://doi.org/10.1109/ICTC59104.2023.10394982>.

FARINA, F. et al. A comparison of resting-state EEG and structural MRI for classifying Alzheimer's disease and mild cognitive impairment. **NeuroImage**, v. 215, p. 116795, 2020. Disponível em: <https://doi.org/10.1016/j.neuroimage.2020.116795>.

FAZEKAS, F. et al. MR signal abnormalities at 1.5 T in Alzheimer's dementia and normal aging. **AJR. American Journal of Roentgenology**, v. 149, n. 2, p. 351–356, 1987. Disponível em: <https://doi.org/10.2214/ajr.149.2.351>.

FETAL-NEONATAL NEUROIMAGING & DEVELOPMENTAL SCIENCE CENTER. **med2image: converts medical images to more displayable formats**. [S.l.]: GitHub, 2022. <https://github.com/FNNDSC/med2image>.

FUKUSHIMA, K.; MIYAKE, S. Neocognitron: a new algorithm for pattern recognition tolerant of deformations and shifts in position. **Pattern Recognition**, v. 15, n. 6, p. 455–469, 1982. Disponível em: <https://doi.org/10.1016/0031-3203(82)90024-3>.

GHOSH, A.; KRISHNAMOORTHY, P. DifEvoDenseFed: Differential Evolution Optimization based DenseNet Federated Model in Alzheimer's detection. In: **2023 International Conference on Networking and Network Applications (NaNA)**. [s.n.], 2023. p. 102–107. Disponível em: <https://doi.org/10.1109/NaNA60520.2023.10279624>.

GUAN, H. et al. Federated learning for medical image analysis: A survey. **Pattern Recognition**, v. 150, p. 110424, 2024. Disponível em: <https://doi.org/10.1016/j.patcog.2024.110424>.

GUAN, H. et al. **Federated Learning for Medical Image Analysis: A Survey**. 2024. Disponível em: <https://arxiv.org/abs/2306.05980>.

HAYNES, W. Benjamini–hochberg method. In: DUBITZKY, W. et al. (Ed.). **Encyclopedia of Systems Biology**. New York, NY: Springer, 2013. Disponível em: <https://doi.org/10.1007/978-1-4419-9863-7_1215>.

HE, H.; GARCIA, E. A. Learning from Imbalanced Data. **IEEE Transactions on Knowledge and Data Engineering**, v. 21, n. 9, p. 1263–1284, 2009. Disponível em: <https://doi.org/10.1109/TKDE.2008.239>.

HE, K. et al. Deep residual learning for image recognition. In: **2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [s.n.], 2016. p. 770–778. Disponível em: <https://doi.org/10.1109/CVPR.2016.90>.

HSU, T.-M. H.; QI, H.; BROWN, M. **Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification**. 2019. Disponível em: <https://arxiv.org/abs/1909.06335>.

HUANG, Y.-L.; YANG, H.-C.; LEE, C.-C. Federated Learning via Conditional Mutual Learning for Alzheimer's Disease Classification on T1w MRI. In: **2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)**. [s.n.], 2021. p. 2427–2432. Disponível em: <https://doi.org/10.1109/EMBC46164.2021.9630983>.

IMBOCCIOLI, F.; CIALONE, G.; FERRETTI, S. Blockchain-driven Federated Learning for Alzheimer's MRI Image Classification. In: **2024 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)**. [s.n.], 2024. p. 739–744. Disponível em: <https://doi.org/10.1109/PerComWorkshops59983.2024.10502127>.

KAIROUZ, P. et al. Advances and open problems in federated learning. **Foundations and Trends® in Machine Learning**, v. 14, n. 1-2, p. 1–210, 2021. Disponível em: <https://doi.org/10.1561/2200000083>.

KAPOOR, R. et al. **FedCL-Ensemble Learning: A Framework of Federated Continual Learning for Alzheimer's Disease Detection using Ensemble Learning**. 2024. Disponível em: <https://arxiv.org/abs/2411.12756>.

KERBY, D. S. The simple difference formula: An approach to teaching nonparametric correlation. **Comprehensive Psychology**, SAGE Publications, v. 3, p. 11–IT, 2014.

KHALIL, K. et al. A Federated Learning Model Based on Hardware Acceleration for the Early Detection of Alzheimer's Disease. **Sensors**, v. 23, n. 19, p. 8272, 2023. Disponível em: <https://doi.org/10.3390/s23198272>.

KIM, K. W.; MACFALL, J. R.; PAYNE, M. E. Classification of white matter lesions on magnetic resonance imaging in elderly persons. **Biological Psychiatry**, v. 64, n. 4, p. 273–280, 2008. Disponível em: <https://doi.org/10.1016/j.biopsych.2008.03.024>.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering. 2007. EBSE Technical Report EBSE-2007-01.

KOEDAM, E. L. G. et al. Atrophy of the posterior cingulate cortex is not more severe in Alzheimer's disease than in other neurodegenerative diseases. **Journal of Neurology, Neurosurgery & Psychiatry**, v. 83, n. 4, p. 384–391, 2012. Disponível em: <https://doi.org/10.1136/jnnp-2011-300405>.

LAKHAN, A. et al. FDCNN-AS: Federated deep convolutional neural network Alzheimer detection schemes for different age groups. **Information Sciences**, v. 677, p. 120833, 2024. Disponível em: <https://doi.org/10.1016/j.ins.2024.120833>.

LEI, B. et al. Hybrid federated learning with brain-region attention network for multi-center Alzheimer's disease detection. **Pattern Recognition**, v. 153, p. 110423, 2024. Disponível em: <https://doi.org/10.1016/j.patcog.2024.110423>.

LI, T. et al. FedWCM: Unleashing the potential of momentum-based federated learning in long-tailed scenarios. **arXiv preprint arXiv:2507.14980**, 2025.

LI, T. et al. Federated Optimization in Heterogeneous Networks. In: **Proceedings of the 3rd MLSys Conference**. [s.n.], 2020. Disponível em: <https://arxiv.org/abs/1812.06127>.

LI, W. et al. Privacy-preserving federated brain tumour segmentation. In: **International Workshop on Machine Learning in Medical Imaging (MLMI)**. Cham: Springer, 2019. (Lecture Notes in Computer Science, v. 11861), p. 133–141. Disponível em: <https://doi.org/10.1007/978-3-030-32692-0_16>.

MACEDO, T. d. A.; ROCHA, M. d. S. A radiologia na era da inteligência artificial: desafios e oportunidades. **Radiologia Brasileira**, v. 57, p. e1en, 2024. Disponível em: <https://doi.org/10.1590/0100-3984.2024.57.1e1en>.

MANTEGAZINE, L.; GEYER, C. Supervised split federated learning for alzheimer's disease classification: an evaluation of deep neural networks. **Neural Computing and Applications**, 2026. Disponível em: <https://doi.org/10.1007/s00521-025-11694-5>.

MANTEGAZINE, L. et al. Performance evaluation of neural networks in federated learning for classification of alzheimer's disease stages. In: **Anais do Seminário Integrado de Software e Hardware (SEMISH)**. SBC, 2025. Disponível em: <https://sol.sbc.org.br/index.php/semish/article/view/36797>.

MCMAHAN, H. B. et al. **Federated learning of deep networks using model averaging**. 2016. Disponível em: <https://arxiv.org/abs/1602.05629>.

MHALANKAR, A.; KORDE, A.; KHATRI, R. Enhancing Digital Integrity: A Robust Deepfake Image Detection System. **International Journal of Innovative Research in Technology**, v. 11, n. 11, p. 5109–5115, 2024.

MIHELČIĆ, M. et al. Using redescription mining to relate clinical and biological characteristics of Alzheimer's disease. **PLoS ONE**, v. 12, n. 10, p. e0187364, 2017. Disponível em: <https://doi.org/10.1371/journal.pone.0187364>.

MITROVSKA, A. et al. Secure federated learning for alzheimer's disease detection. **Frontiers in Aging Neuroscience**, v. 16, p. 1324032, 2024. Disponível em: <https://doi.org/10.3389/fnagi.2024.1324032>.

NAEEM, A. et al. A Comprehensive Analysis of Recent Deep- and Federated-Learning-Based Methodologies for Brain Tumor Diagnosis. **Journal of Personalized Medicine**, v. 12, n. 2, p. 275, 2022. Disponível em: <https://doi.org/10.3390/jpm12020275>.

NIMESHIKA, G. N.; SUBITHA, D. Enhancing Alzheimer's disease classification through split federated learning and GANs for imbalanced datasets. **PeerJ Computer Science**, v. 10, p. e2459, 2024. Disponível em: <https://doi.org/10.7717/peerj-cs.2459>.

NITRINI, R. et al. Diagnóstico da doença de Alzheimer: recomendações da Academia Brasileira de Neurologia. **Dementia & Neuropsychologia**, v. 16, n. 3 Suppl 1, p. 25–39, 2022. Disponível em: <https://doi.org/10.1590/1980-5764-DN-2022-S102PT>.

OUYANG, X. et al. ADMarker: A Multi-Modal Federated Learning System for Monitoring Digital Biomarkers of Alzheimer's Disease. In: **The 30th Annual International Conference on Mobile Computing and Networking (MobiCom '24)**. [s.n.], 2024. Disponível em: <https://doi.org/10.1145/3643555.3678385>.

PALOP, J. J.; MUCKE, L. Epilepsy and cognitive impairments in Alzheimer's disease. **Archives of Neurology**, v. 66, n. 4, p. 435–440, 2009. Disponível em: <https://doi.org/10.1001/archneurol.2009.15>.

QU, L.; BALACHANDAR, N.; RUBIN, D. L. **An Experimental Study of Data Heterogeneity in Federated Learning Methods for Medical Imaging**. 2021. Disponível em: <https://arxiv.org/abs/2107.08371>.

RAMESH, J. V. N. et al. A Federated Learning Paradigm for Early Alzheimer's Detection using MRI Imaging. In: **2024 International Conference on Embedded and Software Systems (ICESC)**. [s.n.], 2024. p. 1685–1692. Disponível em: <https://doi.org/10.1109/ICESC60866.2024.10595209>.

RAUNIYAR, A. et al. **Federated learning for medical applications: A taxonomy, current trends, challenges and future research directions**. 2022. Disponível em: <https://arxiv.org/abs/2208.03392>.

REISBERG, B. et al. The Global Deterioration Scale for assessment of primary degenerative dementia. **American Journal of Psychiatry**, v. 139, n. 9, p. 1136–1139, 1982. Disponível em: <https://doi.org/10.1176/ajp.139.9.1136>.

REY, D.; NEUHäUSER, M. Wilcoxon-signed-rank test. In: LOVRIC, M. (Ed.). **International Encyclopedia of Statistical Science**. Berlin, Heidelberg: Springer, 2011. Disponível em: <https://doi.org/10.1007/978-3-642-04898-2_616>.

ROSHAN, J.; ANITHA, J.; JULIET, D. S. Federated Learning for Alzheimer's Diagnosis: Leveraging MRI Data. In: **2024 International Conference on Advanced Computing and Communication Systems (ICACCS)**. [s.n.], 2024. p. 2352–2356. Disponível em: <https://doi.org/10.1109/ICACCS60882.2024.10542159>.

SAHID, M. A. et al. Enhancing Parallelism in Cross-silo Federated Learning for Brain Disease Classification. In: **2024 International Conference on Computer and Automation Engineering (COMPAS)**. [s.n.], 2024. Disponível em: <https://doi.org/10.1109/COMPAS61241.2024.10542475>.

SCHELTENS, P. et al. Atrophy of the medial temporal lobe on MRI in Alzheimer's disease and normal aging. **Journal of Neurology, Neurosurgery & Psychiatry**, v. 58, n. 4, p. 505–509, 1995. Disponível em: <https://doi.org/10.1136/jnnp.58.4.505>.

SCHILLING, L. P. et al. Diagnosis of Alzheimer's disease: recommendations of the Scientific Department of Cognitive Neurology and Aging of the Brazilian Academy of Neurology. **Dementia & Neuropsychologia**, v. 16, n. 3 Suppl 1, p. 25–39, 2022. Disponível em: <https://doi.org/10.1590/1980-5764-DN-2022-S102EN>.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). **Biometrika**, Oxford University Press, v. 52, n. 3/4, p. 591–611, 1965.

SONG, F. et al. HSGO: Harmonized Swarm Learning with Guided Optimization for Alzheimer's Disease Diagnosis. **IEEE Journal of Biomedical and Health Informatics**, 2025. Accepted for publication. Disponível em: <https://doi.org/10.1109/JBHI.2024.3431718>.

SPERLING, R. A. et al. Toward defining the preclinical stages of Alzheimer's disease. **Alzheimer's & Dementia**, v. 7, n. 3, p. 280–292, 2011. Disponível em: <https://doi.org/10.1016/j.jalz.2011.03.003>.

SRINIDHI, V.; VICTOR, A. A Federated Deep Learning Method for Chronic Disease Diagnosis. In: **2023 4th Global Conference for Advancement in Technology (GCAT)**. [s.n.], 2023. Disponível em: <https://doi.org/10.1109/GCAT59993.2023.10328221>.

SRINIVASU, P. N. et al. Enhancing medical image classification via federated learning and pre-trained model. **Egyptian Informatics Journal**, v. 27, p. 100530, 2024. Disponível em: <https://doi.org/10.1016/j.eij.2024.100530>.

SUN, H. et al. An improved deep residual network prediction model for the early diagnosis of alzheimer's disease. **Sensors**, v. 21, n. 12, p. 4182, 2021. Disponível em: <https://doi.org/10.3390/s21124182>.

THAPA, C. et al. **SplitFed: When Federated Learning Meets Split Learning**. 2022. Disponível em: <https://arxiv.org/abs/2004.12088>.

TRIVEDI, N. K.; JAIN, S.; AGARWAL, S. Identifying and Categorizing Alzheimer's Disease with Lightweight Federated Learning Using Identically Distributed Images. In: **2024 12th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)**. [s.n.], 2024. p. 1–5. Disponível em: <https://doi.org/10.1109/ICRITO61763.2024.10582260>.

VEPAKOMMA, P. et al. **Split learning for health: Distributed deep learning without sharing raw patient data**. 2018. Disponível em: <https://arxiv.org/abs/1812.00564>.

VERMUNT, L. et al. Duration of preclinical, prodromal, and dementia stages of Alzheimer's disease in relation to age, sex, and APOE genotype. **Alzheimer's & Dementia**, v. 15, n. 7, p. 888–898, 2019. Disponível em: <https://doi.org/10.1016/j.jalz.2019.04.001>.

WALLACH, E.; SILER, S.; DENG, J. **The More is not the Merrier: Investigating the Effect of Client Size on Federated Learning**. 2025. Disponível em: <https://arxiv.org/abs/2504.08198>.

WEI, K. et al. Federated learning with differential privacy: Algorithms and performance analysis. **IEEE Transactions on Information Forensics and Security**, v. 15, p. 3454–3469, 2020. Disponível em: <https://doi.org/10.1109/TIFS.2020.2988575>.

WEI, W. et al. FedCPC: An Effective Federated Contrastive Learning Method for Privacy-Preserving Early-Stage Alzheimer's Speech Detection. In: **2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)**. [s.n.], 2023. p. 1–8. Disponível em: <https://doi.org/10.1109/ASRU57964.2023.10389959>.

XU, Y. et al. Systematic review of coexistent epileptic seizures and Alzheimer's disease. **Journal of the American Geriatrics Society**, v. 69, n. 7, p. 2011–2020, 2021. Disponível em: <https://doi.org/10.1111/jgs.17144>.

ZHOU, S. K. et al. A review of deep learning in medical imaging: imaging traits, technology trends, case studies with progress highlights, and future promises. **Proceedings of the IEEE**, v. 109, n. 5, p. 820–838, 2021. Disponível em: <https://doi.org/10.1109/JPROC.2021.3054390>.

Apêndice A


```
!pip install torchmetrics
```

```
import os, json, copy, time, random
from pathlib import Path
from collections import Counter, defaultdict

import numpy as np
import pandas as pd
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchmetrics.functional as F_tm
import torchvision
import torchvision.transforms as transforms
import torchvision.datasets as datasets
import torchvision.models as models
from torch.utils.data import DataLoader, Dataset, Subset, Sampler
from sklearn.model_selection import StratifiedGroupKFold
from sklearn.utils.class_weight import compute_class_weight
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

```
SEED = 12
random.seed(SEED)
np.random.seed(SEED)
torch.manual_seed(SEED)
torch.cuda.manual_seed(SEED)
if torch.cuda.is_available():
    torch.backends.cudnn.deterministic = True
    torch.backends.cudnn.benchmark = False
    print(f"Usando GPU: {torch.cuda.get_device_name(0)}")

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# ajuste para seu caminho
data_path = '/content/drive/MyDrive/ADNI4'
```

```
Usando GPU: Tesla T4
```

```
os.makedirs("artifacts", exist_ok=True)

experiment_config = {
    "experiment_name": "SlitFedAlzheimer_FedAvg",
    "threat_model": {
        "adversary_type": "passive",
        "knowledge": "white-box",
        "compromised_clients": 0.0,
        "goal": "inference"
    },
    "privacy": {
        "use_dp": False,
        "dp_epsilon": None,
        "dp_delta": None,
        "gradient_clipping": None,
        "secure_aggregation": False
    },
    "reproducibility": {"seed": SEED}
}
with open("artifacts/experiment_config.json","w") as f:
    json.dump(experiment_config, f, indent=2)

CM_EVERY = 5
```

```
class RandomNoise(object):
    def __init__(self, p=0.5, mean=0., std=0.1):
        self.p,self.mean,self.std=p,mean,std
    def __call__(self, tensor):
        if torch.rand(1).item() < self.p:
            return tensor + torch.randn(tensor.size(), device=tensor.device) * self.std + self.mean
        return tensor
    def __repr__(self): return f"{self.__class__.__name__}(p={self.p}, mean={self.mean}, std={self.std})"

class CustomDataset(Dataset):
    def __init__(self, subset, transform=None):
        self.subset, self.transform = subset, transform
    def __getitem__(self, index):
```

Apêndice A


```
            image, label = self.subset[index]
            if self.transform: image = self.transform(image)
            return image, label
        def __len__(self): return len(self.subset)

    train_transform = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.Grayscale(num_output_channels=3),
        transforms.RandomHorizontalFlip(p=0.5),
        transforms.RandomRotation(15),
        transforms.RandomAffine(degrees=10, translate=(0.1,0.1), scale=(0.85,1.15)),
        transforms.ToTensor(),
        RandomNoise(p=0.2, mean=0., std=0.08),
        transforms.Normalize([0.5]*3, [0.5]*3),
    ])
    test_transform = transforms.Compose([
        transforms.Resize((224,224)),
        transforms.Grayscale(num_output_channels=3),
        transforms.ToTensor(),
        transforms.Normalize([0.5]*3,[0.5]*3)
    ])

    full_dataset = datasets.ImageFolder(root=data_path)

    # Paths & labels
    all_samples = full_dataset.samples  # list of (path, class_idx)
    all_paths   = [p for p, _ in all_samples]
    labels_all  = np.array([lbl for _, lbl in all_samples], dtype=int)
    classes = full_dataset.classes
    num_classes = len(classes)
```

```
    import re

    root = Path(data_path)
    _PATTERNS = [
        re.compile(r'(\d{3}_S_\d{4})'),               # 002_S_0295
        re.compile(r'ADNI[_-](\d{3}_S_\d{4})', re.I), # ADNI_002_S_0295
        re.compile(r'PTID[_-]?(\d{3}_S_\d{4})', re.I)
    ]

    def extract_patient_id_from_filename(path_str: str) -> str:
        fname = Path(path_str).name
        for pat in _PATTERNS:
            m = pat.search(fname)
            if m:
                return m.group(1)
        # Fallback: if nothing matches, use the stem => one scan == one "patient"
        return Path(fname).stem

    patient_ids = np.array([extract_patient_id_from_filename(p) for p in all_paths])
    IDX_TO_PATIENT = {i: patient_ids[i] for i in range(len(all_samples))}
```

```
    sgkf = StratifiedGroupKFold(n_splits=5, shuffle=True, random_state=SEED)
    all_idx = np.arange(len(full_dataset))
    train_idx, test_idx = next(sgkf.split(all_idx, y=labels_all, groups=patient_ids))

    # Sanity: patient disjoint
    train_pats = set(patient_ids[train_idx]); test_pats = set(patient_ids[test_idx])
    assert train_pats.isdisjoint(test_pats), "Leakage: patient appears in both train and test!"

    # Datasets
    main_train_dataset = CustomDataset(Subset(full_dataset, train_idx), transform=train_transform)
    main_test_dataset  = CustomDataset(Subset(full_dataset, test_idx),  transform=test_transform)
```

```
    def partition_by_patient_balanced_no_empty(idx_array, idx_to_patient, labels_all, desired_users, seed=1234):
        """
        Patient-exclusive + class-balanced + no empty clients.
        If too few patients, reduces the effective number of users automatically.
        Returns (dict_users, eff_users).
        """
        rng = np.random.default_rng(seed)
        pats = sorted({idx_to_patient[i] for i in idx_array})
        if not pats:
            return {c: set() for c in range(desired_users)}, 0

        pat_to_idxs = defaultdict(list)
        for i in idx_array:
            pat_to_idxs[idx_to_patient[i]].append(i)
```

Apêndice A


```
        def maj_label(p):
            cnt = Counter(labels_all[j] for j in pat_to_idxs[p])
            return int(cnt.most_common(1)[0][0])

        pats_by_class = defaultdict(list)
        for p in pats:
            pats_by_class[maj_label(p)].append(p)
        for c in list(pats_by_class.keys()):
            rng.shuffle(pats_by_class[c])

        eff_users = min(desired_users, len(pats))
        buckets = [set() for _ in range(eff_users)]

        # round-robin by class for diversity
        while any(pats_by_class.values()):
            for c in list(pats_by_class.keys()):
                if pats_by_class[c]:
                    for b in range(eff_users):
                        if pats_by_class[c]:
                            buckets[b].add(pats_by_class[c].pop())

        # fix any empties by redistributing
        all_pats = [p for b in buckets for p in b]
        non_empty = [b for b in buckets if b]
        if len(non_empty) < eff_users:
            buckets = [set() for _ in range(len(non_empty) or 1)]
            for k, p in enumerate(all_pats):
                buckets[k % len(buckets)].add(p)
            eff_users = len(buckets)

        out = {b: set(i for p in buckets[b] for i in pat_to_idxs[p]) for b in range(eff_users)}
        for b in range(eff_users, desired_users):  # padding
            out[b] = set()
        return out, eff_users

    def class_dist_for(idxs, labels_all, classes):
        if len(idxs) == 0:
            return {c: 0 for c in classes}
        arr = np.array([labels_all[i] for i in idxs], dtype=int)
        return dict(zip(classes, np.bincount(arr, minlength=len(classes)).tolist()))
```

```
    class ResNet50_client_side(nn.Module):
        def __init__(self):
            super().__init__()
            resnet = models.resnet50(weights=models.ResNet50_Weights.DEFAULT)
            # freeze early layers
            for name, param in resnet.named_parameters():
                if name.startswith(("conv1", "bn1")):
                    param.requires_grad = False
            self.features = nn.Sequential(
                resnet.conv1, resnet.bn1, resnet.relu, resnet.maxpool, resnet.layer1, resnet.layer2
            )
        def forward(self, x):
            return self.features(x)

    class ResNet50_server_side(nn.Module):
        def __init__(self, num_classes=3):
            super().__init__()
            resnet = models.resnet50(weights=models.ResNet50_Weights.DEFAULT)
            self.features = nn.Sequential(resnet.layer3, resnet.layer4)
            self.pool = nn.AdaptiveAvgPool2d((1, 1))
            num_ftrs = resnet.fc.in_features
            self.classifier = nn.Sequential(nn.Dropout(p=0.5), nn.Linear(num_ftrs, num_classes))
        def forward(self, x):
            x = self.features(x)
            x = self.pool(x)
            x = torch.flatten(x,1)
            return self.classifier(x)

    class CombinedModel(nn.Module):
        def __init__(self, client_model, server_model):
            super().__init__()
            self.client_model=client_model
            self.server_model=server_model
        def forward(self, x):
            return self.server_model(self.client_model(x))
```

```
    class DenseNet169_client_side(nn.Module):
        def __init__(self):
            super().__init__()
```

Apêndice A


```
        dn = models.densenet169(weights=models.DenseNet169_Weights.DEFAULT)
        feats = dn.features  # Sequential
        # congela bem o começo
        for n, p in dn.named_parameters():
            if any(n.startswith(k) for k in ["features.conv0", "features.norm0"]):
                p.requires_grad = False
        self.features = nn.Sequential(
            feats.conv0, feats.norm0, feats.relu0, feats.pool0,
            feats.denseblock1, feats.transition1,
            feats.denseblock2
        )
    def forward(self, x):
        return self.features(x)

class DenseNet169_server_side(nn.Module):
    def __init__(self, num_classes=3):
        super().__init__()
        dn = models.densenet169(weights=models.DenseNet169_Weights.DEFAULT)
        feats = dn.features
        self.features_tail = nn.Sequential(
            feats.transition2,
            feats.denseblock3, feats.transition3,
            feats.denseblock4, feats.norm5
        )
        self.pool = nn.AdaptiveAvgPool2d((1, 1))
        num_ftrs = dn.classifier.in_features  # 1664
        self.classifier = nn.Linear(num_ftrs, num_classes)

    def forward(self, x):
        x = self.features_tail(x)
        x = F.relu(x, inplace=True)
        x = self.pool(x)
        x = torch.flatten(x, 1)
        return self.classifier(x)

def make_models(backbone: str, num_classes: int):
    b = backbone.lower()
    if b == "resnet50":
        net_glob_client = ResNet50_client_side().to(device)
        net_glob_server = ResNet50_server_side(num_classes=num_classes).to(device)
    elif b == "densenet169":
        net_glob_client = DenseNet169_client_side().to(device)
        net_glob_server = DenseNet169_server_side(num_classes=num_classes).to(device)
    else:
        raise ValueError(f"Unknown backbone: {backbone}")
    return net_glob_client, net_glob_server
```

```
class BalancedBatchSampler(Sampler):
    def __init__(self, dataset: CustomDataset, n_classes: int, n_per_class: int):
        self.labels = [lbl for _, lbl in dataset]
        self.class_to_idxs = defaultdict(list)
        for i, y in enumerate(self.labels):
            self.class_to_idxs[int(y)].append(i)

        max_len = max((len(v) for v in self.class_to_idxs.values() if len(v) > 0), default=0)
        if max_len == 0:
            self.length = 0
            self.n_classes = n_classes
            self.n_per_class = n_per_class
            return

        for c in range(n_classes):
            if len(self.class_to_idxs[c]) == 0:
                for alt in range(n_classes):
                    if len(self.class_to_idxs[alt]) > 0:
                        self.class_to_idxs[c] = [self.class_to_idxs[alt][0]]
                        break
            while len(self.class_to_idxs[c]) < max_len:
                self.class_to_idxs[c].extend(self.class_to_idxs[c])

        self.n_classes = n_classes
        self.n_per_class = n_per_class
        self.length = max_len // n_per_class  # inteiro OK

    def __len__(self):
        return self.length

    def __iter__(self):
        import random as _random
        if self.length == 0:
            return iter([])
        class_iters = {c: iter(_random.sample(idxs, len(idxs))) for c, idxs in self.class_to_idxs.items()}
```

Apêndice A


```
            for _ in range(self.length):
                batch = []
                for c in range(self.n_classes):
                    for _ in range(self.n_per_class):
                        try:
                            batch.append(next(class_iters[c]))
                        except StopIteration:
                            class_iters[c] = iter(self.class_to_idxs[c])
                            batch.append(next(class_iters[c]))
                yield batch
```

```
    class EarlyStopping:
        def __init__(self,patience=20,verbose=True,delta=0,save_path='chkp.pt'):
            self.patience,self.verbose,self.delta,self.save_path=patience,verbose,delta,save_path
            self.counter,self.best_score,self.early_stop,self.val_loss_min=0,None,False,np.Inf
        def __call__(self,val_loss,models):
            score=-val_loss; net_client, net_server = models
            if self.best_score is None:
                self.best_score=score; self.save_checkpoint(val_loss, net_client, net_server)
            elif score<self.best_score+self.delta:
                self.counter+=1
                if self.verbose: print(f'EarlyStopping counter: {self.counter}/{self.patience}')
                if self.counter>=self.patience: self.early_stop=True
            else:
                self.best_score=score; self.save_checkpoint(val_loss, net_client, net_server); self.counter=0
        def save_checkpoint(self,val_loss,net_client,net_server):
            if self.verbose: print(f'Validation loss decreased ({self.val_loss_min:.6f}-->{val_loss:.6f}). Saving mo
            torch.save({'net_glob_client_state_dict':net_client.state_dict(),
                        'net_glob_server_state_dict':net_server.state_dict()}, self.save_path)
            self.val_loss_min=val_loss

    def FedAvg_weighted(w_locals, counts):
        """Weighted average by client data size."""
        if not w_locals:
            return {}
        import copy as _copy, torch as _torch
        N = float(sum(counts)) if sum(counts) > 0 else 1.0
        w_avg = _copy.deepcopy(w_locals[0])
        for k in w_avg.keys():
            if not w_avg[k].dtype.is_floating_point:
                continue
            w_avg[k] = sum((w[k].float() * (n / N) for w, n in zip(w_locals, counts)))
        return w_avg

    MAX_GRAD_NORM = 1.0

    def train_server(fx_client, y, net_server, optimizer_server, criterion, num_classes):
        net_server.train(); optimizer_server.zero_grad()
        fx_server = net_server(fx_client)
        loss = criterion(fx_server, y)
        # batch accuracy only
        with torch.no_grad():
            preds = fx_server.argmax(dim=1)
            acc = F_tm.accuracy(preds, y, task='multiclass', num_classes=num_classes).item()
        loss.backward()
        torch.nn.utils.clip_grad_norm_(net_server.parameters(), max_norm=MAX_GRAD_NORM)
        dfx_client = fx_client.grad.clone().detach()
        optimizer_server.step()
        return dfx_client, loss.item(), acc

    def evaluate_loader_aggregated(net_client, net_server, loader, criterion, num_classes, device):
        net_client.eval(); net_server.eval()
        all_logits, all_labels = [], []
        total_loss, n_batches = 0.0, 0

        with torch.no_grad():
            for images, labels in loader:
                images, labels = images.to(device), labels.to(device)
                fx_client = net_client(images)
                fx_server = net_server(fx_client)
                loss = criterion(fx_server, labels)
                total_loss += float(loss.item()); n_batches += 1
                all_logits.append(fx_server.detach().cpu())
                all_labels.append(labels.detach().cpu())

        if n_batches == 0:
            return 0.0, [0.0]*5

        logits = torch.cat(all_logits, dim=0)
        y_true = torch.cat(all_labels, dim=0)
        y_pred = logits.argmax(dim=1)
```

Apêndice A


```
        acc  = F_tm.accuracy(y_pred, y_true, task='multiclass', num_classes=num_classes).item()
        prec = F_tm.precision(y_pred, y_true, average='macro', task='multiclass', num_classes=num_classes).item()
        rec  = F_tm.recall(y_pred, y_true, average='macro', task='multiclass', num_classes=num_classes).item()
        f1   = F_tm.f1_score(y_pred, y_true, average='macro', task='multiclass', num_classes=num_classes).item()
        try:
            auc = F_tm.auroc(torch.softmax(logits, dim=1), y_true, task='multiclass', num_classes=num_classes).item(
        except Exception:
            auc = 0.0

        avg_loss = total_loss / n_batches
        return avg_loss, [acc, prec, rec, f1, auc]

    def evaluate_accuracy(net,loader,device,return_conf_matrix=False,num_classes=3):
        net.eval(); all_preds,all_labels,all_outputs=[],[],[]
        with torch.no_grad():
            for images,labels in loader:
                images,labels=images.to(device),labels.to(device)
                outputs=net(images); _,predicted=torch.max(outputs,1)
                all_outputs.append(outputs.cpu())
                all_preds.extend(predicted.cpu().numpy())
                all_labels.extend(labels.cpu().numpy())
        if len(all_preds) == 0:
            if return_conf_matrix:
                return 0,0,0,0,0, np.zeros((num_classes,num_classes), dtype=int)
            return 0,0,0,0,0
        all_outputs=torch.cat(all_outputs,dim=0)
        all_preds=torch.tensor(all_preds); all_labels=torch.tensor(all_labels)
        accuracy = F_tm.accuracy(all_preds,all_labels,task='multiclass',num_classes=num_classes).item()
        precision=F_tm.precision(all_preds,all_labels,average='macro',task='multiclass',num_classes=num_classes).ite
        recall  = F_tm.recall(all_preds,all_labels,average='macro',task='multiclass',num_classes=num_classes).item()
        f1      = F_tm.f1_score(all_preds,all_labels,average='macro',task='multiclass',num_classes=num_classes).item
        try:
            auc     = F_tm.auroc(F.softmax(all_outputs,dim=1),all_labels,task="multiclass",num_classes=num_classes).
        except ValueError:
            auc = 0.0
        cm=confusion_matrix(all_labels.cpu().numpy(),all_preds.cpu().numpy())
        if return_conf_matrix: return accuracy,precision,recall,f1,auc,cm
        return accuracy,precision,recall,f1,auc

    def _get_indices_from(dataset):
        if isinstance(dataset, Subset):
            return dataset.indices
        if isinstance(dataset, CustomDataset) and isinstance(dataset.subset, Subset):
            return dataset.subset.indices
        raise ValueError("evaluate_by_patient: preciso de Subset(...) ou CustomDataset(Subset(...)).")

    def evaluate_by_patient(model, dataset_or_subset, idx_to_patient_map, batch_size=64, device="cuda"):
        loader = DataLoader(dataset_or_subset, batch_size=batch_size, shuffle=False, num_workers=0)
        model.eval()
        all_probs, all_labels = [], []
        with torch.no_grad():
            for images, labels in loader:
                images = images.to(device)
                logits = model(images)
                probs = torch.softmax(logits, dim=1).cpu().numpy()
                all_probs.append(probs)
                all_labels.extend(labels.numpy())
        all_probs = np.concatenate(all_probs, axis=0)
        original_indices = _get_indices_from(dataset_or_subset)
        pats = [idx_to_patient_map[i] for i in original_indices]

        df = pd.DataFrame(all_probs, columns=[f"p_{c}" for c in range(all_probs.shape[1])])
        df["y"] = np.array(all_labels, dtype=int)
        df["patient"] = pats

        agg = df.groupby("patient").agg({f"p_{c}":"mean" for c in range(all_probs.shape[1])})
        y_true = df.groupby("patient")["y"].agg(lambda x: Counter(x).most_common(1)[0][0]).values
        P = agg.values
        y_pred = P.argmax(axis=1)

        from sklearn import metrics as skm
        acc  = (y_pred == y_true).mean().item()
        prec = skm.precision_score(y_true, y_pred, average="macro", zero_division=0)
        rec  = skm.recall_score(y_true, y_pred, average="macro", zero_division=0)
        f1   = skm.f1_score(y_true, y_pred, average="macro", zero_division=0)
        try:
            from sklearn.metrics import roc_auc_score
            auc = roc_auc_score(y_true, P, multi_class="ovr")
        except Exception:
            auc = 0.0
        cm   = skm.confusion_matrix(y_true, y_pred, labels=list(range(P.shape[1])))
        return acc, prec, rec, f1, auc, cm
```

Apêndice A


```
class Client:
    def __init__(self, idx, lr, device, idxs, idxs_test, net_server_initial_weights, weight_decay,
                 batch_size, num_classes, criterion, mu, use_balanced_batch=True, epochs_total=80, local_ep=1):
        self.idx,self.device,self.lr = idx,device,lr
        self.local_ep = local_ep
        self.num_classes = num_classes
        self.criterion = criterion
        self.mu = mu
        self.epochs_total = epochs_total

        self.train_dataset = CustomDataset(Subset(full_dataset, list(idxs)), transform=train_transform)
        self.test_dataset  = CustomDataset(Subset(full_dataset, list(idxs_test)), transform=test_transform)

        WORKERS = 2

        labels_client = [lbl for _, lbl in self.train_dataset]
        counts = Counter(labels_client)
        n_per_class = max(1, batch_size // num_classes)
        use_balanced = (len(np.unique(labels_client)) == num_classes) and (min(counts.values()) >= n_per_class)

        if use_balanced and use_balanced_batch:
            batch_sampler = BalancedBatchSampler(self.train_dataset, n_classes=num_classes, n_per_class=n_per_cl
            try:
                _ = next(iter(batch_sampler))
                self.ldr_train = DataLoader(self.train_dataset, batch_sampler=batch_sampler,
                                            num_workers=WORKERS, pin_memory=True)
            except StopIteration:
                self.ldr_train = DataLoader(self.train_dataset, batch_size=batch_size, shuffle=True,
                                            num_workers=WORKERS, pin_memory=True)
        else:
            self.ldr_train = DataLoader(self.train_dataset, batch_size=batch_size, shuffle=True,
                                        num_workers=WORKERS, pin_memory=True)

        self.ldr_test = DataLoader(self.test_dataset, batch_size=batch_size, shuffle=False,
                                   num_workers=WORKERS, pin_memory=True)

        self.net_server_local_copy = DenseNet169_server_side(num_classes=num_classes).to(self.device)
        self.net_server_local_copy.load_state_dict(net_server_initial_weights)
        self.optimizer_server_local = torch.optim.Adam(self.net_server_local_copy.parameters(),
                                                       lr=self.lr, weight_decay=weight_decay)
        self.scheduler_server = torch.optim.lr_scheduler.CosineAnnealingLR(self.optimizer_server_local,
                                                                           T_max=self.epochs_total, eta_min=1e-6

    def evaluate(self, net_client, net_server):
        return evaluate_loader_aggregated(net_client, net_server, self.ldr_test,
                                          self.criterion, self.num_classes, self.device)

    def train(self, net_client, epoch_idx=None, progressive_unfreeze=False):
        net_client.train(); net_client.to(self.device)

        if progressive_unfreeze and epoch_idx is not None and epoch_idx >= 10:
            for name, p in net_client.named_parameters():
                if ("denseblock2" in name) or ("layer2" in name) or ("client_model.5" in name):
                    p.requires_grad = True

        client_params = []
        low_lr, high_lr = 1e-5, self.lr
        low_group, high_group = [], []
        for n,p in net_client.named_parameters():
            if not p.requires_grad:
                continue
            if ("denseblock2" in n) or ("layer2" in n) or ("client_model.5" in n):
                low_group.append(p)
            else:
                high_group.append(p)
        if low_group:  client_params.append({"params": low_group, "lr": low_lr})
        if high_group: client_params.append({"params": high_group, "lr": high_lr})
        if not client_params:
            client_params = [{"params": [p for p in net_client.parameters() if p.requires_grad], "lr": self.lr}]

        optimizer_client = torch.optim.Adam(client_params, weight_decay=1e-4)
        scheduler_client = torch.optim.lr_scheduler.CosineAnnealingLR(optimizer_client,
                                                                      T_max=self.epochs_total, eta_min=1e-6)

        global_weights = copy.deepcopy(net_client.state_dict())

        batch_losses, batch_accs = [], []
        for _ in range(self.local_ep):
            for images, labels in self.ldr_train:
                images, labels = images.to(self.device), labels.to(self.device)
                optimizer_client.zero_grad()
```

Apêndice A


```
                fx = net_client(images)
                client_fx = fx.clone().detach().requires_grad_(True)

                dfx, loss, acc = train_server(client_fx, labels, self.net_server_local_copy,
                                              self.optimizer_server_local, self.criterion, self.num_classes)

                fx.backward(dfx)

                if self.mu > 0.0:
                    proximal_term = 0.0
                    for name, param in net_client.named_parameters():
                        if param.requires_grad:
                            proximal_term += torch.sum((param - global_weights[name].to(self.device))**2)
                    ((self.mu/2) * proximal_term).backward()

                torch.nn.utils.clip_grad_norm_(net_client.parameters(), max_norm=1.0)
                optimizer_client.step()

                batch_losses.append(loss); batch_accs.append(float(acc))

        scheduler_client.step(); self.scheduler_server.step()

        avg_loss = sum(batch_losses)/len(batch_losses) if batch_losses else 0.0
        avg_acc  = sum(batch_accs)/len(batch_accs)    if batch_accs  else 0.0
        return net_client.state_dict(), self.net_server_local_copy.state_dict(), avg_loss, [avg_acc,0,0,0,0], le
```

```
def _safe_list(x):
    return x if (isinstance(x, (list, tuple)) and len(x) > 0) else [0.0]

def plot_history(history, num_users, save_dir="artifacts"):
    Path(save_dir).mkdir(parents=True, exist_ok=True)
    epochs_range = range(1, len(_safe_list(history['train_acc'])) + 1)

    # Accuracy
    plt.figure(figsize=(10, 6))
    plt.plot(epochs_range, _safe_list(history['train_acc']), marker='o', linewidth=2, label='Train Accuracy')
    plt.plot(epochs_range, _safe_list(history['test_acc']),  marker='s', linewidth=2, label='Validation Accuracy
    plt.xlabel('Epoch', fontsize=13); plt.ylabel('Accuracy', fontsize=13)
    plt.title(f'Accuracy over Epochs ({num_users} Clients)', fontsize=15)
    plt.legend(); plt.grid(True, linestyle='--', alpha=0.4); plt.tight_layout()
    plt.savefig(f"{save_dir}/accuracy_{num_users}c.png", dpi=300, bbox_inches='tight')
    plt.close()

    # Loss
    plt.figure(figsize=(10, 6))
    plt.plot(epochs_range, _safe_list(history['train_loss']), marker='o', linewidth=2, label='Train Loss')
    plt.plot(epochs_range, _safe_list(history['test_loss']),  marker='s', linewidth=2, label='Validation Loss')
    plt.xlabel('Epoch', fontsize=13); plt.ylabel('Loss', fontsize=13)
    plt.title(f'Loss over Epochs ({num_users} Clients)', fontsize=15)
    plt.legend(); plt.grid(True, linestyle='--', alpha=0.4); plt.tight_layout()
    plt.savefig(f"{save_dir}/loss_{num_users}c.png", dpi=300, bbox_inches='tight')
    plt.close()

    # Recall
    plt.figure(figsize=(10, 6))
    plt.plot(epochs_range, _safe_list(history['train_recall']), marker='o', linewidth=2, label='Train Recall (ma
    plt.plot(epochs_range, _safe_list(history['test_recall']),  marker='s', linewidth=2, label='Validation Recal
    plt.xlabel('Epoch', fontsize=13); plt.ylabel('Recall', fontsize=13)
    plt.title(f'Recall over Epochs ({num_users} Clients)', fontsize=15)
    plt.legend(); plt.grid(True, linestyle='--', alpha=0.4); plt.tight_layout()
    plt.savefig(f"{save_dir}/recall_{num_users}c.png", dpi=300, bbox_inches='tight')
    plt.close()

    # AUC
    plt.figure(figsize=(10, 6))
    plt.plot(epochs_range, _safe_list(history['train_auc']), marker='o', linewidth=2, label='Train AUC (macro)')
    plt.plot(epochs_range, _safe_list(history['test_auc']),  marker='s', linewidth=2, label='Validation AUC (mac
    plt.xlabel('Epoch', fontsize=13); plt.ylabel('AUC', fontsize=13)
    plt.title(f'AUC over Epochs ({num_users} Clients)', fontsize=15)
    plt.legend(); plt.grid(True, linestyle='--', alpha=0.4); plt.tight_layout()
    plt.savefig(f"{save_dir}/auc_{num_users}c.png", dpi=300, bbox_inches='tight')
    plt.close()

def plot_confusion_matrix(cm, class_names, num_users, save_dir="artifacts", suffix=""):
    Path(save_dir).mkdir(parents=True, exist_ok=True)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cbar=True, cmap='Blues',
                xticklabels=class_names, yticklabels=class_names)
    plt.xlabel('Predicted Label', fontsize=13)
    plt.ylabel('True Label', fontsize=13)
```

Apêndice A


```
        plt.title(f'Confusion Matrix ({num_users} Clients){suffix}', fontsize=15)
        plt.tight_layout()
        plt.savefig(f"{save_dir}/confusion_matrix_{num_users}c{suffix}.png", dpi=300, bbox_inches='tight')
        plt.close()

    def summarize_distribution(dict_users, labels_all, classes, num_clients):
        """Build a DataFrame with per-class counts per client for one setup."""
        rows = []
        for c in sorted(dict_users.keys()):
            idxs = list(dict_users[c])
            if len(idxs) == 0:
                counts = {cls: 0 for cls in classes}
            else:
                labs = [int(labels_all[i]) for i in idxs]
                binc = np.bincount(labs, minlength=len(classes)).tolist()
                counts = dict(zip(classes, binc))
            rows.append({
                **counts,
                "client": f"C{c+1}",
                "setup":  f"{num_clients} clients",
                "group":  f"{num_clients}c-C{c+1}"
            })
        return pd.DataFrame(rows)

    def plot_single_distribution_357(snapshots, labels_all, classes, save_path="artifacts/distribution_3_5_7_clients
        """
        Create ONE chart with class distributions per client
        for the 3, 5 and 7 clients setups.
        """
        dfs = []
        for k in sorted(snapshots.keys()):  # k = 3, 5, 7 (if available)
            df_k = summarize_distribution(snapshots[k], labels_all, classes, k)
            dfs.append(df_k)
        if not dfs:
            print("[WARN] No distribution snapshot available for plotting.")
            return
        df_all = pd.concat(dfs, ignore_index=True)

        # Long format
        df_melt = df_all.melt(id_vars=["setup", "client", "group"],
                              value_vars=classes, var_name="Class", value_name="Samples")

        # X-axis order: 3c-C1..C3, 5c-C1..C5, 7c-C1..C7
        def _sort_key(g):
            left, right = g.split("-C")   # "3c", "1"
            return (int(left.replace("c","").strip()), int(right))
        df_melt = df_melt.sort_values(by="group", key=lambda s: s.map(_sort_key))

        plt.figure(figsize=(14, 6))
        sns.barplot(data=df_melt, x="group", y="Samples", hue="Class", dodge=True)
        plt.title("Class distribution per client for 3 / 5 / 7 clients")
        plt.xlabel("Setup–Client (e.g., 3c-C1 = 3 clients, client 1)")
        plt.ylabel("Number of samples")
        plt.xticks(rotation=45, ha="right")
        plt.grid(axis="y", linestyle="--", alpha=0.3)
        plt.tight_layout()
        plt.savefig(save_path, dpi=300, bbox_inches="tight")
        plt.close()
        print(f"[OK] Distribution chart saved at: {save_path}")
```

```
    client_counts = [3,5,7]  # scale experiments
    batch_size = 64
    epochs, frac, lr, weight_decay = 5, 1.0, 1e-4, 1e-4
    mu = 0.01  # FedProx (0.0 => FedAvg)
    progressive_unfreeze = True
    local_ep_default = 1

    # Class weights from global TRAIN
    labels_train_global = [label for _, label in main_train_dataset]
    present_classes = np.unique(labels_train_global)
    num_classes = len(classes)
    full_weights = np.ones((num_classes,), dtype=np.float32)
    if len(present_classes) > 0:
        partial_weights = compute_class_weight(class_weight='balanced', classes=present_classes, y=labels_train_globa
        for c, w in zip(present_classes, partial_weights):
            full_weights[int(c)] = float(w)

    criterion_global = nn.CrossEntropyLoss(weight=torch.tensor(full_weights, dtype=torch.float32, device=device),
                                           label_smoothing=0.05)

    dict_users_snapshots = {}
```

Apêndice A


```
for num_users in client_counts:
    print("\n" + "="*80)
    print(f"STARTING TRAINING FOR {num_users} CLIENTS")
    print("="*80 + "\n")

    # Partition train/test
    dict_users, eff_train = partition_by_patient_balanced_no_empty(train_idx, IDX_TO_PATIENT, labels_all, num_use
    dict_users_test, eff_test = partition_by_patient_balanced_no_empty(test_idx,  IDX_TO_PATIENT, labels_all, num

    # Retry seeds if empties
    tries = 0
    while (any(len(dict_users[c])==0 for c in range(num_users))) and tries < 5:
        tries += 1
        alt_seed = SEED + 100*tries
        dict_users, eff_train = partition_by_patient_balanced_no_empty(train_idx, IDX_TO_PATIENT, labels_all, num
        dict_users_test, eff_test = partition_by_patient_balanced_no_empty(test_idx,  IDX_TO_PATIENT, labels_all,

    train_clients = [c for c in range(num_users) if len(dict_users[c]) > 0]
    empty_clients = [c for c in range(num_users) if len(dict_users[c]) == 0]
    if empty_clients:
        print(f"[Warn] Empty clients (ignored): {empty_clients}")

    # Audit print
    for c in range(num_users):
        tr_dist = class_dist_for(list(dict_users[c]), labels_all, classes)
        te_dist = class_dist_for(list(dict_users_test[c]), labels_all, classes)
        print(f"[TRAIN] client {c}: n={len(dict_users[c])}  dist={tr_dist}")
        print(f"[TEST ] client {c}: n={len(dict_users_test[c])} dist={te_dist}")

    dict_users_snapshots[num_users] = copy.deepcopy(dict_users)

    # Models (DenseNet169 split)
    net_glob_client = DenseNet169_client_side().to(device)
    net_glob_server = DenseNet169_server_side(num_classes=num_classes).to(device)
    w_glob_client = net_glob_client.state_dict()
    w_glob_server = net_glob_server.state_dict()

    history = {
        'train_loss':[], 'train_acc':[], 'test_loss':[], 'test_acc':[],
        'train_recall':[], 'test_recall':[], 'train_auc':[], 'test_auc':[]
    }

    checkpoint_path = f'best_model_{num_users}_clients.pt'
    early_stopping = EarlyStopping(patience=20, verbose=True, save_path=checkpoint_path)

    # ===== TRAIN LOOP =====
    for iter_epoch in range(epochs):
        start_time = time.time()
        m = max(int(frac * len(train_clients)), 1)
        idxs_users = np.random.choice(train_clients, m, replace=False)

        w_locals_client, w_locals_server, ns = [], [], []
        round_train_loss, round_train_metrics = [], []

        # STEP 1: LOCAL CLIENT TRAINING
        for idx in idxs_users:
            local = Client(idx, lr, device, dict_users[idx], dict_users_test[idx],
                           w_glob_server, weight_decay, batch_size, num_classes, criterion_global, mu,
                           use_balanced_batch=True, epochs_total=epochs, local_ep=local_ep_default)
            w_c, w_s, t_loss, t_metrics, n_i = local.train(net_client=copy.deepcopy(net_glob_client),
                                                           epoch_idx=iter_epoch,
                                                           progressive_unfreeze=progressive_unfreeze)
            w_locals_client.append(w_c)
            w_locals_server.append(w_s)
            ns.append(n_i)
            round_train_loss.append(t_loss)
            round_train_metrics.append(t_metrics)

        # STEP 2: SERVER AGGREGATION
        w_glob_client = FedAvg_weighted(w_locals_client, ns)
        w_glob_server = FedAvg_weighted(w_locals_server, ns)
        net_glob_client.load_state_dict(w_glob_client)
        net_glob_server.load_state_dict(w_glob_server)

        # STEP 3: GLOBAL EVALUATION
        round_test_loss, round_test_metrics = [], []
        for idx in idxs_users:
            local = Client(idx, lr, device, dict_users[idx], dict_users_test[idx],
                           w_glob_server, weight_decay, batch_size, num_classes, criterion_global, mu,
                           use_balanced_batch=True, epochs_total=epochs)
            test_loss, test_metrics = local.evaluate(net_glob_client, net_glob_server)
            round_test_loss.append(test_loss)
            round_test_metrics.append(test_metrics)
```

Apêndice A


```
# History
history['train_loss'].append(sum(round_train_loss) / len(round_train_loss) if round_train_loss else 0.0)
if round_train_metrics:
    avg_train_metrics = [sum(col) / len(col) for col in zip(*round_train_metrics)]
else:
    avg_train_metrics = [0.0]*5
history['train_acc'].append(avg_train_metrics[0])
history['train_recall'].append(avg_train_metrics[2])
history['train_auc'].append(avg_train_metrics[4])

history['test_loss'].append(sum(round_test_loss) / len(round_test_loss) if round_test_loss else 0.0)
if round_test_metrics:
    avg_test_metrics = [sum(col) / len(col) for col in zip(*round_test_metrics)]
else:
    avg_test_metrics = [0.0]*5
history['test_acc'].append(avg_test_metrics[0])
history['test_recall'].append(avg_test_metrics[2])
history['test_auc'].append(avg_test_metrics[4])

# Confusion Matrix (Validation) periódico
if ((iter_epoch + 1) % CM_EVERY == 0) or (iter_epoch == 0):
    y_true_all, y_pred_all = [], []
    net_glob_client.eval(); net_glob_server.eval()
    with torch.no_grad():
        for idx in idxs_users:
            local_eval = Client(
                idx, lr, device, dict_users[idx], dict_users_test[idx],
                w_glob_server, weight_decay, batch_size, num_classes, criterion_global, mu,
                use_balanced_batch=True, epochs_total=epochs
            )
            for images, labels in local_eval.ldr_test:
                images = images.to(device)
                logits = net_glob_server(net_glob_client(images))
                preds  = logits.argmax(dim=1).cpu().numpy()
                y_pred_all.append(preds)
                y_true_all.append(labels.numpy())
    if len(y_true_all) > 0:
        y_true = np.concatenate(y_true_all)
        y_pred = np.concatenate(y_pred_all)
        cm_epoch = confusion_matrix(y_true, y_pred, labels=list(range(num_classes)))
        plot_confusion_matrix(
            cm_epoch, classes, num_users,
            save_dir="artifacts",
            suffix=f"_epoch{iter_epoch+1:03d}"
        )

print(
    f"Epoch {iter_epoch+1}/{epochs} | "
    f"Train Loss: {history['train_loss'][-1]:.4f} Acc: {history['train_acc'][-1]:.4f} | "
    f"Test  Loss: {history['test_loss'][-1]:.4f} "
    f"Acc: {history['test_acc'][-1]:.4f} "
    f"Rec: {history['test_recall'][-1]:.4f} "
    f"AUC: {history['test_auc'][-1]:.4f} | "
    f"Time: {time.time()-start_time:.2f}s"
)

early_stopping(history['test_loss'][-1], (net_glob_client, net_glob_server))
if early_stopping.early_stop:
    print("Early stopping triggered.")
    try:
        y_true_all, y_pred_all = [], []
        net_glob_client.eval(); net_glob_server.eval()
        with torch.no_grad():
            for idx in idxs_users:
                local_eval = Client(
                    idx, lr, device, dict_users[idx], dict_users_test[idx],
                    w_glob_server, weight_decay, batch_size, num_classes, criterion_global, mu,
                    use_balanced_batch=True, epochs_total=epochs
                )
                for images, labels in local_eval.ldr_test:
                    images = images.to(device)
                    logits = net_glob_server(net_glob_client(images))
                    preds  = logits.argmax(dim=1).cpu().numpy()
                    y_pred_all.append(preds)
                    y_true_all.append(labels.numpy())
        if len(y_true_all) > 0:
            y_true = np.concatenate(y_true_all)
            y_pred = np.concatenate(y_pred_all)
            cm_last = confusion_matrix(y_true, y_pred, labels=list(range(num_classes)))
            plot_confusion_matrix(cm_last, classes, num_users, save_dir="artifacts", suffix="_last")
    except Exception as e:
        print(f"[WARN] Could not save last confusion matrix: {e}")
```

Apêndice A


```
                print(f"[WARN] Could not save last confusion matrix: {e}")
            break

    print(f"\nTraining for {num_users} clients completed!")
    plot_history(history, num_users, save_dir="artifacts")

    # Avaliação final global (image-level)
    try:
        checkpoint_path = f'best_model_{num_users}_clients.pt'
        checkpoint = torch.load(checkpoint_path, map_location=device)
        net_glob_client.load_state_dict(checkpoint['net_glob_client_state_dict'])
        net_glob_server.load_state_dict(checkpoint['net_glob_server_state_dict'])
        final_model = CombinedModel(net_glob_client, net_glob_server).to(device)

        test_loader_global = DataLoader(main_test_dataset, batch_size=batch_size, shuffle=False)
        acc, prec, rec, f1, auc, cm = evaluate_accuracy(final_model, test_loader_global,
                                                        device, return_conf_matrix=True, num_classes=num_classes)

        print(f"\n--- Final Metrics (image-level, {num_users} Clients) ---")
        print(f"Accuracy: {acc:.4f} | Precision: {prec:.4f} | Recall: {rec:.4f} | F1: {f1:.4f} | AUC: {auc:.4f}")
        plot_confusion_matrix(cm, classes, num_users, save_dir="artifacts", suffix="_image")

        # Patient-level
        try:
            acc_p, prec_p, rec_p, f1_p, auc_p, cm_p = evaluate_by_patient(
                final_model,
                main_test_dataset,
                IDX_TO_PATIENT,
                batch_size=batch_size,
                device=device
            )
            print(f"--- Final Metrics (patient-level, {num_users} Clients) ---")
            print(f"Accuracy: {acc_p:.4f} | Precision: {prec_p:.4f} | Recall: {rec_p:.4f} | F1: {f1_p:.4f} | AUC:
            plot_confusion_matrix(cm_p, classes, num_users, save_dir="artifacts", suffix="_patient")
        except Exception as e:
            print(f"[WARN] Patient-level evaluation failed: {e}")

    except FileNotFoundError:
        print(f"[WARN] Checkpoint not found: {checkpoint_path}")
    except Exception as e:
        print(f"[ERROR] Final evaluation failed: {e}")

try:
    plot_single_distribution_357(
        snapshots=dict_users_snapshots,
        labels_all=labels_all,
        classes=classes,
        save_path="artifacts/distribuicao_3_5_7_clientes.png"
    )
except Exception as e:
    print(f"[WARN] Falha ao gerar o gráfico único de distribuição: {e}")
```

Apêndice B

File - validation_stats

```
/usr/local/bin/python3.10 /Users/luanr/pycharm/FedAlzheimer/FedAlzheimer/validation_stats.py
   Estrutura validada: chaves consistentes entre fases

================================================================================
 RELATÓRIO DE VALIDAÇÃO DE MÉTRICAS - FEDERATED LEARNING
================================================================================

   1. VALIDAÇÃO INTERNA - FASE 1
--------------------------------------------------------------------------------

⚠️  ALTA VARIÂNCIA DETECTADA:
  • F1: CV médio = 0.1232, 6 configs afetadas

   VALORES SUSPEITOS:
  • F1 = 0.77% em DenseNet169+FedAvg+32+3c
  • F1 = 0.75% em DenseNet169+FedAvg+32+5c
  • F1 = 0.73% em DenseNet169+FedAvg+32+7c
  • F1 = 0.82% em DenseNet169+FedAvg+64+3c
  • F1 = 0.77% em DenseNet169+FedAvg+64+5c
  • F1 = 0.76% em DenseNet169+FedAvg+64+7c
  • F1 = 0.77% em ResNet50+FedAvg+32+3c
  • F1 = 0.72% em ResNet50+FedAvg+32+5c
  • F1 = 0.07% em ResNet50+FedAvg+32+7c
  • F1 = 0.71% em ResNet50+FedAvg+64+3c
  • F1 = 0.69% em ResNet50+FedAvg+64+5c
  • F1 = 0.55% em ResNet50+FedAvg+64+7c
  • F1 = 0.79% em DenseNet169+FedProx+32+3c
  • F1 = 0.73% em DenseNet169+FedProx+32+5c
  • F1 = 0.75% em DenseNet169+FedProx+32+7c
  • F1 = 0.76% em DenseNet169+FedProx+64+3c
  • F1 = 0.78% em DenseNet169+FedProx+64+5c
  • F1 = 0.72% em DenseNet169+FedProx+64+7c
  • F1 = 0.73% em ResNet50+FedProx+32+3c
  • F1 = 0.69% em ResNet50+FedProx+32+5c
  • F1 = 0.35% em ResNet50+FedProx+32+7c
```

Apêndice B

File - validation_stats

```
   • F1 = 0.76% em ResNet50+FedProx+64+3c
   • F1 = 0.69% em ResNet50+FedProx+64+5c
   • F1 = 0.47% em ResNet50+FedProx+64+7c
   • AUC = 0.95% em DenseNet169+FedAvg+32+3c
   • AUC = 0.96% em DenseNet169+FedAvg+32+5c
   • AUC = 0.95% em DenseNet169+FedAvg+32+7c
   • AUC = 0.97% em DenseNet169+FedAvg+64+3c
   • AUC = 0.92% em DenseNet169+FedAvg+64+5c
   • AUC = 0.93% em DenseNet169+FedAvg+64+7c
   • AUC = 0.94% em ResNet50+FedAvg+32+3c
   • AUC = 0.91% em ResNet50+FedAvg+32+5c
   • AUC = 0.50% em ResNet50+FedAvg+32+7c
   • AUC = 0.88% em ResNet50+FedAvg+64+3c
   • AUC = 0.89% em ResNet50+FedAvg+64+5c
   • AUC = 0.77% em ResNet50+FedAvg+64+7c
   • AUC = 0.95% em DenseNet169+FedProx+32+3c
   • AUC = 0.92% em DenseNet169+FedProx+32+5c
   • AUC = 0.94% em DenseNet169+FedProx+32+7c
   • AUC = 0.93% em DenseNet169+FedProx+64+3c
   • AUC = 0.95% em DenseNet169+FedProx+64+5c
   • AUC = 0.93% em DenseNet169+FedProx+64+7c
   • AUC = 0.90% em ResNet50+FedProx+32+3c
   • AUC = 0.88% em ResNet50+FedProx+32+5c
   • AUC = 0.63% em ResNet50+FedProx+32+7c
   • AUC = 0.89% em ResNet50+FedProx+64+3c
   • AUC = 0.90% em ResNet50+FedProx+64+5c
   • AUC = 0.67% em ResNet50+FedProx+64+7c

    Estatísticas de Coeficiente de Variação (Fase 1
  ):
    metric  cv_mean   cv_std   cv_min   cv_max
 Accuracy 0.096959 0.151261 0.000000 0.772143
Precision 0.130168 0.180714 0.000000 0.830647
    Recall 0.094877 0.111639 0.000000 0.448433
        F1 0.123157 0.183999 0.000000 0.874392
       AUC 0.064174 0.094956 0.002439 0.350172

================================================
  ============================
    2. VALIDAÇÃO INTERNA - FASE 2
----------------------------------------------
```

Apêndice B

File - validation_stats

```
 74 ----------------------------
 75    Variância aceitável em todas as métricas
 76
 77    VALORES SUSPEITOS:
 78   • F1 = 0.99% em DenseNet169+FedAvg+32+3c
 79   • F1 = 0.98% em DenseNet169+FedAvg+32+5c
 80   • F1 = 0.98% em DenseNet169+FedAvg+32+7c
 81   • F1 = 0.98% em DenseNet169+FedAvg+64+3c
 82   • F1 = 0.98% em DenseNet169+FedAvg+64+5c
 83   • F1 = 0.98% em DenseNet169+FedAvg+64+7c
 84   • F1 = 0.98% em ResNet50+FedAvg+32+3c
 85   • F1 = 0.98% em ResNet50+FedAvg+32+5c
 86   • F1 = 0.96% em ResNet50+FedAvg+32+7c
 87   • F1 = 0.98% em ResNet50+FedAvg+64+3c
 88   • F1 = 0.96% em ResNet50+FedAvg+64+5c
 89   • F1 = 0.93% em ResNet50+FedAvg+64+7c
 90   • F1 = 0.99% em DenseNet169+FedProx+32+3c
 91   • F1 = 0.98% em DenseNet169+FedProx+32+5c
 92   • F1 = 0.97% em DenseNet169+FedProx+32+7c
 93   • F1 = 0.99% em DenseNet169+FedProx+64+3c
 94   • F1 = 0.98% em DenseNet169+FedProx+64+5c
 95   • F1 = 0.96% em DenseNet169+FedProx+64+7c
 96   • F1 = 0.99% em ResNet50+FedProx+32+3c
 97   • F1 = 0.98% em ResNet50+FedProx+32+5c
 98   • F1 = 0.98% em ResNet50+FedProx+32+7c
 99   • F1 = 0.96% em ResNet50+FedProx+64+3c
100   • F1 = 0.88% em ResNet50+FedProx+64+5c
101   • F1 = 0.88% em ResNet50+FedProx+64+7c
102   • AUC = 1.00% em DenseNet169+FedAvg+32+3c
103   • AUC = 1.00% em DenseNet169+FedAvg+32+5c
104   • AUC = 1.00% em DenseNet169+FedAvg+32+7c
105   • AUC = 0.99% em DenseNet169+FedAvg+64+3c
106   • AUC = 1.00% em DenseNet169+FedAvg+64+5c
107   • AUC = 1.00% em DenseNet169+FedAvg+64+7c
108   • AUC = 1.00% em ResNet50+FedAvg+32+3c
109   • AUC = 0.99% em ResNet50+FedAvg+32+5c
110   • AUC = 1.00% em ResNet50+FedAvg+32+7c
111   • AUC = 1.00% em ResNet50+FedAvg+64+3c
112   • AUC = 0.99% em ResNet50+FedAvg+64+5c
113   • AUC = 0.99% em ResNet50+FedAvg+64+7c
114   • AUC = 0.99% em DenseNet169+FedProx+32+3c
```

Apêndice B

File - validation_stats

```
115   • AUC = 1.00% em DenseNet169+FedProx+32+5c
116   • AUC = 0.99% em DenseNet169+FedProx+32+7c
117   • AUC = 1.00% em DenseNet169+FedProx+64+3c
118   • AUC = 1.00% em DenseNet169+FedProx+64+5c
119   • AUC = 0.99% em DenseNet169+FedProx+64+7c
120   • AUC = 1.00% em ResNet50+FedProx+32+3c
121   • AUC = 1.00% em ResNet50+FedProx+32+5c
122   • AUC = 1.00% em ResNet50+FedProx+32+7c
123   • AUC = 0.99% em ResNet50+FedProx+64+3c
124   • AUC = 0.97% em ResNet50+FedProx+64+5c
125   • AUC = 0.97% em ResNet50+FedProx+64+7c
126
127    Estatísticas de Coeficiente de Variação (Fase 2
    ):
128    metric  cv_mean   cv_std   cv_min   cv_max
129  Accuracy 0.014637 0.012564 0.007253 0.071597
130 Precision 0.016226 0.015168 0.004098 0.083447
131    Recall 0.018552 0.008714 0.010012 0.055345
132        F1 0.017686 0.014116 0.008267 0.081891
133       AUC 0.006269 0.006263 0.000600 0.030782
134
135 ================================================
    ==============================
136    3. COMPARAÇÃO FASE 1 vs FASE 2
137 ------------------------------------------------
    ------------------------------
138
139    Resumo Estatístico:
140    metric  mean_phase1  mean_phase2
    pct_improvement  cohens_d effect_size          t_p
     significant_fdr
141  Accuracy     0.751767     0.970100        29.
    042699  1.341046       large 1.054775e-06
                  ***
142 Precision     0.695375     0.968479        39.
    274372  1.947956       large 1.834073e-09
                  ***
143    Recall     0.708333     0.966608        36.
    462353  2.257409       large 1.107458e-10
                  ***
144        F1     0.681121     0.967354        42.
```

Apêndice B

File - validation_stats

```
144 023870  1.792745       large 8.339526e-09
                ***
145       AUC     0.878512     0.994246        13.
    173783  0.998628       large 6.094058e-05
                ***
146
147    Intervalos de Confiança 95%:
148    metric  mean_diff  ci_95_lower  ci_95_upper
149  Accuracy   0.218333     0.153196     0.283470
150 Precision   0.273104     0.217012     0.329196
151    Recall   0.258275     0.212501     0.304049
152        F1   0.286233     0.222355     0.350111
153       AUC   0.115733     0.069367     0.162100
154
155 ================================================
    ==============================
156    4. COMPARAÇÃO ENTRE ALGORITMOS
157 ------------------------------------------------
    ------------------------------
158
159    Fase 1 - ANOVA entre algoritmos:
160    metric   f_stat  anova_p significant
161  Accuracy 0.026069 0.873207          No
162 Precision 0.015948 0.900652          No
163    Recall 0.001803 0.966516          No
164        F1 0.015050 0.903476          No
165       AUC 0.011810 0.914446          No
166
167    Fase 2 - ANOVA entre algoritmos:
168    metric   f_stat  anova_p significant
169  Accuracy 0.950842 0.340106          No
170 Precision 1.006566 0.326635          No
171    Recall 1.095472 0.306626          No
172        F1 1.061548 0.314057          No
173       AUC 2.331646 0.141018          No
174
175 ================================================
    ==============================
176    5. VERIFICAÇÃO DE CONSISTÊNCIA COM TABELAS DA
    DISSERTAÇÃO
177 ------------------------------------------------
```

Apêndice B

File - validation_stats

```
177 ------------------------------
178
179 ================================================
    ==============================
180    6. DETECÇÃO DE OUTLIERS
181 ------------------------------------------------
    ------------------------------
182
183 ⚠️  Outliers detectados na Fase 1:
184   Total: 20 configurações
185  Método   Modelo  Batch  Clientes      metric
    z_score
186  FedAvg ResNet50     32         7  Accuracy μ 3.
    998052
187  FedAvg ResNet50     64         7  Accuracy μ 0.
    396677
188 FedProx ResNet50     32         7  Accuracy μ 2.
    083575
189 FedProx ResNet50     64         7  Accuracy μ 0.
    523654
190  FedAvg ResNet50     32         7 Precision μ 3.
    222727
191  FedAvg ResNet50     64         7 Precision μ 0.
    998927
192 FedProx ResNet50     32         7 Precision μ 2.
    342944
193 FedProx ResNet50     64         7 Precision μ 1.
    603735
194  FedAvg ResNet50     32         7    Recall μ 3.
    251577
195  FedAvg ResNet50     64         7    Recall μ 1.
    040755
196 FedProx ResNet50     32         7    Recall μ 2.
    031419
197 FedProx ResNet50     64         7    Recall μ 1.
    788857
198  FedAvg ResNet50     32         7        F1 μ 3.
    702473
199  FedAvg ResNet50     64         7        F1 μ 0.
    797443
200 FedProx ResNet50     32         7        F1 μ 2.
```

Apêndice B

File - validation_stats

```
200 018371
201 FedProx ResNet50     64          7        F1 μ 1.
    280702
202  FedAvg ResNet50     32          7       AUC μ 3.
    288620
203  FedAvg ResNet50     64          7       AUC μ 0.
    901404
204 FedProx ResNet50     32          7       AUC μ 2.
    176790
205 FedProx ResNet50     64          7       AUC μ 1.
    821839
206
207 ⚠️  Outliers detectados na Fase 2:
208   Total: 13 configurações
209  Método   Modelo  Batch  Clientes      metric
    z_score
210  FedAvg ResNet50     64          7  Accuracy μ 1.
    268154
211 FedProx ResNet50     64          5  Accuracy μ 2.
    911500
212 FedProx ResNet50     64          7  Accuracy μ 3.
    046570
213 FedProx ResNet50     64          5 Precision μ 3.
    025471
214 FedProx ResNet50     64          7 Precision μ 2.
    947421
215  FedAvg ResNet50     64          7    Recall μ 1.
    547518
216 FedProx ResNet50     64          5    Recall μ 2.
    557863
217 FedProx ResNet50     64          7    Recall μ 3.
    180475
218  FedAvg ResNet50     64          7        F1 μ 1.
    237146
219 FedProx ResNet50     64          5        F1 μ 2.
    938241
220 FedProx ResNet50     64          7        F1 μ 2.
    980592
221 FedProx ResNet50     64          5       AUC μ 2.
    972664
222 FedProx ResNet50     64          7       AUC μ 3.
```

Apêndice B

File - validation_stats

```
222 033189
223
224 ================================================
    ==============================
225  FIM DO RELATÓRIO
226 ================================================
    ==============================
227
228    Resultados salvos em:
229   • validation_phase_comparison.csv
230   • validation_algo_phase1.csv
231   • validation_algo_phase2.csv
232   • validation_table_consistency.csv
233
234 Process finished with exit code 0
235
```