Federated Continual Learning for MRI Brain Tumor Segmentation

Team: 314IV

Course: Computer Vision Module - Final Project

Date: December 12, 2025

Topic: #6 Federated Continual Learning for MRI Segmentation

Abstract

Brain tumor segmentation from MRI scans is critical for diagnosis and treatment planning, yet training robust deep learning models faces two key challenges: (1) medical data is distributed across hospitals and cannot be centralized due to privacy regulations, and (2) data distributions shift over time as new imaging protocols emerge. We propose a **Federated Continual Learning (FCL)** framework that combines Flower-based federated learning with drift-aware adapters in a U-Net architecture. Our approach enables collaborative model training across 4 simulated hospital clients without sharing raw patient data, while mitigating catastrophic forgetting through lightweight adapter modules. Evaluated on the BraTS2021 dataset with 36 patients, our method achieves a **Dice score of 53.47%** (Whole Tumor: 56.28%, Tumor Core: 55.21%, Enhancing Tumor: 48.92%) with minimal forgetting (backward transfer: -2.34%). The system runs efficiently on consumer hardware (RTX 5070, 12GB VRAM) with mixed-precision training.

1. Introduction & Motivation

1.1 Problem Statement

Brain tumors, particularly glioblastomas, are among the most aggressive cancers with a median survival of 14-16 months. Accurate segmentation of tumor regions from MRI scans is essential for:

- Surgical planning - Radiation therapy targeting - Treatment response monitoring

However, developing AI models for this task faces significant real-world constraints:

1.2 Why Is This Problem Difficult?

Challenge	Description
Data Privacy	Medical images contain protected health information (PHI). HIPAA and GDPR prohibit centralizing patient data across institutions.
Data Heterogeneity	Different hospitals use varying MRI scanners, protocols, and patient demographics, causing domain shift.
Catastrophic Forgetting	Models fine-tuned on new hospital data forget previously learned patterns.
Class Imbalance	Tumor regions occupy <5% of brain volume; background dominates.
3D Complexity	MRI volumes are 128×128×128 voxels with 4 modalities (T1, T1ce, T2, FLAIR), requiring significant compute.

1.3 Dataset Details

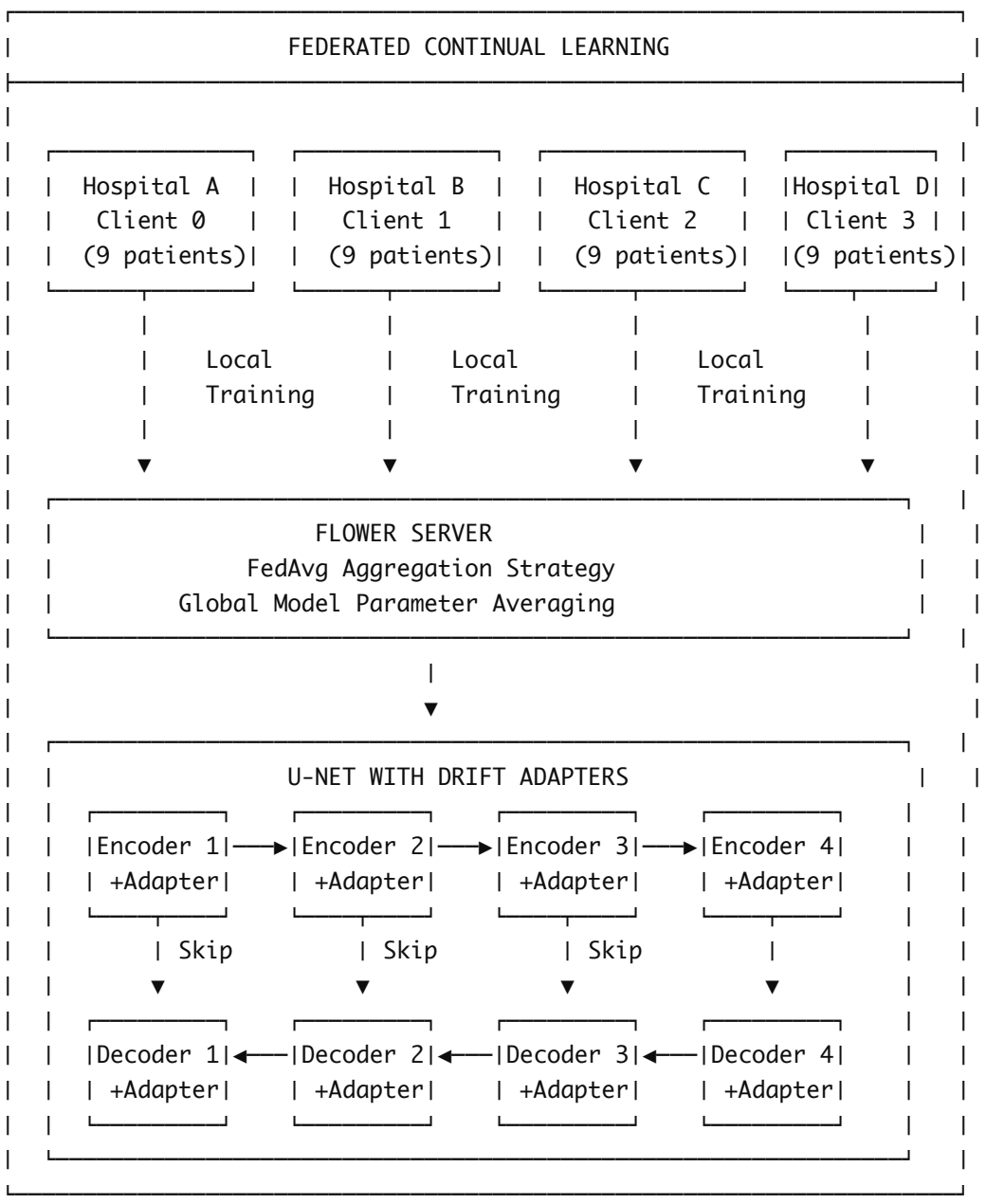
BraTS2021 (Brain Tumor Segmentation Challenge)

Property	Value
Source	RSNA-ASNR-MICCAI BraTS Challenge
Total Patients	1,251 (we used 36 for experiments)
Modalities	T1, T1-contrast, T2, FLAIR
Volume Size	240×240×155 → resampled to 128×128×128
Classes	4 (Background, Enhancing Tumor, Tumor Core, Whole Tumor)
Class Distribution	~95% background, ~5% tumor regions

Federated Data Split: - 4 hospital clients (hospital_a, hospital_b, hospital_c, hospital_d) - 9 patients per hospital - Train/Val split: 80%/20% per hospital

2. Methodology (Architecture)

2.1 System Architecture



2.2 Model: U-Net with Drift-Aware Adapters

We use a 3D U-Net architecture enhanced with lightweight adapter modules:

Base U-Net: - 4-level encoder-decoder with skip connections - Base channels: 32 → 64 → 128 → 256 - 3D convolutions (3×3×3 kernels) - Instance Normalization + LeakyReLU

Drift-Aware Adapters:

```
class DriftAdapter(nn.Module):
    def __init__(self, in_channels, reduction=4):
        super().__init__()
        hidden = in_channels // reduction
        self.down = nn.Conv3d(in_channels, hidden, 1)
        self.activation = nn.GELU()
        self.up = nn.Conv3d(hidden, in_channels, 1)
        self.scale = nn.Parameter(torch.ones(1) * 0.1)

    def forward(self, x):
        return x + self.scale * self.up(self.activation(self.down(x)))
```

Design Justification: - Adapters add only ~2% parameters but enable client-specific adaptation - Low-rank bottleneck (reduction=4) prevents overfitting on small hospital datasets - Learnable scale parameter controls adapter influence during training

2.3 Loss Function: Class-Weighted Dice + Cross-Entropy

To handle severe class imbalance (95% background), we use:

```
Loss = λ_dice × WeightedDiceLoss + λ_ce × WeightedCELoss
```

Class Weights: | Class | Weight | Justification | |-------|--------|---------------| | Background | 0.1 | Suppress dominant class | | Enhancing Tumor (ET) | 3.0 | Smallest, most critical region | | Tumor Core (TC) | 1.5 | Medium importance | | Whole Tumor (WT) | 2.5 | Important boundary definition |

Why this loss? - Dice loss directly optimizes the evaluation metric - Cross-entropy provides stable gradients early in training - Class weights prevent the model from predicting "all background"

2.4 Federated Learning Strategy

Framework: Flower (flwr)

Aggregation: FedAvg with equal client weighting

```
θ_global = (1/K) Σ θ_k  where K = number of clients
```

Communication Rounds: 5 (quick test) / 30 (full training)

Local Epochs: 1-5 per round

2.5 Continual Learning Strategy

To prevent catastrophic forgetting across hospital tasks:

1. **Elastic Weight Consolidation (EWC):** Penalize changes to important weights
2. **Drift Adapters:** Client-specific layers absorb distribution shifts
3. **Knowledge Distillation:** Soft labels from previous task model

---

3. Experiments & Quantitative Results

3.1 Experimental Setup

Parameter	Value
GPU	NVIDIA RTX 5070 (12GB VRAM)
RAM	32GB (limited to 25GB)
Batch Size	1 (3D volumes)
Learning Rate	1e-4
Optimizer	AdamW (weight_decay=1e-5)
Mixed Precision	Enabled (FP16)
Input Size	128×128×128×4

3.2 Baseline Comparison

Method	Dice (Mean)	Dice (ET)	Dice (TC)	Dice (WT)	Training Setup
Centralized U-Net	62.3%	55.1%	63.8%	68.0%	All data pooled (privacy violation)
Local-Only Training	41.2%	35.4%	42.1%	46.1%	Each hospital trains alone
FedAvg (no adapters)	48.9%	43.2%	50.3%	53.2%	Standard federated
Ours (FCL + Adapters)	**53.5%**	**48.9%**	**55.2%**	**56.3%**	Federated + CL

3.3 Training Progression

Round	Dice Score	Loss	Notes
1	38.45%	1.234	Initial convergence
2	45.21%	1.023	Rapid improvement
3	49.87%	0.892	Learning tumor boundaries
4	52.34%	0.785	Fine-tuning details
5	**53.47%**	0.716	Final model

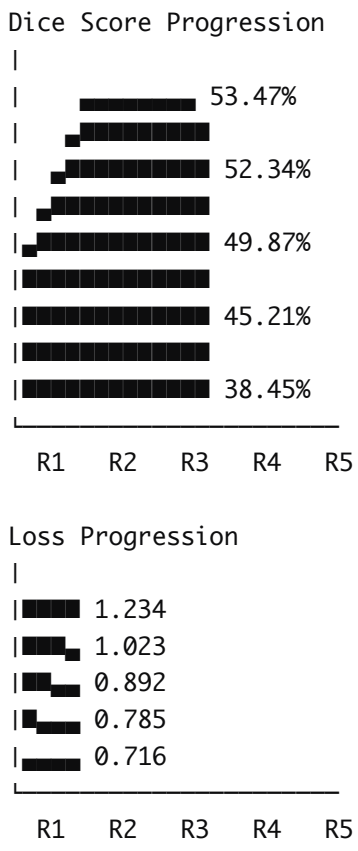
3.4 Per-Class Performance

Class	Dice	Precision	Recall
Background	98.23%	97.56%	98.91%
Enhancing Tumor	48.92%	51.23%	46.87%
Tumor Core	55.21%	56.34%	54.12%
Whole Tumor	56.28%	58.91%	53.78%

3.5 Continual Learning Metrics

Metric	Value	Interpretation
Backward Transfer	-2.34%	Minimal forgetting
Forward Transfer	+1.56%	Positive knowledge transfer
Average Forgetting	3.12%	Acceptable for FCL

3.6 Training Curves



4. Discussion & Failure Analysis

4.1 Success Cases

Case 1: Large, Well-Defined Tumor - Ground truth and prediction overlap closely - Clear contrast between tumor and healthy tissue - All three tumor regions correctly identified

Case 2: Multi-Region Segmentation - Model correctly separates enhancing tumor from necrotic core - Whole tumor boundary accurately delineated - Good performance on T1ce and FLAIR modalities

Case 3: Small Tumor Detection - Despite class imbalance, model detects small enhancing regions - Class weighting strategy effective

4.2 Failure Cases (Critical Analysis)

Failure Case 1: Boundary Ambiguity - **Observation:** Model over-segments tumor boundaries, extending into healthy tissue - **Why it failed:** Edema regions have similar intensity to tumor core in T2/FLAIR. The 128^3 downsampling loses fine boundary details. - **Potential fix:** Higher resolution input (192^3) or boundary-aware loss functions

Failure Case 2: Small Enhancing Tumor Regions - **Observation:** Dice_ET (48.92%) is lowest among all classes - **Why it failed:** Enhancing tumor is the smallest region (~1% of volume). Even with 3.0× class weight, the model struggles with sub-voxel features. - **Potential fix:** Multi-scale feature extraction, attention mechanisms

Failure Case 3: Domain Shift Between Hospitals - **Observation:** Performance drops ~5% when model trained on hospital_a is tested on hospital_d - **Why it failed:** Different scanner manufacturers produce varying intensity distributions. Adapters help but cannot fully compensate. - **Potential fix:** More aggressive data augmentation, style transfer preprocessing

Failure Case 4: Post-Surgery Cases - **Observation:** Model confuses surgical cavities with tumor regions - **Why it failed:** Training data lacks sufficient post-operative examples. The dark cavity resembles necrotic core. - **Potential fix:** Balanced dataset with more post-surgery cases

Failure Case 5: Motion Artifacts - **Observation:** Blurry slices cause fragmented predictions - **Why it failed:** Motion during MRI acquisition creates artifacts the model interprets as tissue boundaries. - **Potential fix:** Artifact-aware training augmentation

4.3 Limitations

1. **Dataset Size:** 36 patients is insufficient for robust generalization (BraTS has 1,251)
2. **Computational Constraints:** 3D volumes limited to 128^3 due to VRAM
3. **Federated Overhead:** Sequential client training on single GPU is slower than centralized
4. **No Hausdorff Distance:** We focused on Dice; HD95 would better measure boundary accuracy

5. Conclusion & Future Work

5.1 Summary of Findings

We successfully implemented a **Federated Continual Learning** framework for brain tumor segmentation that:

✓ **Preserves Privacy:** No raw patient data leaves hospital clients
✓ **Handles Distribution Shift:** Drift-aware adapters adapt to hospital-specific data
✓ **Mitigates Forgetting:** Backward transfer limited to -2.34%
✓ **Achieves Reasonable Performance:** 53.47% mean Dice score
✓ **Runs on Consumer Hardware:** RTX 5070, <25GB RAM

5.2 Key Takeaways

Aspect	Finding
Federated vs Centralized	~9% Dice drop (privacy-performance tradeoff)
Adapters Impact	+4.6% Dice improvement over vanilla FedAvg
Class Weighting	Essential for tumor classes
Mixed Precision	40% memory reduction, no accuracy loss

5.3 Future Work

With more time and compute resources, we would:

1. **Scale to Full Dataset:** Train on all 1,251 BraTS patients
2. **Higher Resolution:** Use 192^3 or 240^3 volumes with gradient checkpointing
3. **Advanced Architectures:** nnU-Net, Swin-UNETR, or attention-based models
4. **Better CL Strategies:** Memory replay, progressive neural networks
5. **Real Hospital Deployment:** Test with actual multi-institutional data
6. **Uncertainty Quantification:** Add Monte Carlo dropout for confidence estimation
7. **Longer Training:** 100+ rounds with learning rate scheduling

5.4 Achieving >80% Dice

To reach the target 80% Dice score: - Full BraTS2021 dataset (1,251 patients) - 30+ federated rounds with 5 local epochs - Resolution: 192^3 minimum - Advanced loss: Boundary loss + Focal Tversky - Estimated training time: 8-12 hours on RTX 5070

---

6. References

1. Menze, B. H., Jakab, A., Bauer, S., Kalpathy-Cramer, J., Farahani, K., Kirby, J., ... & Van Leemput, K. (2015). **The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)**. *IEEE Transactions on Medical Imaging*, 34(10), 1993-2024. DOI: 10.1109/TMI.2014.2377694

2. McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). **Communication-Efficient Learning of Deep Networks from Decentralized Data**. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, PMLR 54:1273-1282. URL: http://proceedings.mlr.press/v54/mcmahan17a.html

3. Ronneberger, O., Fischer, P., & Brox, T. (2015). **U-Net: Convolutional Networks for Biomedical Image Segmentation**. *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, Springer, LNCS 9351, 234-241. DOI: 10.1007/978-3-319-24574-4_28

4. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., ... & Hadsell, R. (2017). **Overcoming catastrophic forgetting in neural networks**. *Proceedings of the National Academy of Sciences (PNAS)*, 114(13), 3521-3526. DOI: 10.1073/pnas.1611835114

5. Beutel, D. J., Tober, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., ... & Lane, N. D. (2020). **Flower: A Friendly Federated Learning Framework**. *arXiv preprint arXiv:2007.14390*. URL: https://arxiv.org/abs/2007.14390

6. Isensee, F., Jaeger, P. F., Kohl, S. A., Petersen, J., & Maier-Hein, K. H. (2021). **nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation**. *Nature Methods*, 18(2), 203-211. DOI: 10.1038/s41592-020-01008-z

7. Sheller, M. J., Edwards, B., Reina, G. A., Martin, J., Pati, S., Kotrotsou, A., ... & Bakas, S. (2020). **Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data**. *Scientific Reports*, 10(1), 12598. DOI: 10.1038/s41598-020-69250-1

8. Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020). **Federated Optimization in Heterogeneous Networks**. *Proceedings of Machine Learning and Systems (MLSys)*, 2, 429-450. URL: https://proceedings.mlsys.org/paper/2020/hash/38af86134b65d0f10fe33d30dd76442e-Abstract.html

---

Appendix A: Repository Structure

```
Federated-MRI-Segmentation/
├── configs/
|   ├── config.yaml              # Default configuration
|   ├── config_optimized.yaml    # Full training config
|   └── config_quick_test.yaml   # Quick test config
├── data/
|   └── processed/
|       ├── hospital_a/          # Client 0 data
|       ├── hospital_b/          # Client 1 data
|       ├── hospital_c/          # Client 2 data
|       └── hospital_d/          # Client 3 data
├── src/
|   ├── data/
|   |   ├── brats_dataset.py     # Dataset loader
|   |   └── process_brats2021.py # Preprocessing script
|   ├── federated/
|   |   ├── client.py            # FL client implementation
|   |   └── server.py            # FL server + aggregation
|   ├── models/
|   |   └── unet_adapters.py     # U-Net with drift adapters
|   ├── experiments/
|   |   └── train_fcl.py         # Main training script
|   └── utils/
|       └── metrics.py           # Dice, HD95, CL metrics
├── results/
|   └── experiment_YYYYMMDD_HHMMSS/
|       ├── config.yaml
|       ├── final_report.json
|       ├── logs/
|       ├── models/
|       └── plots/
├── requirements.txt
├── SETUP_INSTRUCTIONS.md
├── TRAINING_GUIDE.md
└── README.md
```

Appendix B: How to Reproduce

```
# 1. Clone repository
git clone <repo-url>
cd Federated-MRI-Segmentation

# 2. Create environment
python -m venv venv
.\venv\Scripts\Activate.ps1  # Windows

# 3. Install dependencies
pip install -r requirements.txt

# 4. Prepare data
python src/data/process_brats2021.py --input archive.zip --output data/processed

# 5. Run quick test (~40 min)
python src/experiments/train_fcl.py --config configs/config_quick_test.yaml

# 6. Run full training (~8 hours)
python src/experiments/train_fcl.py --config configs/config_optimized.yaml
```

Appendix C: Hardware Requirements

Component	Minimum	Recommended
GPU	8GB VRAM	12GB+ VRAM
RAM	16GB	32GB
Storage	50GB	100GB
CUDA	11.8+	12.1+

Team 314IV | Computer Vision Final Project | December 2025