# University of Padova


Department of Mathematics

*Master Thesis in Big data Management & Analytics*


# Federated learning: A study of its applications in seizure prediction & colorectal polyp segmentation tasks


*Supervisor*

Professor Alessandro Sperduti

University of Padova

*Co-supervisor*

Professor Alberto Testolin

University of Padova

*Master Candidate*

Andrés Gabriel Espinal Hernández


*Academic Year*

2023-2024

In deep appreciation for all the mentors I have encountered during my journey; directly or indirectly materialized as family, friends, professors, or colleagues. Thanks for blessing me with the gift of knowledge and the motivation to pursue it.

# Abstract

Machine learning (ML) algorithms have achieved exceptional levels of performance in automating tasks across several domains. However, in a traditional ML or Centralized Learning (CL) approach, the data needs to first be collected in a centralized location before an ML model can be built, raising a series of data privacy-related issues in domains that deal with sensitive data, like finance, government, supply chains, and healthcare. The siloed nature of these data sources stifles the power of ML and has recently sparked the interest of the research community in techniques like Federated Learning (FL), which enables parties to participate in a collaborative effort to build a robust model without sharing their private data. In this work, FL is studied from a healthcare perspective to investigate its applicability in two domains: *seizure prediction* using EEG signals, and *polyp segmentation* using colonoscopy images. A systematic analysis of FL algorithms, model architectures, datasets, configurations, and hyperparameters is presented in this work. Results suggest that FL can produce equivalent, and even better results than CL in the aforementioned tasks, even for poorly performing centralized models. Hyperparameter tuning is highlighted as a key component of good FL models, and additional insights on the effect of federated hyperparameters are presented in an effort to motivate future research in hyperparameter optimization.

# Contents


Abstract **v**

List of figures **ix**

List of tables **xv**

Listing of acronyms **xvii**

1 Introduction **1**
- 1.1 Background & motivation . . . 1
- 1.2 Research questions . . . 2
- 1.3 Research objectives . . . 3
- 1.4 Justification & impact . . . 4

2 Background work & literature review 7
- 2.1 Federated learning . . . 7
  - 2.1.1 Taxonomy . . . 9
  - 2.1.2 Federated averaging strategies for model updates . . . 12
  - 2.1.3 FL systems review . . . 17
  - 2.1.4 Applications in the medical domain . . . 21
  - 2.1.5 Limitations . . . 22
  - 2.1.6 Future work & research directions . . . 23
- 2.2 Machine learning techniques . . . 24
- 2.3 Medical case studies & FL . . . 25
  - 2.3.1 EEG seizure detection/prediction . . . 25
  - 2.3.2 Polyp segmentation . . . 26

3 Methodology & experiment design **31**
- 3.1 Datasets & data preparation . . . 31
  - 3.1.1 EEG signals dataset . . . 31
  - 3.1.2 Polyp segmentation dataset . . . 33
  - 3.1.3 Dataset partitioning summary . . . 35
- 3.2 Hardware & software characteristics . . . 36
- 3.3 Hyperparameter tuning . . . 36
  - 3.3.1 Centralized setting . . . 37

3.3.2 Federated setting . . . 37
3.4 Model architectures . . . 39
3.5 Evaluation metrics . . . 40
3.6 Federated setup & experiment design . . . 42

4 Experimental results & discussion **43**
4.1 Experimental results . . . 43
4.1.1 Exploratory data analysis (EDA) . . . 44
4.1.2 Hyperparameter tuning . . . 47
4.1.3 Centralized model evaluation . . . 49
4.1.4 Federated model evaluation . . . 51
4.1.5 Effect of FL client size . . . 53
4.1.6 Federated model convergence . . . 56
4.1.7 Execution time . . . 58
4.1.8 Resources consumption . . . 59
4.2 Analysis & interpretation . . . 61

5 Conclusions & recommendations **65**

Appendix A Supplementary information **67**
A.1 Hyperparameter grids . . . 67
A.2 Hyperparameter tuning results . . . 69
A.3 Federated results . . . 75
A.4 Software settings . . . 78
A.5 EEG Extracted features . . . 79

References **81**

Acknowledgments **85**

# Listing of figures


2.1 General formulation of FL as defined by Rieke et al. [1]. . . . . 7
2.2 Example of the network architecture and communication process for an FL healthcare application by Nguyen et al. [2]. . . . . 8
2.3 Federated Averaging (FedAvg) algorithm as proposed by McMahan et al. [3] . 13
2.4 Adaptive Federated Algorithms: FedAdam, FedAdagrad & FedYogi as proposed by Reddi et al. [4] . . . . . 14
2.5 FedProx algorithm as proposed by Li et al. [5] . . . . . 16
2.6 Choice of input data across EEG studies reviewed by Craik et al. [6]. Panel A indicates the breakdown of input formulations across all EEG studies, the inner circle groups the outer options into categories. Panel B summarizes the choice of input category per EEG task, it is worth noting seizure detection studies use signal values predominantly. . . . . 27
2.7 ML technique implemented across EEG studies reviewed by Craik et al. [6]. Seizure detection studies focus mostly on RNNs and CNNs . . . . . 28
2.8 Schematic of a semantic segmentation task of colorectal polyps (Peralta et al., 2021) [7] . . . . . 30

3.1 Scalp positioning for the 20 common EEG channels used by Shafiezadeh et al. [8]. . . . . 32
3.2 Segmentation for the EEG recordings of epileptic patients studied by Shafiezadeh et al. [8]. Panel A displays 480 minutes of recording from the F7 channel and its division into the *interictal*, *preictal*, *ictal*, and *postictal* stages. Highlighted in green are the zones of the recording used for the binary classification task. Panels B through E show 20 s magnifications of the recordings from the 20 channels at the beginning of each stage. . . . . 33
3.3 Polyp dataset example, the upper row displays the source images and the bottom row shows the ground truth binary masks used to identify the ROI of the polyps. . . . . 34
3.4 Architecture for the FNN used for the seizure prediction task. . . . . 39
3.5 CaraNet architecture from Lou et al. [9]. . . . . 40
3.6 Evaluation metrics for the seizure prediction task (FNN). . . . . 41
3.7 Evaluation metrics for the polyp segmentation task (CNN). . . . . 41

4.1 Class distribution for the EEG dataset. Label 0 (blue) indicates a period of regular brain activity (no seizure) and label 1 (orange) indicates an atypical surge in brain activity (seizure incoming). . . . . 44
4.2 Per patient class distribution for the EEG dataset. Label 0 (blue) indicates a period of regular brain activity (no seizure) and label 1 (orange) indicates an atypical surge in brain activity (seizure incoming). . . . . 45
4.3 Class distribution for each of the polyp datasets across centralized partitioning strategy. Label 0 (Blue) is assigned to black pixels that do not contain the ROI, and 1 (orange) represents white pixels that contain the ROI containing the polyp. . . . . 45
4.4 Distribution of class labels in the EEG dataset per FL configuration (2, 4, 8, and 16 clients), and FL client for the stratified sampling and patient-aware datasets. The blue portion of the bar indicates the proportion of the negative labels, and the orange portion the proportion of positive labels. . . . . 46
4.5 Distribution of class labels in the polyp dataset per FL configuration (2, 4, 6, and 8 clients) and FL client. The blue portion of the bar indicates the proportion of the negative labels (black pixels in the segmentation mask), and the orange portion is the proportion of positive labels (white pixels in the segmentation mask). . . . . 47
4.6 Training results for the centralized seizure prediction task (FNN). Training loss (blue) against validation loss (orange) is plotted against the two dataset sampling approaches. . . . . 50
4.7 Model results for the centralized seizure prediction task (FNN). Test metrics are displayed accompanied by a snapshot of the training and validation metrics in the training epoch that minimized validation loss. . . . . 50
4.8 Training results for the polyp segmentation task (CNN). Mean Dice is plotted for training (blue) and validation (orange) datasets. Dice values closer to 1 indicate a closer match of inferred and ground truth segmentation masks. . . 51
4.9 Model results for the polyp segmentation task (CNN). Test mean dice is displayed accompanied by a snapshot of the training and validation metrics in the training epoch that minimized validation loss. . . . . 51
4.10 Mean test AUROC results for the federated seizure prediction task (FNN) across FL client sizes. Each panel represents one of the federated strategies evaluated, and each panel displays the results for the stratified and patient-aware datasets. Mean AUROC across datasets and FL client configurations is reported per federated strategy panel (line). . . . . 52
4.11 Mean test dice results for the federated polyp segmentation task (CNN) across FL client sizes. Each bar represents one of the federated strategies evaluated. Mean dice across strategies and FL client configurations are shown as a line. . 52

4.12 Model results for the federated seizure prediction task (FNN) partitioned by FL client sizes. Each bar represents one of the federated strategies and client configurations evaluated. The mean AUROC metric across strategies is shown as a line. . . . . 53

4.13 Graph showing the balance between sensitivity (blue) and specificity (orange) test metrics for the federated seizure prediction task (FNN). The vertical panels represent the stratified and patient-aware datasets, and the horizontal panels the federated strategy. Inside each cell, the horizontal axis displays the amount of FL clients, and the vertical axis the sensitivity/specificity values. . . 54

4.14 Mean test dice results for the federated polyp segmentation task (CNN) partitioned by FL client sizes. Each bar represents one of the federated strategies and client configurations evaluated. Mean dice across strategies are shown as a line. Empty bars (FedYogi with 2 and 4 clients) indicate numerical instabilities in the models that resulted in NaNs. . . . . 55

4.15 Training loss convergence curves for all tasks. The first two panels display the seizure prediction tasks curves with the stratified and patient-aware datasets respectively, while the third panel displays the polyp segmentation task curves. Rows represent each FL strategy employed (with different colors) and training loss is plotted per FL round in each cell. Each cell has an independent scale based on the loss function values for each configuration. . . . . 57

4.16 Comparison of training loss convergence curves for all tasks. FedAvg (dark blue) is compared against the adaptive strategies: FedAdam (orange), FedAdagrad (red), and FedYogi (sky blue). The first two panels display the seizure prediction tasks curves with the stratified and patient-aware datasets respectively, while the third panel displays the polyp segmentation task curves. . . . . 58

4.17 Execution time (seconds) of each of the models per federated strategy and client configuration, for the seizure prediction task (FNN). . . . . 59

4.18 Execution time (seconds) of each of the models per federated strategy and client configuration, for the polyp segmentation task (CNN). . . . . 59

4.19 Centralized setting average VRAM Memory consumption (MB) per training epoch for the networks used. . . . . 60

4.20 Federated setting average VRAM Memory consumption (MB) per FL algorithm, client configuration, dataset, and training round for the networks used. 61

A.1 Hyperparameter grid results for **FedAdam** in the seizure prediction task (FNN). Evaluation loss over the last 125 rounds was averaged to acquire the values for each cell across a combination of $\tau$, $\eta$, and $\eta_l$ combinations. Each combination was explored against a *stratified* and a *patient-aware* data sampling strategy. Minimum evaluation loss across configurations is marked with an (x). 69

A.2 Hyperparameter grid results for **FedAdagrad** in the seizure prediction task (FNN). Evaluation loss over the last 125 rounds was averaged to acquire the values for each cell across a combination of $\tau$, $\eta$, and $\eta_l$ combinations. Each combination was explored against a *stratified* and a *patient-aware* data sampling strategy. Minimum evaluation loss across configurations is marked with an (x). . . . . 70

A.3 Hyperparameter grid results for **FedYogi** in the seizure prediction task (FNN). Evaluation loss over the last 125 rounds was averaged to acquire the values for each cell across a combination of $\tau$, $\eta$, and $\eta_l$ combinations. Each combination was explored against a *stratified* and a *patient-aware* data sampling strategy. Minimum evaluation loss across configurations is marked with an (x). 71

A.4 Hyperparameter grid results for **FedProx** in the seizure prediction task (FNN). Evaluation loss over the last 125 epochs was averaged to acquire the values for each cell $\mu$ choice. Each choice was explored against a *stratified* and a *patient-aware* data sampling strategy. Minimum evaluation loss across configurations is marked with an (x). . . . . 72

A.5 Hyperparameter grid results for **FedAdam** in the polyp segmentation task (CNN). Evaluation loss over the last 3 rounds was averaged to acquire the values for each cell across a combination of $\tau$, $\eta$, and $\eta_l$ combinations. Maximum evaluation mean dice coefficient across configurations is marked with an (x). . . . . 72

A.6 Hyperparameter grid results for **FedAdagrad** in the polyp segmentation task (CNN). Evaluation loss over the last 3 rounds was averaged to acquire the values for each cell across a combination of $\tau$, $\eta$, and $\eta_l$ combinations. Maximum evaluation mean dice coefficient across configurations is marked with an (x). . 73

A.7 Hyperparameter grid results for **FedYogi** in the polyp segmentation task (CNN). Evaluation loss over the last 3 rounds was averaged to acquire the values for each cell across a combination of $\tau$, $\eta$, and $\eta_l$ combinations. Maximum evaluation mean dice coefficient across configurations is marked with an (x). . . . 74

A.8 Hyperparameter grid results for **FedProx** in the polyp segmentation task (FNN). Evaluation loss over the last 3 rounds was averaged to acquire the values for each cell $\mu$ choice. Maximum evaluation mean dice coefficient across configurations is marked with an (x). . . . . 74

A.9 Results of the federated models on the polyp segmentation task (CNN). All combinations of FL strategies and client configurations are displayed in this table. Test metrics are accompanied by the training and validation metrics encountered in the FL round that maximized the validation mean dice metric (Round best model column). Dice values for models with instabilities are filled in with 0's (FedYogi with 2 and 4 clients) . . . . . 75

A.10 Results of the federated models on the seizure prediction task (FNN). All combinations of FL strategies, client configurations, and datasets are displayed in this table. Test metrics are accompanied by the training and validation metrics encountered in the FL round that minimized validation loss (Round best model column). . . . . 76
A.11 Training loss convergence curves for the federated seizure prediction task (FNN). The left panel contains results for the stratified sampling dataset, while the right panel contains results for the patient-aware dataset. Rows represent each FL strategy employed (with different colors), and columns are divided by the number of FL clients. Each cell represents a model trained with a specific FL strategy, FL client configuration, and dataset. Training loss is plotted per FL round in each cell, and the best round (the one that minimizes evaluation loss) is highlighted using a dotted line. Every cell has an independent scale based on the configuration loss values. . . . . 77
A.12 Training loss convergence curves for the federated polyp segmentation task (CNN). Rows represent each FL strategy employed (with different colors), and columns are divided by the number of FL clients. Each cell represents a model trained with a specific FL strategy, FL client configuration, and dataset. Training loss is plotted per FL round in each cell, and the best round (the one that minimizes evaluation loss) is highlighted using a dotted line. Every cell has an independent scale based on the configuration loss values. . . . . 78
A.13 Conda Environment: List of the main Python packages used and their versions. 78
A.14 List of the APIs (Python MNE package, version 3.8.5) used to extract the signal features from the EEG dataset by Shafiezadeh et al. [8]. . . . . 79

# Listing of tables

2.1 FL taxonomy considered for the medical cases studied in this work. . . . . 11
2.2 Popularity of FLS Projects. . . . . 20
2.3 Landscape of FLSs based on the taxonomy proposed by Li et al. [10]. . . . . 21

3.1 EEG seizure dataset partitioning strategy. . . . . 35
3.2 Polyp dataset partitioning strategy. . . . . 35
3.3 Hardware configuration of the GCP VM used to run experiments . . . . . 36

4.1 Best combination of federated hyperparameters found for the seizure prediction (FNN) and polyp segmentation (CNN) tasks across datasets and FL strategies. . . . . 49

# Listing of acronyms

**FL** . . . . . Federated Learning

**CL** . . . . . Centralized Learning

**FLS** . . . . . Federated Learning System

**ML** . . . . . Machine Learning

**DL** . . . . . Deep Learning

**FNN** . . . . . Feed-Forward Neural Network

**RNN** . . . . . Recurrent Neural Network

**CNN** . . . . . Convolutional Neural Network

**CaraNet** . . . . . Context Axial Reverse Attention Network

**CFP** . . . . . Channel-wise Feature Pyramid

**IoU** . . . . . Intersection over Union

**Res2Net** . . . . . Residual Resolution Network

**SGD** . . . . . Stochastic Gradient Descent

**P2P** . . . . . Peer-to-Peer

**SMPC** . . . . . Secure Multi-Party Computation

**DP** . . . . . Differential Privacy

**FedAvg** . . . . . Federated Averaging

**IID** . . . . . Independent and Identically Distributed

**EEG** . . . . . Electroencephalogram

**ROI** . . . . . Region of Interest

**GCP** . . . . . Google Cloud Platform

**VM** . . . . . Virtual Machine

**CV** . . . . . Cross-Validation

**ROC** . . . . . Receiver Operating Characteristic

**AUROC** . . . . . Area Under the ROC Curve

**BCE** . . . . . Binary Cross-Entropy

**CRC** . . . . . Colorectal Cancer

**EHR** . . . . . Electronic Health Record

# 1 Introduction

## 1.1 Background & motivation

Machine learning (ML) algorithms are computational models that have been designed to analyze data, identify patterns, and make predictions or decisions without explicit programming, by learning entirely from data. These types of algorithms have achieved exceptional levels of success in tasks like healthcare, natural language processing (NLP), image recognition, finance, and recommendation systems. Traditionally, these models require the data to be collected in a centralized location before training can start. However, bringing the data to a central location might pose regulatory, ethical, legal, and technical challenges related to privacy and protection of the data [1] in some domains. These challenges lead to the problem of *data islands*, which is a phenomenon commonly seen in areas such as finance, government, supply chains, and healthcare [10] where data is isolated and out of reach for ML development. This makes it difficult to build robust ML models, especially in domains like healthcare where publicly available data is scarce. An example of such scarcity is present in public colorectal polyp datasets, which are often used to detect colorectal cancer (CRC); a type of cancer with high mortality rates, and significant miss rates by clinicians [11]. Traditional ML approaches are unable to leverage this type of data, due to its siloed nature, which highlights the importance of data privacy-preserving techniques in ML.

Federated Learning (FL) is a technique where different clients or parties collaborate to learn

a model on a central server while keeping individual client data decentralized [12]. FL eliminates the issue of data islands by training an independent model on each client's local data, and then aggregating these models to produce a global model that learns of each of the independent data distributions. A typical FL process consists of the following steps, which are repeated for a set of training rounds: 1) A set of FL clients is selected amongst a pool of participants to train a local model, b) once training is done, the local model parameters are collected and aggregated in a coordinator server, c) the server uses the aggregated parameters to construct a global model, d) the global model parameters are then redistributed to the clients. These steps are repeated until achieving convergence, at the end of training a global model that can be used for inference in all the clients is produced. This process is often accompanied by privacy-preserving techniques to securely move model weights through the network. Recent developments in FL have enabled the development of complex machine-learned models in a distributed manner, particularly in the medical domain while preserving privacy and addressing security concerns [13]. FL introduces a wealth of new opportunities in healthcare by enabling large-scale research on rare diseases [1], that would be otherwise impossible in a centralized setting due to the aforementioned data privacy challenges.

## 1.2 Research questions

In this section, the main problems investigated in this work are presented. All of these inquiries have been formulated in the context of two medical tasks of critical importance to the field a) A seizure prediction task using a dataset composed of features extracted from time intervals of EEG signals and b) A colorectal polyp segmentation task using colonoscopy images and binary segmentation masks from a set of polyp datasets. These problems have been addressed through the application of Deep Learning (DL) techniques in a Federated Learning (FL) setting which is the subject of this study.

**Assumptions.** Under the assumptions that: a) simulated federated workloads output comparable model evaluation results to those executed in distributed resources; b) results obtained across evaluation metrics are independent of the presence of privacy mechanisms for model updates (differential privacy, cryptographic methods, ...), as the impact these methods have on model results is quantifiable; and c) the data partitioning strategies implemented in this study to distribute data across training devices lead to data distributions that are comparable to those found in real federated workloads. The purpose of this research is to answer the following question.

**Main Research Question**

Is Federated Learning (FL) a viable alternative, capable of achieving equivalent levels of evaluation and generalization performance to Centralized Learning (CL), when applied to medical tasks in the domain of seizure prediction and colorectal polyp segmentation?

To work towards the goal of answering the main research question, a series of specific research questions are listed below. All these questions will be constantly addressed throughout this manuscript.

**Specific Research Questions**

- **Q1.** How do federated approaches fare against their centralized counterparts in terms of *convergence, evaluation metrics, execution times, resource consumption, and generalization performance*?
- **Q2.** What is the effect that the number of FL clients has on these metrics?
- **Q3.** Are some FL averaging strategies more robust to non-IID datasets than others?
- **Q4.** Do adaptive FL averaging strategies improve the convergence performance of the models?
- **Q5.** What is the effect that the choice of hyperparameters has on the behavior of models across averaging strategies?

## 1.3 Research objectives

**General Objective.** Contribute to the advancement of research on Federated Learning (FL) by analyzing the performance of FL approaches on a set of medical tasks, across a series of DL architectures, FL strategies, client configurations, hyperparameter choices, and datasets.

**Specific Objectives**

- Provide a systematic overview of the current state-of-the-art techniques relevant to DL, FL, and the case of studies reviewed in this work.

- Design, implement and analyze the experiments required to answer each of the research questions.
- Guarantee that the results of said experiments are transparent and reproducible.
- Identify interesting research directions for FL on the aforementioned tasks.

## 1.4 Justification & impact

Recent research has demonstrated FL models can achieve comparable levels of performance to those trained in centrally hosted data sets, and superior to models that only see centralized data [1]. However, one of the limitations of FL techniques comes from their privacy-preserving nature: researchers are often unable to investigate data upon which models are being trained to acquire a deeper understanding of their behavior [1]. For this reason, there is a need to systematically analyze the behavior of federated models in relation to the data distributions they are trained on, and this is often only possible by performing FL simulations. To aggravate the problem, publicly available medical datasets are scarce which highlights even more the importance of FL. Systems trained in federated settings can yield less biased decisions and better generalization performance in rare cases due to their exposure to more complete data distributions [1], which is of special value in healthcare. The two medical cases studied in this work (seizure prediction and polyp detection) can benefit from the application of FL.

Epilepsy is a severe neurological disease that leads to recurrent seizures and affects around 65 million people worldwide [8]. This has motivated the development of techniques and devices to automatically anticipate seizures using EEG signals [6]. Across studies, it has been found that models applied to seizure detection/prediction using EEG data can achieve decent levels of performance. However, model evaluation for this task is often based on questionable randomized cross-validation schemes, which can introduce correlated signals [8] and exhibit pessimistic results when generalizing to unseen patient data. Finding correlated samples of EEG signals across hospitals in real scenarios is unlikely, as a result of each hospital holding an independent set of patients. Additionally, seizure data is often imbalanced, as seizure episodes happen less often than regular brain activity [8], and different hospitals might contain different distributions of data, which further complicates the development of ML models. For this reason, investigating seizure prediction from a federated perspective provides a more realistic view of the performance these types of algorithms can have when deployed. Furthermore, there

might be opportunities to improve the performance of centralized ML models through the use of FL techniques designed for dealing with heterogeneous datasets.

Colorectal polyps are a precursor for a particularly deadly type of cancer called Colorectal Cancer (CRC), which accounts for 10% of overall cancer cases worldwide [7]. Even though patient survival rates can improve with early detection of the polyps, clinicians miss between 14% and 30% of the polyps in colonoscopy images [11]. Although several model architectures have found success in automatic polyp detection and segmentation tasks, they are usually not robust to detecting small medical objects, which are very common in colorectal polyps [9]. For this reason, architectures like CaraNet [9] have been designed, which are more sensitive to the inherent size differences of polyps and provide better detection and segmentation results. However, the success of ML techniques heavily relies on the amount and quality of data used for training. Unfortunately, publicly annotated polyp datasets are scarce [11] and the vast majority of colonoscopy images are kept private in hospital data islands. For this reason, studying polyp segmentation from an FL perspective can help overcome data privacy issues and enable researchers to perform studies across a wealth of otherwise inaccessible data. As of the date of writing, no studies were found considering FL for the segmentation of small polyps using the CaraNet architecture, making this the first study of this kind.

# 2

# Background work & literature review

The intention of this chapter is to provide an overview of the literature and build a foundation to explain the experiments carried out. This chapter is divided into three sections: Section 2.1 reviews Federated Learning (FL), Section 2.2 reviews the considered machine learning architectures and techniques, and Section 2.3 reviews the state of the art on the medical cases studied.

## 2.1 Federated learning

Rieke et al. [1] define Federated Learning (FL) as a "learning paradigm that seeks to address the problem of data governance and privacy by training algorithms collaboratively without exchanging the data itself". A general formulation of FL is presented in Figure 2.1. $\mathcal{L}$ denotes the global loss function, which is obtained through a weighted combination of $K$ local losses $\{\mathcal{L}_k\}_{k=1}^{K}$ and computed for the private dataset $X_k$ which resides in each of the $K$ parties or *FL Clients* and never shared among them [1]. Therefore, the objective of FL is to find a set of parameters $\phi$ that minimizes a global loss function based on the collaborative effort of a set of independent FL clients.

$$\min_{\phi} \mathcal{L}(X;\phi) \quad \text{with} \quad \mathcal{L}(\mathrm{X};\phi) = \sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{w_k}\ \mathcal{L}_{\mathrm{k}}(\mathrm{X_k};\phi)$$

**Figure 2.1:** General formulation of FL as defined by Rieke et al. [1].

An example of a typical FL application for the medical domain inspired by this formulation is shown in Figure 2.2. An example of the architectural components (left panel) and communication process (right panel) for a medical application is displayed: a set of devices or *FL clients* (wearables, mobile devices, computers, etc.) is selected from an available pool of clients. Each of the selected clients uses their local datasets to train a model, these local model parameters are then uploaded to an *aggregation server* that collects each of the parameters and aggregates them to construct a global model. Finally, the global model is redistributed to each client as the new local model, and the process is repeated for a set of *FL rounds* until convergence is achieved. The result of this process is a global model at the end of training that aggregates the characteristics of the models of each of the FL clients, and can then be redistributed and used for inference. Note that none of the training devices communicate between themselves, and no data (other than the model parameters) is communicated through the network. This allows the FL network to produce a model that takes into account each individual client's data distribution while keeping their data private and secure. A similar client-server architecture is simulated throughout this study for the study of FL in medical tasks.

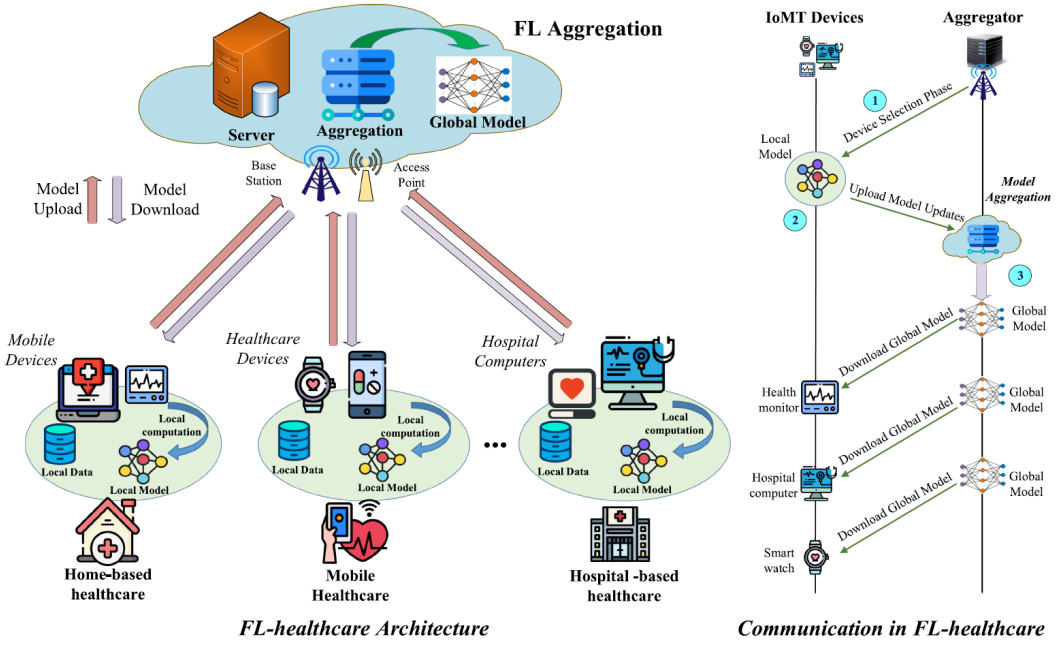


**Figure 2.2:** Example of the network architecture and communication process for an FL healthcare application by Nguyen et al. [2].

### 2.1.1 Taxonomy

FL solutions can be classified according to different criteria, as defined by Li et al. [10]. It is important to take into account the following criteria when designing FL solutions, as the choices done in the design phase depend on the domain and the task at hand. The specific FL taxonomy used for this study is summarized in Table 2.1.

- **Data Partitioning**: Indicates the strategy adopted to distribute the data across the parties involved in an FL setting. The choice of what partitioning strategy to use depends on the type of data and task at hand.
  - *Horizontal*: Also known as *feature-based partitioning*. The parties have the same feature space but little to no intersection in the sample space. In other words, from a data set perspective, each node holds the same columns, but different observations from the data. The advantage of this configuration is that it follows a security-first approach naturally while allowing learning independence across nodes [12] (each node can independently train its own model).
  - *Vertical*: Also known as *sample-based partitioning*. The parties have different feature spaces but intersecting sample spaces. In other words, from a data perspective, each node has different columns belonging to intersecting observations. This setting works well for data sets that are highly intersecting by nature [12], but also works well for data sets that don't intersect naturally. For the second scenario, *Entity Alignment* can be used to identify overlapping samples on related data sets by analyzing the feature space across different nodes, further enriching feature content across the FL network.
  - *Hybrid*: A combination of *horizontal* and *vertical* partitioning strategies. Under this scenario *Transfer Learning* can be leveraged to discover insights on data sets that are not directly related, using data sets with small intersecting spaces/from different domains.

- **Machine Learning Model**: The type of Machine Learning (ML) technique used to accomplish the task through an FL setting. The choice of ML technique defines the logistics of how the FL network operates. The most used ML models in FL settings are listed below.
  - *Neural Networks*: Currently the most used technique, sparked by several FL studies on Stochastic Gradient Descent (SGD) and the use of neural networks with multiple layers, or Deep learning (DL). Even though neural networks can achieve state-of-the-art levels of performance, they suffer from limited interpretability.

  - *Decision Trees*: Highly efficient to train and best suited for tasks that require interpretability.
  - *Linear Models*: Well-designed FL systems that deal with Linear/Logistic Regression and Support Vector Machines (SVM) exist.

- **Privacy Mechanism**: FL does not provide any security or privacy guarantees by itself but it sets the stage to implement mechanisms with this objective [12]. The privacy mechanisms are the measures put in place to securely move learned weights across the FL network. Several strategies have been proposed to securely update and distribute the weights of the FL network.

  - *Cryptographic Methods*: *Homomorphic Encryption* and *Secure Multi-Party Computation* allow parties to encrypt messages and perform operations on these encrypted messages before finally being decrypted securely to get the final result.
  - *Differential Privacy*: Injects random noise to the data or to the model parameters to provide statistical privacy guarantees for records and protection against *model poisoning*, which is a malicious attack that consists in influencing the global model weights to alter its behavior. The disadvantage of these methods is that they tend to have a detrimental impact on model performance.

- **Communication Architecture**: The communication strategy used by the parties involved in the FL network to update model weights.

  - *Centralized*: A manager node is in charge of aggregating the local weight updates from the models and sending back training results. This architecture has the advantage of being simpler but not the most resilient, as it involves a single point of failure.
  - *Decentralized*: Communications are performed across parties: each party is able to update the global weights of the model directly. Some examples of these communication architectures are Peer-to-Peer (P2P), Graph, and recently Blockchain networks.

- **Averaging Strategy**: To build a global model, a set of local models are averaged during training. The specific steps of how this averaging is done, and how the updates are communicated between parties are defined by the federated averaging strategy or FL algorithm. Several strategies exist to address different challenges in FL (heterogeneous data distributions, convergence, communication costs, etc.) and are explored more deeply in Section 2.1.2.

- **Scale of Federation**: The characteristics of the devices participating in a FL task, based on the amount of data they hold, computational power, and number of devices in the network.

  - *Cross-Silo*: The nodes are organizations or data centers. The amount of nodes under this scale tends to be small, but each node holds a large amount of data and has the computational power to carry out expensive computations. Under this configuration, most clients participate in each federated round and nodes are capable of maintaining state between rounds [4].
  - *Cross-Device*: The nodes are usually mobile devices with limited computational resources. The number of nodes under this scale tends to be large, but the nodes usually hold small amounts of data and compute lighter workloads. On cross-device configurations only a small fraction of the clients might participate in each federated round and clients are not allowed to maintain state [4]. *Edge Computing* is an example of cross-device FL.

- **Motivation of Federation**: Motive for the parties to actively participate in the FL task.

  - *Regulation*: The parties are involved in an FL setting due to requirements, dependencies, or other binding reasons. Example of this is FL done inside a company or organization.
  - *Incentive*: The parties participate in a FL setting on an opt-in basis, in exchange for some incentive. Examples of this are hospitals that join an FL network to benefit from robust models that have been trained with examples from other hospitals.

| | |
|---:|---|
| **Data Partitioning** | Horizontal |
| **Machine Learning Model** | Neural Networks |
| **Privacy Mechanism** | Not Considered |
| **Communication Architecture** | Centralized |
| **Averaging Strategies** | FedAvg, FedAdam, FedAdagrad, FedYogi, FedProx |
| **Scale of Federation** | Cross-Silo |
| **Motivation of Federation** | Regulation |

**Table 2.1:** FL taxonomy considered for the medical cases studied in this work.

### 2.1.2 Federated averaging strategies for model updates

In FL, a set of FL clients with independent models and datasets, collaboratively train a global model that inherits the underlying characteristics of these datasets, while remaining agnostic to each other's data distributions. In order to do this, an *averaging strategy* is required to aggregate each of the independent model updates into one single global model. Several solutions have been proposed to solve this problem, as it is explained in this section.

One of the first algorithms to be developed, and currently the state-of-the-art method and most used strategy for FL [5] is *the Federated Averaging* or *FedAvg* strategy introduced by McMahan et al. [3]. This federated strategy is used for gradient descent/ascent-based learning approaches. Pseudocode for this algorithm can be found in Figure 2.3.

- **FedAvg**: Clients perform multiple rounds of training, updating their weights with their local datasets, usually with (but not limited to) SGD as a local optimizer. Each client takes one (or more) local steps of gradient descent on its model and data. Then, the server fetches each of the client weights and performs a weighted average to acquire a new set of weights to update the global model. This set of weights represents what has been learned across all FL client's data distributions. The aggregated weights are then redistributed to each of the participating clients and a new *round* of FL is started, repeating the same process for a given amount of federated rounds.

  The amount of work the whole FL network does is defined by 3 quantities. Namely, $C$ which represents the fraction of clients that perform computation on each round; $E$ the number of training passes or *epochs* each client makes over its local dataset on each FL round, and $B$, which represents the local mini-batch size used for the client updates. These parameters need to be calibrated according to the nature of the problem and data at hand, the neural network architecture, and the hardware characteristics of the client/server. For instance, not all FL clients are required to participate in a given FL round (participating nodes are usually sampled randomly from a pool of available clients), and adding too many clients might be detrimental to the global model performance [3]. Increasing the amount of local SGD updates each FL node does, can result in smaller communication costs and faster convergence, but it can also cause clients to *overwork* and diverge from the global solution. Finally, clients might have different hardware characteristics (for example different VRAM) and $B$ needs to be calibrated accordingly.

  Despite being one of the most widely used FL algorithms FedAvg currently suffers from two problems. The first one involves *client drift* which causes clients' local models to diverge from the optimal global solution. This can occur especially in federated settings with heterogeneous data distributions across nodes, as local models might overemphasize client-specific features, leading to overfitted client solutions that don't generalize

**Algorithm 1** `FederatedAveraging`. The $K$ clients are indexed by $k$; $B$ is the local minibatch size, $E$ is the number of local epochs, and $\eta$ is the learning rate.

**Server executes:**
  initialize $w_0$
  **for** each round $t = 1, 2, \dots$ **do**
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of $m$ clients)
    **for** each client $k \in S_t$ **in parallel do**
      $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

**ClientUpdate**$(k, w)$: *// Run on client $k$*
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)
  **for** each local epoch $i$ from 1 to $E$ **do**
    **for** batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$
  return $w$ to server

**Figure 2.3:** Federated Averaging (FedAvg) algorithm as proposed by McMahan et al. [3]

well when averaged. The second problem is its *lack of adaptivity* [4] which might lead to convergence issues in some scenarios as a consequence of using fixed learning rates.

Reddi et al. [4] proposed a series of *adaptive federated strategies* to deal with the adaptivity issues inherent to FedAvg with a focus on the *cross-silo* setting where only a fraction of the FL clients might participate per round. These models can be seen as extensions of the original FedAvg algorithm proposed by McMahan et al. [3]. Their work focuses on adaptive server optimization, a technique that aims to reduce the communication costs that result from saving the state of optimizers across clients.

Adaptive learning rates can be highly beneficial in some DL tasks to improve the convergence of the models. Just as in the centralized setting, an adaptive learning rate can aid federated models to navigate over the error landscape more efficiently by dynamically decreasing/increasing the step size of gradient descent according to the gradients' magnitude. One such example can be found when training *language models* as they usually exhibit noisy stochastic gradient distributions [4] which might cause the model to miss local minima, or run into vanishing/exploding gradients issues.

All these federated algorithms are inspired by their centralized counterparts (*Adam*, *Adagrad*, and *Yogi* optimizers). Due to their reinterpretation in the federated domain, they require specifying a few additional hyperparameters. Parameter $\tau$ represents the *degree of adaptivity* of the algorithm, with smaller values of $\tau$ indicating higher levels of adaptivity. Additionally, the server learning rate $\eta$ and client learning rate $\eta_l$ need to be specified. The choice of these hyperparameters depends on the underlying data characteristics of the task at hand. In their ex-

perimental results, Reddi et al. [4] found that parameters $\eta$ and $\eta_l$ are inversely related for non-adaptive optimizers like FedAvg, which means both parameters need to be tuned together. On the other hand, when inspecting the hyperparameter grids visually (plotting accuracy against $\eta$ and $\eta_l$), they found that parameters $\eta$ and $\eta_l$ delimitate rectangular shapes in hyperparameter space where the accuracy of the models is optimized. This suggests that the choice of $\eta$ might be more important to tune than $\eta_l$, as there are several choices of $\eta_l$ that generate good results once a good choice of $\eta$ is picked. Pseudocode for the described adaptive algorithms can be found in Figure 2.4.

- **FedAdam**: Similar to the centralized Adam optimizer it combines an adaptive learning rate (like the Adagrad optimizer) and momentum-based methods (such as those in the RMSProp optimizer). The optimizer's goal is to adapt the learning rate of the model based on the average of past gradients and squared gradients to more efficiently traverse the loss function landscape. Parameters $\beta_1$ and $\beta_2$ represent the exponential decay rate of the first and second moment estimates (mean and variance of the gradients, respectively).
- **FedAdagrad**: This is another adaptive optimizer whose purpose is to adjust the learning rate based on the history of the squared gradients. Unlike Adam, it doesn't include momentum-based updates, though.
- **FedYogi**: Similar to Adam implements both an adaptive learning rate and momentum.

**Algorithm 2** FEDADAGRAD, FEDYOGI, and FEDADAM

Initialization: $x_0$, $v_{-1} \geq \tau^2$, decay parameters $\beta_1, \beta_2 \in [0, 1)$
**for** $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  **for** each client $i \in \mathcal{S}$ **in parallel do**
    **for** $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$
    $\Delta_i^t = x_{i,K}^t - x_t$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $m_t = \beta_1 m_{t-1} + (1 - \beta_1)\Delta_t$
  $v_t = v_{t-1} + \Delta_t^2$ (FEDADAGRAD)
  $v_t = v_{t-1} - (1 - \beta_2)\Delta_t^2 \operatorname{sign}(v_{t-1} - \Delta_t^2)$ (FEDYOGI)
  $v_t = \beta_2 v_{t-1} + (1 - \beta_2)\Delta_t^2$ (FEDADAM)
  $x_{t+1} = x_t + \eta \frac{m_t}{\sqrt{v_t} + \tau}$

**Figure 2.4:** Adaptive Federated Algorithms: FedAdam, FedAdagrad & FedYogi as proposed by Reddi et al. [4]

One frequent problem that arises from federating learning across nodes is the existence of heterogeneous data. Naturally, federated clients might belong to different entities with very heterogeneous sources of data. Providing a rather naive example, a particular hospital might have a higher incidence of a specific type of cancer like *Mesothelioma*, which is a rare and aggressive form of cancer that develops mostly in the lungs as a result of asbestos exposure. If this hospital happens to be located in a community where asbestos exposure is common, it is logical to think the incidences of this type of cancer might be over-represented when comparing it to other hospitals participating in the same FL project. This is defined by Li et al. [5] as *statistical heterogeneity*. This imposes a challenge on model averaging as clients might overfit to these over-represented samples, compromising the global model generalization performance. Heterogeneous data distributions pose challenges for FL algorithms and strategies, as many are assuming independently and identically distributed (IID) data across the participants [1].

Another frequent problem, especially on *cross-device* FL settings is that FL clients may differ in the characteristics of their systems. Consequently, it is logical to think that some FL clients might require more local rounds of gradient descent than others due to the nature of its data before sending the updates to the server for averaging. This hints at the necessity of varying the amount of work each local client does (namely the number of gradient descent steps) across time. This is defined by Li et al. [5] as *system heterogeneity*

An averaging strategy that attempts to solve both *statistical* and *systems* heterogeneity has been proposed by Li et al. [5]. Similar to the adaptive algorithms mentioned before, FedProx is an algorithm that extends the functionality of FedAvg. To see the code for this algorithm refer to Figure 2.5.

- **FedProx**: This is an algorithm that attempts to solve issues caused by the divergence of local models from the global model due to heterogeneous datasets. FedProx implements this through a *regularization term* that is added to the loss function of each of the federated clients. The intuition behind this strategy is to add a penalization term to the local loss functions before minimizing them in a way that local models are penalized when they start diverging from global solutions. The penalty term is defined as the L1 Norm of the difference between the local model weights $w$ and the global model weights $w^t$ scaled by a parameter $\mu$ known as the *proximal term*, that limits the impact of variable local updates. Besides ensuring local updates don't diverge too much from the best global solutions, it safely allows each FL client to perform a variable amount of work based on the heterogeneity degree of its data [5]:

$$\frac{\mu}{2}\|w - w^t\|^2.$$

It is worth noting that setting the value of $\mu = 0$ yields a vanilla FedAvg strategy, as the penalty term becomes zero and each gradient descent step minimizes only the original loss function. The choice of $\mu$ is critical for the performance of the FedProx algorithm. Heuristically, a large choice of $\mu$ can slow down convergence as updates are forced to be closer to the initial global weights. As training progresses, the weights acquired at each federated round will be very similar to the previous one, and "recycled" for the next round as a result of the proximal term. On the other hand, small values of $\mu$ might have a negligible regularization impact on the loss function [5].

The main federated averaging strategies used in this work have been reviewed. *FedAvg* is the current state-of-the-art algorithm for averaging model updates and requires calibration of the learning rate and the local epochs hyperparameters. Even though it provides satisfactory results in several scenarios it suffers from problems like *client drift* and *lack of adaptivity*. To solve the adaptivity problems, adaptive federated strategies have been proposed based on their centralized adaptive optimizer counterparts. *FedAdam*, *FedAdagrad*, and *FedYogi* are all adaptive algorithms that dynamically adapt learning rates using different degrees of momentum and require calibration of the degree of adaptivity $\tau$ and server and client learning rates $\eta$, $\eta_l$ parameters. Finally, to deal with *statistical* and *systems heterogeneity* the *FedProx* algorithm has been proposed, which introduces a regularization term in the local client loss functions to prevent local updates from straying too far from the solution of the global model.

**Algorithm 2** `FedProx` (Proposed Framework)

**Input:** $K, T, \mu, \gamma, w^0, N, p_k, k = 1, \cdots, N$
**for** $t = 0, \cdots, T-1$ **do**
  Server selects a subset $S_t$ of $K$ devices at random (each device $k$ is chosen with probability $p_k$)
  Server sends $w^t$ to all chosen devices
  Each chosen device $k \in S_t$ finds a $w_k^{t+1}$ which is a $\gamma_k^t$-inexact minimizer of: $w_k^{t+1} \approx \arg\min_w \ h_k(w;\ w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2$
  Each device $k \in S_t$ sends $w_k^{t+1}$ back to the server
  Server aggregates the $w$'s as $w^{t+1} = \frac{1}{K}\sum_{k \in S_t} w_k^{t+1}$
**end for**

**Figure 2.5:** FedProx algorithm as proposed by Li et al. [5]

### 2.1.3 FL systems review

As a natural consequence of the application of federated techniques over and over again in studies across different domains, some patterns started to become evident with time. Eventually, researchers realized that there are *"common methods and building blocks for building FL algorithms"* [10], and as in other fields in computer science, frameworks started to emerge to ease the development of FL tasks and avoid boilerplate code. These FL frameworks, more formally known as Federated Learning Systems (FLSs) [10], are usually presented in the form of libraries and APIs that facilitate a faster development cycle of FL solutions. In this section, the main offerings of FLS/frameworks are reviewed as of the time of writing. For a compact overview of all the FLSs capabilities refer to Table 2.3.

First, let's review Li et al. [10] definition of an FLS:

> In a federated learning system, multiple parties collaboratively train machine learning models without exchanging their raw data. The output of the system is a machine-learning model for each party (which can be the same or different). A practical federated learning system has the following constraint: given an evaluation metric such as test accuracy, the performance of the model learned by federated learning should be better than the model learned by local training with the same model architecture.

It is worth to note the terms FLS and FL Framework are used interchangeably across this research. Given that there are several independent FLS projects out there, currently gaining traction, and reviewing them all is beyond the scope of this manuscript, the list of FL Frameworks presented is not meant to be exhaustive. This comprehensive list of FLSs was built following leads on FL surveys like the ones from Li et al. [10], querying for *Federated Learning* projects in GitHub and searches against Google Scholar & DBLP:

- **FATE**: *(Federated AI Technology Enabler)* [14] is a Python-based industrial-grade FL open-source framework maintained by the Linux Foundation. It implements secure computation protocols based on homomorphic encryption and Secure Multi-Party Computation (SMPC), and works on horizontally and vertically partitioned datasets. Amongst the ML algorithms supported by this framework are: logistic regression, tree-based algorithms, deep learning, and transfer learning algorithms. It also supports deployment on single or multiple machines. Even though it has good documentation, it might not be the best choice for researchers as implementing new federated algorithms requires modifying FATE's source code [10].

- **Tensorflow Federated (TFF)**: This is an open-source framework developed by Google aimed to build machine learning models on TensorFlow, a widely used deep learning framework. TFF's interfaces are organized in two layers, namely the Federated Learning (FL) API, which enables the application of included federated implementations to already existing TensorFlow models, and the Federated Core (FC) API, which is a set of low-level interfaces that allow developers expressing new federated algorithms [15, 10]. Using TFF is a reasonable choice for models that have been already coded in TensorFlow due to its tight integration with the framework.

- **PySyft**: This is a Python framework proposed by OpenMined, an open-source AI community, and organization. It can be set up to work with two of the most used Deep Learning Frameworks: TensorFlow & PyTorch. PySyft experiments can run against a single machine (through simulation of FL clients) or on multiple machines [10]. Additionally to its Federated capabilities, it can run Federated data science workloads through a numpy-like interface that enables a permission-based workflow on remotely owned data. PySyft also provides differential privacy, and encrypted computation as a means to secure model weights [16]. Even though it is a framework with a lot of activity in Github [10] and flexibility due to its framework support, it has recently become a challenging framework to work with. PySyft's APIs change substantially during each release and the documentation as of the time of reviewing is outdated, which leads to a steep learning curve.

- **PaddleFL**: Is a federated framework based on the *PaddlePaddle* [10] DL platform that works both on Python and C++. The framework provides functionality for both horizontal and vertical federated workloads and secure aggregation protocols. It provides scalable deployment of workloads across multiple computing nodes, which makes it suitable for large-scale FL applications. PaddleFL architecture is modular which also makes it a good framework for research.

- **FedML**: FedML [17] is an open research library and benchmark designed to facilitate FL research. It supports several FL computing paradigms like on-device training, distributed computing, and single-machine simulation. It provides standardized benchmarks alongside synthetic and real-world datasets to facilitate fair performance comparison. Its architecture is made up of two components, the FedML-API and FedML-core. FedML-API provides a high-level API for users to build distributed training applications, focusing on algorithmic implementations and abstracting low-level communication details. FedML-core separates distributed communication and model training into two modules, supporting various network topologies and security/privacy functions. Overall, FedML aims to overcome the limitations of existing FL frameworks by providing a comprehensive and flexible toolkit for FL algorithm development and fair performance comparison under diverse computing paradigms and configurations.

- **Flower**: Flower [18], is a recent language-agnostic federated framework that provides high-level abstractions aimed to make experimentation and research on FL easy. It works very well with frameworks that use NumPy-like interfaces and also provides interfaces for frameworks that don't. Despite lacking functionality to preserve the privacy of model updates, Flower provides functionality to extend FL implementations to mobile and wireless clients with heterogeneous resources. Also, unlike other frameworks, Flower focuses on simulations with a large number of federated clients through their gRPC communication protocol and Virtual Client Engine (VCE) [18]. Overall, Flower is a good choice for researchers and individuals that require to migrate already existing models without being constrained by their deep learning framework choice. Flower's documentation is well maintained, and as a consequence federating existing centralized models is relatively painless. However, due to how recent the framework is (the original paper by Beutel et al. is from 2022 [18]) there are still some bugs and unexpected behaviors that need to be polished.

- **Nvidia Flare**: Nvidia's *Federated Learning Application Runtime Environment* [19], is a recently open-sourced software development kit (SDK) that offers state-of-the-art FL algorithms and federated ML approaches. Additionally, it offers privacy-preserving systems for multiparty collaboration using techniques like homomorphic encryption and differential privacy. NVFlare is a framework that is both adept in research simulations and in real-world production settings. Even though originally designed with healthcare applications in mind, NVFlare is not limited to healthcare and can be applied across various domains. Developers can bring their own data science workflows implemented in popular libraries like PyTorch, TensorFlow, or even pure NumPy, and apply them in a federated setting. NVFlare has been used in real use case scenarios for breast mammography classification, prostate segmentation, pancreas segmentation, chest X-ray (CXR), and electronic health record (EHR) analysis to predict oxygen requirements for COVID-19 patients.

- **OpenFL**: [20] is an open-source project maintained by Intel Labs and the University of Pennsylvania that works with TensorFlow, PyTorch, and other ML and DL frameworks. It is mostly coded in Python, and FL plans containing the rules and configurations for federation are orchestrated through YAML files. What distinguishes this framework from others is that it implements privacy protection policies through mutually-authenticated TLS connections and supports trusted execution environments (TEEs) like Intel SGX (Which implements secure enclaves within the CPU to protect computations providing confidentiality and data integrity guarantees). OpenFL has been implemented in a Federated Tumor Segmentation Initiative with 56 clinical sites spread worldwide. The framework exhibits some weaknesses as the FL plan has to be agreed upon by all parties before the workload begins, which may not suit all FL scenarios. On the other hand, Intel SGX benefits require specialized hardware to be exploited and may

| Federated Framework | Github | Popularity (Github Stars) |
|---|---|---|
| PySyft | https://github.com/OpenMined/PySyft | 8.9k |
| FATE | https://github.com/FederatedAI/FATE | 5.1k |
| FedML | https://github.com/FedML-AI/FedML | 3.1k |
| Flower | https://github.com/adap/flower | 2.8k |
| Tensorflow Federated (TFF) | https://github.com/tensorflow/federated | 2.1k |
| OpenFL | https://github.com/securefederatedai/openfl | 561 |
| PaddleFL | https://github.com/PaddlePaddle/PaddleFL | 456 |
| IBM Federated Learning | https://github.com/IBM/federated-learning-lib | 427 |
| Nvidia Flare | https://github.com/NVIDIA/NVFlare | 405 |

**Table 2.2:** Popularity of FLS Projects.

also introduce additional complexity in production deployments.

- **IBM Federated Learning**: [21] is a Python library that provides modular components for networking, protocol handling, data handling, and FL training modules. Due to its modular design, IBM Federated Learning is able to provide a communication infrastructure independent of the FL algorithm and machine learning library used. Communication can be performed through Flask, gRPC, and WebSockets. The framework supports FL algorithms for training neural networks, decision trees, and machine learning models. Like other frameworks, it also supports securing model updates through SMPC and differential privacy. It provides FL Model implementations for Keras, PyTorch, TensorFlow, Scikit-learn, and RLlib, allowing ML professionals to leverage their preferred libraries when specifying and training models. Even though IBM Federated Learning is available on GitHub, it can only be used for experimental and non-commercial purposes.

As it can be seen in Table 2.2 PySyft is currently the most popular FLS with the most active developer community. Table 2.3 summarizes the capabilities of each of the FL frameworks. It is evident from this summary that only a few of them support vertical partitioning and all of them support neural networks and linear ML models (with a few supporting tree-based methods). Additionally, most of the frameworks support federating models originally built in TensorFlow and Pytorch, being PaddleFL the only exception as it was built exclusively for the Paddle Paddle framework. Furthermore, the privacy mechanism offerings vary across frameworks, being those more targeted for production pipelines the ones that support cryptographic methods and differential privacy. Finally, most of the frameworks provide functionality to run single-machine experiments (simulations) and multi-resource workloads, with only a few of them providing on-device functionality. Overall, there is a vast offering of FL frameworks developers can choose from to satisfy their model federation needs.

| Characteristics | | PySyft | FATE | FedML | Flower | TFF | OpenFL | PaddleFL | IBM FL | NVFlare |
|---|---|---|---|---|---|---|---|---|---|---|
| **License** | Apache License 2.0 | X | X | X | X | X | X | X | | X |
| | Non-Commercial Use | | | | | | | | X | |
| **Data Partitioning** | Horizontal | X | X | X | X | X | X | X | X | X |
| | Vertical | | X | X | | | | X | | X |
| **ML Models** | Neural Networks | X | X | X | X | X | X | X | X | X |
| | Decision Trees | | X | | X | | | | X | X |
| | Linear Models | X | X | X | X | X | X | X | X | X |
| **ML Frameworks** | TensorFlow | X | X | X | X | X | X | | X | X |
| | PyTorch | X | X | X | X | | X | | X | X |
| | Scikit-Learn | | X | | X | | X | | | X |
| | XGBoost | | X | | X | | | | | X |
| | Paddle Paddle | | | | | | | X | | |
| **Privacy Mechanisms** | CM | X | X | X | | | | X | X | X |
| | DP | X | | | | X | X | | X | X |
| **Computing Paradigms** | Simulation | X | X | X | X | X | | X | | X |
| | Distributed | X | X | X | X | X | X | X | X | X |
| | On-Device | | | X | | X | | | | |

**Table 2.3:** Landscape of FLSs based on the taxonomy proposed by Li et al. [10].

**Choice of Framework.** Flower was chosen as the FLS for this study due to its ease of use, focus on research, and comprehensive documentation and examples. The framework is well integrated with *PyTorch* (the ML framework used) and facilitates the migration of existing models to the federated setting. Furthermore, it has been used successfully to train Feed-Forward Neural Networks (FNNs) and Convolutional Neural Networks (CNNs) which are the techniques used in this study.

### 2.1.4 Applications in the medical domain

FL has contributed to the development of applications across an extensive and diverse collection of domains. Due to its capabilities to exploit otherwise siloed datasets and its evident benefits to data privacy, it has recently gained traction in healthcare applications [6]. In this section, a set of medical use cases where FL has been applied is presented, based on the systematic review from Nguyen et al. [2].

- **EHR Management**: Valuable digital medical information can be captured from Electronic Hospital Records (EHRs). However, this type of private data is highly sensitive and metadata removal is not enough to address privacy concerns. FL enables hospitals to leverage health records to train models that can train on this data. An example of these applications is the prediction of hospitalizations for patients diagnosed with heart

diseases, which can be achieved by using a combination of patients' smartphones and distributed hospital data. Other examples include forecasting patient mortality based on drug features, and the detection of preterm birth.

- **Remote Health Monitoring**: FL has been used to provide in-home health monitoring, a service that has accrued special interest in the medical domain. Personal devices (like mobile phones and wearables) at each home can learn personalized CNN models that when aggregated through FL can achieve accuracies as high as 95% to detect human activity, which would be otherwise impossible using only the individual clients' data. These types of solutions have several applications in fields like assisted living, fall detection, mood analysis, and monitoring treatment effects on patients.

- **Medical Imaging**: A significant proportion of clinical data is composed of medical images (X-rays, CT scans, MRI, PET, EEG, etc.), which presents an interesting opportunity for DL models to take advantage of this type of data across data centers. FL has been successfully implemented in a series of medical imaging tasks like small bowel disease detection, and segmentation tasks such as pancreas cancer, breast and lung cancer, brain tumors, and colorectal polyps among others.

- **COVID-19 Detection & Diagnosis**: The recent pandemic also attracted the attention of FL researchers in an effort to detect COVID-19 cases. Due to concerns with public data sharing across datacentres holding COVID-19 data analytics, strategies to train models while keeping each data center's data private were required. DL techniques such as CNNs have been widely used in the identification of cases by extracting essential features from chest X-rays and computed tomography (CT) scans. When federated, CNNs were able to achieve COVID-19 detection rates of up to 98% when leveraging DL architectures such as ResNet18.

### 2.1.5 Limitations

Despite the many advantages that FL provides, it does not solve all the issues inherent to working with medical data [1]. Some of the main challenges and limitations faced by FL approaches are presented by Banabilah et al. [12]:

- **Reliability of Edge Devices**: Real-time communication might cause batteries of devices to drain which has an effect on the reliability of an edge device to participate in FL. For this reason, FL frameworks provide the functionality to sample a pool of available training clients in each round. Furthermore, some algorithms like *FedProx* [4] include logic to calibrate the amount of work each FL client does.

- **Imbalanced Data**: Since each node might have imbalanced data that is not identically and independently distributed, it can cause model performance and generalization problems as local models overfit. Some federated algorithms like *FedProx* [4] aim to solve this issue by introducing a regularization term that guarantees local updates don't deviate too much from the best global solution.

- **Communication Costs**: Since each training round of FL a set of interactions needs to occur between the server and clients, to transmit local updates and distribute the weights from the global model to the local clients, network congestion can increase the overall communication costs.

- **Privacy Challenges**: FL models are vulnerable to *data Poising*, or *model Poisoning* which is a malicious threat that aims to influence the global model to misclassify specific inputs that might lead to negative effects on the FL clients and the global model. Furthermore, shared information may still indirectly expose the data distributions used for local training, through methods like model inversion and adversarial attacks [1].

- **Model Behavior & Investigations**: Naturally, as access to the local datasets is not possible in a true federated setting, researchers might not be able to investigate the data used to train the models to get a deeper understanding of the results acquired from the global models [1].

### 2.1.6 Future work & research directions

Federated Learning is a relatively new subject of study and as such there are still a wealth of open topics to be investigated. Rauniyar et al. [13] provide an overview of some of the future research directions of FL in the medical setting, which are summarized below.

- **Hyperparameter Optimization**: Tuning of hyperparameters is of vital importance to the performance of FL systems, but it becomes a process difficult to execute in a federated setting due to the lack of incentives for parties to collaborate. One open research area is of designing proper incentives that motivate FL parties to participate in hyperparameter optimization processes.

- **Security and Privacy-Enabled FL Systems**: Malicious attacks like model poisoning are highly undesirable in medical settings. Current strategies to alleviate this problem impose constraints on model performance or have to be applied in controlled settings which motivates the development of new techniques that can adapt to a federated setting.

- **Efficient Communication Paradigm for FL Systems**: This becomes increasingly important in FL applied to edge devices, as the communication overhead required to update the model parameters across the network can create bottlenecks. Efforts have been made to solve this by reducing the amount of FL rounds required (improve model convergence), optimizing communication bandwidths, and compression/sparsification of models.

- **Solving Medical Data Heterogeneity and Statistical Issues of FL Systems**: Non-IID data is the norm in federated learning, and it is well known this type of data degrades the performance of DL models. Even though there are currently some strategies (like *FedProx*) that are targeted towards heterogeneous FL more research needs to be done in this area.

- **Contrastive Learning as a Solution for Un-labelled Medical Data Distribution**: Most of the research done in FL focuses on supervised learning models in which the data is annotated with the expected ground truth labels. Since a significant amount of the medical data generated is unlabeled (for example, colonoscopy images) there is an opportunity to exploit techniques like *contrastive learning* to acquire insights on this type of data. Contrastive learning (otherwise known as self-learning) involves a pre-training procedure in which the model learns to discriminate between similar and distinct data samples from an unlabeled dataset.

- **Benchmarking FL**: Even though several studies have been done for FL across different domains, results tend to be tricky to compare due to differences in experiments and evaluation procedures. This calls for a need for standard methodologies and FL datasets to evaluate FL performance.

## 2.2 Machine learning techniques

In this section, a summarized overview of the DL techniques used in this investigation is provided.

**Feed-Forward Neural Networks (FNNs).** Are the most basic type of learning algorithm used as the basis for most DL methods. These networks are composed of a series of layers containing an interconnected set of units or "neurons" which take the inputs, process these inputs through an *activation function* (a linear combination of the inputs like ReLU or hyperbolic tangent), and produce a new set of outputs for the next layers. The network assigns a set of *weights and biases* to the units and connections between units, which are adjusted through training to

minimize a *loss function* [22]. In a feed-forward network, information flows in a unidirectional path, from the input layers to the output layers. Learning occurs through a process known as *backpropagation*, which consists in applying *gradient descent* to propagate errors across the network in the opposite direction. The intuition behind this technique is that the gradient of the loss function with respect to the weights and biases (the network parameters) can be used to navigate the error landscape in the direction that minimizes loss (the discrepancy between the predicted and ground truth values). As a consequence, through an iterative training process, the error landscape can be navigated through small steps, which will lead to better solutions with time. Once the model is trained it can be used for inference by feeding it unseen data. These types of networks are particularly useful in regression and classification tasks.

**Convolutional Neural Networks (CNNs).** Even though FNNs are useful for both regression and classification tasks they do not exploit spatial information/spatial autocorrelation from the inputs, which might be valuable for several medical tasks (for example identifying polyps in an image). Primarily designed for tasks related to image and video processing, CNNs are designed to efficiently extract features from images. The building blocks of CNNs are the *convolutional layers*, in which a set of filters or *kernels* (a small weights matrix) slides over a region of the image performing element-wise operations and outputting a *feature map* that highlights certain patterns (edges, shapes, textures, etc.) in the input [22]. Similar to FNNs, non-linearity is introduced through an activation function that allows the network to learn complex relationships in the data. *Pooling layers* are used to reduce the dimensionality of the feature maps by retaining the maximum values found in a local region and discarding the rest, reducing computational complexity while retaining important feature information.

## 2.3 Medical case studies & FL

This section reviews the two main medical tasks that were studied in this investigation, *seizure prediction from EEG signals* and *colorectal polyp segmentation from colonoscopy images*. A contextual description of each medical case, its importance, and a literature review of the most common ML and FL techniques applied in these fields are provided.

### 2.3.1 EEG seizure detection/prediction

EEG signals are widely used in several fields such as neural engineering, neuroscience, biomedical engineering, sleep analysis, and seizure detection [6]. The motivation to work with these

signals becomes evident when analyzing diseases like Epilepsy, which is a severe neurological disease that leads to recurrent seizures and affects around 65 million people worldwide [8]. There is evidence that alterations in brain dynamics can be observed before epileptic attacks, which has led to the development of devices to anticipate seizures by analyzing EEG signals [8]. This evidence has led researchers to explore the possibility of applying ML techniques to EEG signals, to both identify and anticipate the occurrence of seizures in patients. In their topical review for EEG classification tasks, Craik et al. [6] summarize the most common ML techniques, data cleansing approaches, and inputs used in seizure detection/prediction tasks.

Several input data formulations have been proposed across EEG studies in line with the type of task and ML technique employed. Specifically, 41% of the studies surveyed used *calculated features*, 39% of the studies used *images*, and the remaining 39% used *signal values* [6]. Specific choices of input data types for each of these categories can be found in Figure 2.6. Besides choosing the right input formulation, preprocessing also requires special attention as EEG signals tend to exhibit a low signal-to-noise ratio (due to electrodes picking unwanted electrical physiological signals, channel cross-talk, and motion artifacts) and high dimensionality [6]. EEG studies implement different strategies to deal with these preprocessing challenges. From all the studies, 41% of them didn't report performing any signal cleansing, while 29% performed manual removal, 8% did automatic removal, and 22% worked directly with the raw signal [6].

Concerning the model architecture choice, the state-of-the-art approaches to tackle seizure detection tasks across studies are split into Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), as shown in Figure 2.7. Seizure detection studies employ RNNs more than any other EEG analysis study, probably due to the added benefit RNNs have on exploiting temporal autocorrelation in the signals. CNNs remain the most used method to perform inference in seizure detection tasks. Average accuracies of up to 84% have been achieved by studies that use CNNs with images or calculated features as inputs, while CNNs using signal values as inputs achieved even higher accuracies of up to 87%, which leads to thinking data cleansing might not be as important when using EEG signals in neural networks as it is in other ML methods [6].

### 2.3.2 Polyp segmentation

*Colorectal polyps* are "protrusions or bumps that develop in the colon as the body produces an excessive number of unwanted cells in the lining of the bowel" [23]. These polyps can develop anywhere in the large intestine or rectum. Colorectal Cancer (CRC), is a type of cancer with

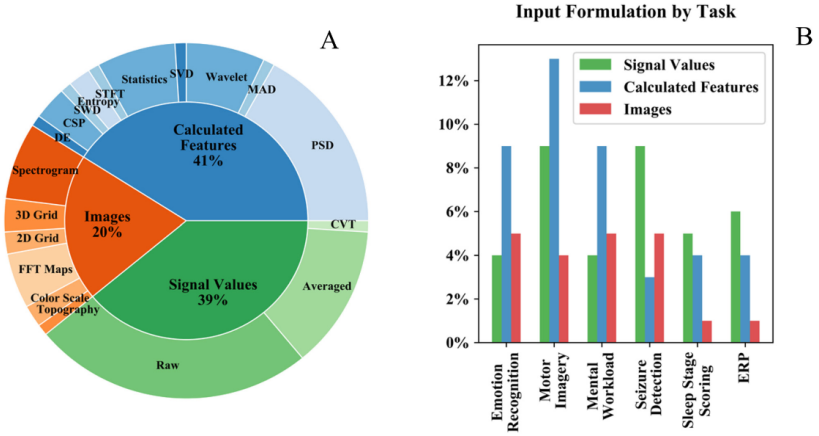


**Figure 2.6:** Choice of input data across EEG studies reviewed by Craik et al. [6]. Panel A indicates the breakdown of input formulations across all EEG studies, the inner circle groups the outer options into categories. Panel B summarizes the choice of input category per EEG task, it is worth noting seizure detection studies use signal values predominantly.

a high mortality rate that might happen if these polyps (which are usually harmless in their initial stages) are left untreated [23]. According to the world health organization's International Agency for Research on Cancer (IARC), approximately 2 million individuals were diagnosed with CRC in 2020, with an approximate 1 million people death toll [23], and CRC accounts for 10% of overall cancer cases worldwide [7]. Additionally, CRC incidence has been found to be higher in countries with a diet high in calories and animal fat and sedentary populations [7]. Colonoscopy is currently the standard procedure to detect polyps, which involves a multistep process consisting of preparing the bowel with a specific solution for a subsequent examination phase where a colonoscopy equipped with a camera and light is inserted into the patient's rectum for an obstruction-free view of the colon [23].

As it is evident, early colorectal polyp detection is of crucial importance to prevent the prevention of CRC. Unfortunately, polyp detection has a significant miss rate among physicians [23] which is usually between 14% to 30%, depending on the type and size of polyp [11]. The high miss rate is a concerning issue, as the standard method for detecting and diagnosing them is through colonoscopy [23]. This leads to the conclusion that automated detection systems can help gastroenterologists increase the detection rate of polyps [7]. As a matter of fact, DL solutions have been able to achieve expert levels of performance in several cases [7]. For this reason, automatic methods for the detection, classification, and segmentation of polyps through ML are needed, which has led to increased interest among the research community.

DL architectures used for polyp detection are usually specialized to excel in a specific type of

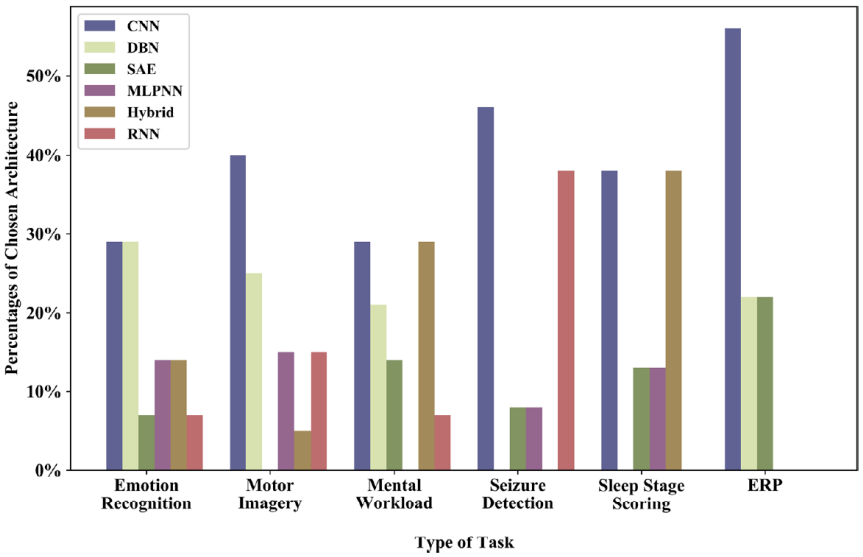


**Figure 2.7:** ML technique implemented across EEG studies reviewed by Craik et al. [6]. Seizure detection studies focus mostly on RNNs and CNNs

.

task. A synthesis of the most used DL approaches and their benefits are presented by Peralta et al. [7], who group these approaches into the following categories:

- **Feature Extractor**: FL architectures are used to automatically create feature vectors which can then be used as input to other classifiers, resulting in hybrid methods that combine deep learning with classical classifiers.
- **Classification**: A classifier labels an image as containing a polyp or not, without providing the precise position information of the polyp.
- **Patch-based**: Utilizes a combination of image patches or tiles to determine the presence and location of a polyp.
- **Bounding-box**: This method provides the location of the polyp through a bounding-box representation, typically consisting of coordinates of the upper right corner, height, and width. A regression layer is commonly employed for this purpose.
- **Semantic Segmentation**: In this approach, each pixel of the image is labeled as either polyp or background. Networks based on encoder-decoder blocks, typically employing Fully Convolutional Networks (FCNs), are commonly selected. The first half of the layers encode the image description by highlighting discriminative features, while the second half maps the low-resolution encoding into full input-resolution feature maps.

This is the task investigated in this work, and a schematic representation of a DL architecture for this task can be found in Figure 2.8.

Despite being a field that can potentially benefit from DL techniques, a series of problems regarding data quality and consistency across datasets imposes some challenges. For optimal performance in DL, it is well known in the research community that datasets with large numbers of examples are needed. Unfortunately, annotated colorectal polyp datasets are scarce and usually contain a limited amount of examples [11]. This requires models to be trained with augmented examples produced through random rotations, flips, and crops to enhance example content [23]. This is further aggravated by data disparity issues, data quality problems like incorrect bowel preparation, light reflections in the footage, and different viewpoints across datasets [23].

Even though working with polyp datasets can be challenging for the reasons mentioned above, it hasn't deterred researchers from exploring a wealth of DL architectures with satisfactory results. The main DL architectures used across colorectal polyp segmentation and detection tasks are reviewed in ELKarazle et al. [23] and summarized below:

- **U-Net**: U-Net is a widely used segmentation network in the medical imaging domain, consisting of an encoder, decoder, and residual connections.
- **SegNet**: SegNet is a semantic segmentation network following an encoder-decoder design, lacking residual connections, and utilizing pixel-wise classification layers.
- **Fully Convolutional Networks (FCN)**: FCN is a flexible segmentation network composed solely of convolutional layers, allowing variable input sizes and faster training speed.
- **Pyramid Scene Parsing Network (PSPNet)**: PSPNet incorporates a pyramid parsing module for accurate global context information aggregation using different-region context aggregation, it is less common in detection tasks compared to SegNet and U-Net.

Polyp classification also benefits from *transfer learning*, which consists of tuning knowledge acquired on a different (but to a degree, related task) to solve a problem on a different domain. This seems to be a preferred method for feature extraction and building polyp classification network backbones in the field (over creating CNN Models from scratch), probably due to the computational efficiency and ease of tuning they offer [23]. ELKarazle et al. [23] provides an overview of the most used pre-trained CNN architectures which are listed below:


**Figure 2.8:** Schematic of a semantic segmentation task of colorectal polyps (Peralta et al., 2021) [7]

- **VGG16**: A standard network with 16 convolutional layers that takes an input size of $224 \times 224$ pixel images used for colorectal polyp classification. It was trained using the ImageNet dataset [24].
- **VGG19**: Similar to VGG16, but using 19 convolutional layers instead of 16.
- **ResNet50**: ResNet50, is a widely used network in computer vision tasks. It consists of 50 layers and an input size of $227 \times 227$ pixels. It implements skip connections to allow smooth information flow and was also trained on the ImageNet dataset. ResNet50 is commonly used as the backbone for specialized polyp segmentation networks.
- **Xception**: Consists of 71 depthwise-separable convolutional layers, that require an input size of $299 \times 299$ pixels and was trained on ImageNet. It is, however, not as commonly used as VGG or ResNet.
- **AlexNet**: An eight-layer deep CNN with an input size of $227 \times 227$ pixels, trained on the ImageNet dataset.
- **GoogLeNet**: GoogLeNet, a 22-layer CNN that takes images $224 \times 224$ pixels. It was trained on the ImageNet dataset for image classification. It is mostly used in ensemble configurations.

# 3

# Methodology & experiment design

The purpose of this chapter is to systematically introduce the configuration of the experiments performed along with design decisions and assumptions made, in an effort to promote reproducibility of the results. This chapter is divided into 5 sections: Section 3.1 provides an overview of the datasets used and the data preprocessing pipeline, Section 3.2 summarizes the hardware configuration and software used, Section 3.3 reviews the hyperparameter tuning setup, Section 3.4 touches on the model architectures studied, Section 3.5 reviews the metrics used to evaluate model performance, and finally, Section 3.6 explains how the federated experiments were set up.

## 3.1 Datasets & data preparation

In this section, a description of the datasets used for each of the medical tasks described in Section 2.3 can be found. For each dataset, the preprocessing steps carried out before feeding the data to the models, their respective labeling strategies, and the data partitioning strategy for model development, evaluation, and testing are explained.

### 3.1.1 EEG signals dataset

- **Dataset description**: For the EEG signals task the dataset from Shafiezadeh et al. [8] paper on seizure prediction was used. This dataset was collected by the Epilepsy and

Clinical Neurophysiology Unit of the Eugenio Medea IRCCS Hospital in Conegliano (Italy). The EEG signals were obtained by using the international standard 10-20 EEG Scalp electrode positioning system, at a sampling rate of 256HZ. The position of the electrodes for each of the EEG channels can be seen in Figure 3.1. The dataset contains a total of 38 patients.

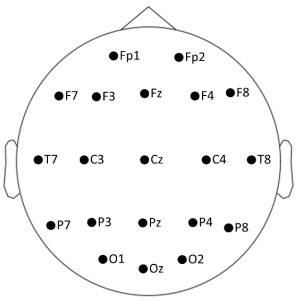


**Figure 3.1:** Scalp positioning for the 20 common EEG channels used by Shafiezadeh et al. [8].

- **Preprocessing**: In their work, the signal was processed through a combination of *notch filters* to remove power line interference, a *high pass filter* to remove Direct Current (DC) offset and fluctuations, and a low pass filter to highlight higher frequencies that might characterize abnormal brain activity. The preprocessed signal was divided into non-overlapping time windows of 5 seconds, and a set of 53 features was extracted using Python's *MNE-Features* package [8]. For a comprehensive list of the features extracted refer to Appendix A.5, Figure A.14. The preprocessed outputs were stored as a zipped *.npz* file containing Numpy arrays for both the features and labels. Working directly with the *.npz* file proved to be slow, so the outputs were extracted and consolidated into Pandas dataframes before finally exporting them as *.csv* files containing the final features and label matrices. The feature values are normalized using z-score normalization before being fed to a feed-forward neural network (FNN) as described in Section 3.4.

- **Data labeling**: The EEG signal can be categorized into four distinct stages as illustrated in Figure 3.2, which are the *interictal state*, which represents periods of regular brain activity between consecutive seizures, the *preictal state*, which is the period between approximately 60-90 mins before seizure onset, the *ictal state*, which corresponds to the seizure occurring, and the *postictal state*, which refers to the period immediately following a seizure for a few minutes [8]. Markers for the beginning and end of the *ictal state* were created by clinicians based on electroclinical information and EEG video monitoring [8]. To this end, one of the classes contains signals between 0 and 30 mins before the seizure (which is supposed to represent a seizure coming), and the other class had signals sampled 30 mins randomly from the *interictal state* (which represents regular brain activity). The prediction task consists of discriminating between these 2 classes.

- **Data partitioning**: The final dataset was partitioned into three splits: a *training* split (which was further split through K-Folds CV to acquire training and evaluation datasets for centralized tuning), a *validation* split for federated hyperparameter tuning, and a *testing* split that was set apart to be presented to the models until the end, for final model evaluation. A total of 15% of the examples were used for the validation set, 15% for the test set, and the remaining 70% for the training set. To address generalization performance on seizure prediction tasks, as demonstrated in Shafiezadeh et al. [8], two different dataset configurations were used for splitting, one that consisted of a stratified random sample of the examples (which led to IID data across splits with cross-contamination of patient data) and a patient-aware random sample of the data (which resulted in non-IID data that effectively separated patients data across splits).

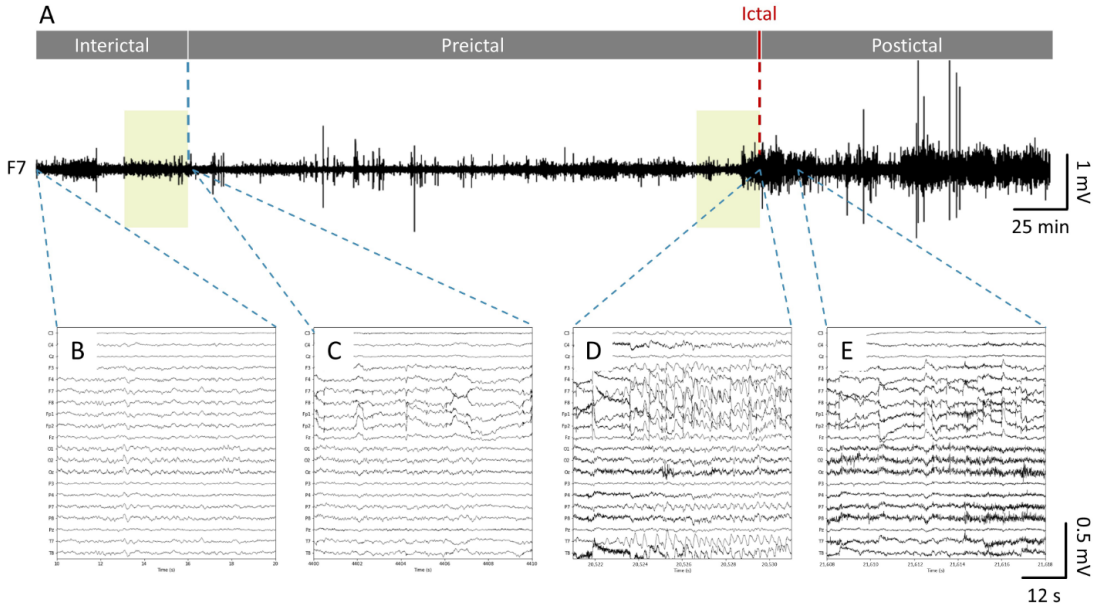


**Figure 3.2:** Segmentation for the EEG recordings of epileptic patients studied by Shafiezadeh et al. [8]. Panel A displays 480 minutes of recording from the F7 channel and its division into the *interictal*, *preictal*, *ictal*, and *postictal* stages. Highlighted in green are the zones of the recording used for the binary classification task. Panels B through E show 20 s magnifications of the recordings from the 20 channels at the beginning of each stage.

### 3.1.2 Polyp segmentation dataset

- **Dataset description**: For the polyp segmentation task, a series of standard benchmark polyp datasets were used following Lou et al. [9] research on the CaraNet architecture, with some minor modifications. Three of the original five polyp segmentation datasets

used in CaraNet: *Kvasir-Seg* [11], *ClinicDB* [9], and *CVC-ColonDB* [9] were used for different purposes. The number of examples, resolution, and approximate size of the polyps in each dataset are explained in Table 3.2.

- **Pre-processing**: Inputs to the Caranet CNN (explained in 3.3) are composed of a source image and a binary mask that specifies the Region Of Interest (ROI) surrounding the polyp as shown in Figure 3.3. Both images were resized to a resolution of $352 \times 352$ pixels, and a multi-scale training strategy with scaling set to 75%, 100%, and 125% resolution was used [9]. Multi-scale training is a commonly used technique in CNNs and image segmentation tasks that feeds scaled versions of images to the network to improve its generalization performance across object sizes, which was one of the objectives for developing CaraNet. Additionally, some image augmentations like random horizontal and vertical 90° flips with the probability of flipping set to 50% were used. Images are ultimately normalized before being fed to the network.

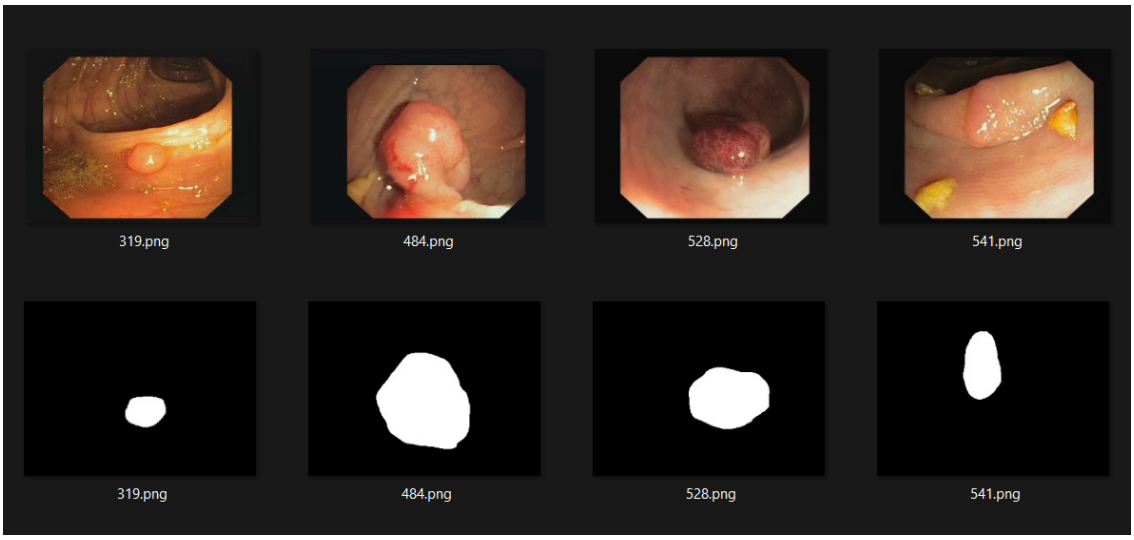


**Figure 3.3:** Polyp dataset example, the upper row displays the source images and the bottom row shows the ground truth binary masks used to identify the ROI of the polyps.

- **Data labeling**: Each dataset contains ground truth binary masks annotated by clinicians. The ROI corresponding to the polyp is represented by the positive pixels in the mask. The purpose of the polyp segmentation task is to generate a binary mask that closely resembles the ground truth mask.

- **Data partitioning**: In the CaraNet paper [9], a combination of samples from the Kvasir and CVC-ClinicDB datasets were used for training and another for evaluation purposes. Several other polyp datasets were used for testing and benchmarking purposes. This data-splitting strategy is kept as the intention was to replicate, to a degree, CaraNet centralized results before performing federated tests. This work uses 90% of Kvasir and

CVC-ClinicDB as the training set, the leftover 10% for model evaluation purposes, and a separate CVC-ColonDB used for testing to measure the model's generalization ability against an unseen dataset. Unlike the EEG dataset strategy, which considers stratified and patient-aware distributions of examples, only one configuration of the polyp dataset is studied here. Stratifying class distributions across partitions would mean redistributing negative and positive pixels from the source binary masks into the splits, which wouldn't make sense methodologically. For this reason, the data partitioning strategy specified outputs non-IID data across splits.

### 3.1.3 Dataset partitioning summary

The final distributions of samples for the *training*, *validation*, and *test* datasets for the seizure prediction and polyp segmentation tasks can be found in Tables 3.1 and 3.2 respectively.

| Configuration | Partition | Total Examples | Number of Patients | Positive Class Proportion |
|---|---|---|---|---|
| **Stratified Sampling** | *Training Set (70%)* | 1317890 | 38 | 20.20% |
| | *Validation Set (15%)* | 282405 | 38 | 20.20% |
| | *Test Set (15%)* | 282405 | 38 | 20.20% |
| **Patient-Aware Sampling** | *Training Set (68%)* | 1287420 | 28 | 20.20% |
| | *Validation Set (16%)* | 293960 | 6 | 19.55% |
| | *Test Set (16%)* | 301320 | 4 | 21.39% |

**Table 3.1:** EEG seizure dataset partitioning strategy.

| Partition | Total Examples | Dataset | Resolution | Dataset Examples | Object Size Ratio |
|---|---|---|---|---|---|
| *Training Set (90%)* | 1450 | Kvasir | 1070x1348 | 900 | 0.79% - 62.13% |
| | | CVC-ClinicDB | 288x384 | 550 | 0.34% - 45.88% |
| *Validation Set (10%)* | 162 | Kvasir | 1070x1348 | 100 | 0.79% - 62.13% |
| | | CVC-ClinicDB | 288x384 | 62 | 0.34% - 45.88% |
| *Test Set (Separate)* | 380 | CVC-ColonDB | 500x574 | 380 | 0.30% - 63.15% |

**Table 3.2:** Polyp dataset partitioning strategy.

## 3.2 Hardware & software characteristics

In this section, the configuration of the resources used to train the models is described.

- **Hardware configuration:** *Google Cloud Platform (GCP)* was used as a cloud provider for the experiments. *Cloud Storage* and *Compute Engine* services were used to persist the data inputs/outputs and perform computations. A Virtual Machine (VM) with the configurations specified in Table 3.3 was used for all the experiments. Federated workloads were simulated through the concurrent execution of FL client processes after the server is initiated. All experiments were run on an Nvidia GPU to leverage CUDA acceleration.

| GCP Tier | OS | GPU | VRAM | RAM | vCPUs | CPU Platform | Disk |
|---|---|---|---|---|---|---|---|
| a2-highgpu-1g | Debian 10 | Nvidia A100 | 40GB | 85GB | 12 | Intel Cascade Lake | 100 GB SSD |

**Table 3.3:** Hardware configuration of the GCP VM used to run experiments

- **Software configuration:** All models were coded in Python using *Pytorch* as the chosen DL framework. The FNN used for the seizure prediction task was built from scratch, while the code for the CNN of the polyp segmentation task was adapted from the original CaraNet paper from Lou et al. [9] *. Models were extended to the federated setting through the *Flower* FL framework from Beutel et al. [18]. A virtual conda environment was configured for all experiment-related tasks. A comprehensive list of all the Python packages and their respective versions can be found in Appendix A.4. The code developed for all the FL experiments can be found in a GitHub repository †.

## 3.3 Hyperparameter tuning

One of the open topics in FL research is hyperparameter optimization, as this is a difficult task to achieve in a federated setting due to parties likely not wanting to contribute to tuning before benefitting from FL [13]. In this section, the approach to performing hyperparameter tuning in the models is presented. Tuning is of vital importance to discover the optimal settings for each model and provides the means to perform fair comparisons of results across techniques and datasets (the best models are compared). Tuning was performed both in the centralized

*CaraNet [9] Github Repository: https://github.com/AngeLouCN/CaraNet

†Research Github Repository: https://github.com/andresespinalh/mt-federated-learning-in-healthcare

and federated setting by leveraging a combination of K-Folds CV, trial-and-error adjustment of parameters, and grid search exploration based on the complexity of the models.

### 3.3.1 Centralized setting

Hyperparameters for the seizure prediction task were calibrated by using a combination of K-Folds Cross-Validation with $K = 3$, and trial-and-error manual adjustment of tuning parameters to minimize the evaluation loss metric (the loss metric is described in Section 3.5). The training set presented in Table 3.1 was split through K-Folds CV, and the model was trained with $K - 1$ of the resulting folds, while the leftover fold was used for evaluation of the loss metric. This was used to calibrate the number of layers, units, and choice of the optimizer (SGD vs. Adam) while the number of epochs and learning rate parameters were tuned manually by running experiments and doing trial-and-error adjustments. The average evaluation loss across folds was used to select the best hyperparameters. For the polyp segmentation task values used in the original CaraNet paper, Lou et al. [9] were used, so tuning wasn't required.

### 3.3.2 Federated setting

The best architectural hyperparameters found in the centralized setting (number of layers, optimizer, local learning rate, etc.) were transferred to the federated setting for a fair comparison across approaches. The number of local epochs and FL rounds, similarly to the centralized setting, were calibrated manually through trial-and-error adjustments that optimized the evaluation metric. This calibration was done in a way that FL training approximately equated to the number of gradient descent steps done in the centralized setting. The batch size per client, in the case of the polyp segmentation task, was adjusted proportionally to the amount of FL clients summoned, to be able to accommodate all the data in VRAM.

Hyperparameter tuning in the federated setting proved to be a challenging task (especially for the polyp dataset), due to the long wait time required to get results back from the experiments. As a consequence of running FL in a simulated environment on a single physical machine, there is an overhead caused by instantiating the aggregation server, and the FL clients, besides the cost of parallel execution through concurrent processes. This is aggravated by the need of running hyperparameter grids through CV to guarantee the statistical significance of changes in the parameters, and their generalization capability. K-Folds CV with $K = 4$ was attempted on a 4-client FL configuration, with each client holding a fold of the data. Each client's fold was used for centralized evaluation in the FL server once, using the validation sets

defined in Section 3.1.3, and the aggregated models that resulted from averaging the parameters of the models trained in the remaining $K - 1$ clients in each FL round. Even though feasible, after estimating the time required to explore all the hyperparameter grids in the federated configuration, it was evident that it would be impractical for the purposes of this investigation to run experiments in this mode, so this approach was ultimately abandoned to avoid wasting computational resources of the GCP VM.

As a literature-aided alternative, the less computationally intensive approach from the original adaptive algorithms paper by Reddi et al. [4] was adopted. In their research, they choose the best hyperparameters for the federated algorithms *FedAdam*, *FedAdagrad*, and *FedYogi* based on the averaged training loss of the models over the last 100 communication rounds. The authors argue that the training loss at a specific round is a *"noisy estimate of the population-level training loss"* due to variable participation of FL clients per round in their experiments, and as a consequence decided to average over the last rounds of training. In this work, a similar approach was used as a naive replacement to K-Folds CV, but instead of using training metrics to choose the best parameters as in [4], metrics resulting from performing inference on a separate validation set as mentioned in Section 3.1.3 were used. Since variable participation of FL clients is out of the scope of this work, averaging evaluation metrics over a few epochs should provide reliable estimates of the generalization performance of the model.

The federated hyperparameter grid explored in this work was inspired by the best configurations across the tasks reported in Reddi et al. [4], and adjusted to the studied tasks through trial and error. These grids can be found in Appendix A.1. The target metric for the seizure prediction task was minimum validation *loss*, and for the polyp segmentation task, the maximum average validation *Dice* coefficient. The target metric was averaged over the last 125 of the 1000 rounds of training for the first task, and the last 3 of the 10 rounds for the second task (the last fourth of FL training rounds on both tasks). The process followed on both tasks to do grid search exploration is described as follows: first, an experiment is run for a specific federated algorithm and choice of hyperparameters. Then at each FL round, after training concludes on all clients, the aggregated parameters are collected and used to create a global model in the FL server, which holds the aforementioned validation datasets. After this, the constructed model is used to do inference on the validation dataset of the subject task to acquire the target evaluation metrics. After all experiments are run, the combination of hyperparameters that resulted in the best evaluation metrics is selected. The best combinations of hyperparameters per task and FL strategy as a result of these experiments can be found in Section 4.1.2. One weakness of this approach is that it might be too optimistic due to the choice of the train-validation-test split,

so future investigations using different subsets of the development set to train and evaluate the model are needed to further guarantee the robustness of the parameters chosen.

## 3.4 Model architectures

**Seizure Prediction Task (FNN).** The architecture used for the FNN is displayed in Figure 3.4. The network is composed of an input layer that consumes each of the 53 calculated features from the EEG signal, followed by 2 fully connected layers of 64 units each, and finishing with a single unit layer that outputs the prediction. ReLU is used as the non-linear activation function for each of the units. This configuration was chosen after performing hyperparameter tuning as explained in the results of Section 4.1.2. A total of 11,841 parameters were required to train this network.

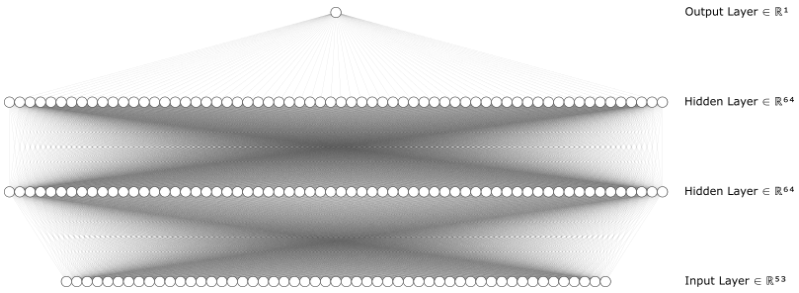


**Figure 3.4:** Architecture for the FNN used for the seizure prediction task.

**Polyp Segmentation Task (CNN).** CaraNet was used as the CNN architecture for this task as introduced by Lou et al. [9]. The architectural components for this CNN can be found in Figure 3.5. The network extracts a combination of low-level features like corners, edges, angles, etc. (features $f_1$, $f_2$), and abstract high-level features (features $f_3$, $f_4$, $f_5$). After this, a parallel partial decoder aggregates the high-level features to produce a global feature map to partially reconstruct the outputted mask. Low-level features contribute less to performance while also exhibiting a higher computational cost so only the high-level features are used [9]. These features are later processed through a lightweight Channel-wise Feature Pyramid (CFP) module to extract multiscale features across 4 channels that will aid in the detection of small polyps. To refine the location of the polyps self-attention techniques were used. The Axial Reverse Attention (A-RA) module was used to focus the attention of the network on the location of the polyp. Axial attention was used to factorize 2D attention into two 1D attention along height and width axes to save computational resources. A sigmoid function is attached at the end

of the network to output the final binary mask prediction. The total number of parameters needed to train this model was 46,642,560.

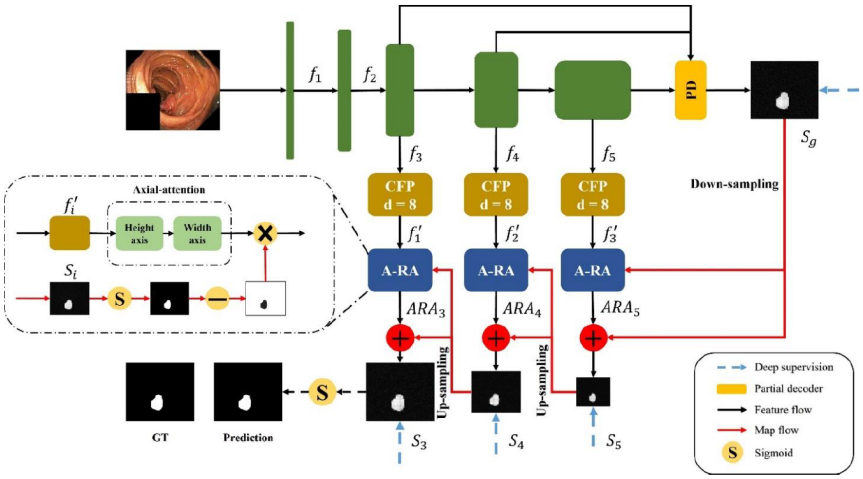


**Figure 3.5:** CaraNet architecture from Lou et al. [9].

## 3.5 Evaluation metrics

- **Seizure Prediction Task (FNN):** Standard binary classification metrics like *Binary Cross-Entropy (BCE) loss, accuracy, sensitivity, specificity, and area under the receiver operating characteristic (AUROC)* were tracked for training, validation, and test metrics. This aligns with the set of metrics used by Shafiezadeh et al. [8] in their seizure prediction experiments. BCE was used as the loss function due to its favorable mathematical properties in binary classification tasks. AUROC was preferred as the main evaluation metric over accuracy, due to its robustness to class imbalance. Sensitivity and specificity metrics are also used as they are standard metrics used in medicine for diagnostics. The definition of each of these metrics is reported in Figure 3.6.

- **Polyp Segmentation Task (CNN):** The Sørensen–Dice coefficient is a standard metric used in object segmentation tasks to compare the pixel-wise results between the ground truth mask and the predicted segmentation masked outputted by the network [11]. This metric considers both the sets' intersection and sizes, with values ranging between 0 and 1. A coefficient of 0 indicates no overlap and 1 indicates a perfect match between the predicted and true segmentation masks. Similarly, Intersection Over Union (IoU) is often used as a threshold-based metric to determine correct detections or segmentations. The binary segmentation masks are often imbalanced datasets, due to the larger proportion

$$\text{BCE Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right)$$

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

$$\text{Sensitivity (Recall)} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}$$

$$\text{AUROC} = \int_{-\infty}^{\infty} \text{TPR}(f)\, d(\text{FPR}(f))$$

**Figure 3.6:** Evaluation metrics for the seizure prediction task (FNN).

of negative pixels than positive pixels in the images. As a result, the balance between false positives and false negatives is important. The dice coefficient was adopted over other metrics (like IoU) as it is the evaluation metric used in the CaraNet network [9] and it takes into account class imbalances inherent to the polyp datasets used. Finally, a combination of the weighted Intersection over Union (IoU) metric and weighted Binary Cross-Entropy (BCE) loss was used as the loss function for the model. The metrics used for this task are presented in Figure 3.7.

$$\text{Dice} = \frac{2|X \cap Y|}{|X| + |Y|} = \frac{2TP}{2TP + FP + FN}$$

$$\text{IoU} = \frac{|X \cap Y|}{|X| \cup |Y|} = \frac{TP}{TP + FP + FN}$$

$$\text{Loss} = L^w_{IoU} + L^w_{BCE}$$

**Figure 3.7:** Evaluation metrics for the polyp segmentation task (CNN).

## 3.6 Federated setup & experiment design

In this section, the configuration of the federated experiments is reviewed. For the *seizure prediction task*, configurations of *2, 4, 8,* and *16* clients were used, for the *polyp segmentation task*, the configurations were *2, 4, 6,* and *8* clients. The datasets specified in Section 3.1.3 were used as a baseline to develop the federated datasets for this stage. Specifically, both datasets' training sets were split in the configurations mentioned by distributing the data in several partitions for local training.

For the *EEG dataset*, training data was partitioned using both *stratified sampling* and *patient-aware sampling* to yield two different configurations analogous to the configurations found in Section 3.1.3. The objective of the splitting strategy was to evaluate the impact of IID and non-IID data in the federated setting across a series of averaging algorithms. It is worth noting that crossing patient data across nodes defeats the purpose of FL, so this strategy was used purely for experimental purposes to make a comparison against the centralized setting easier. In the *polyps dataset*, the data from the Kvasir and CVC-Clinic DB's was distributed across clients uniformly, in such a way that a specific FL client only contained data from one of the datasets, to simulate a real case scenario of hospitals having distinct patient data.

The remaining validation and test sets in Section 3.1.3 have the same purpose in the FL setting as their centralized counterparts. The first dataset is used to perform per-federated-round centralized evaluation in the FL server, by constructing a model with the rounds' aggregated parameters and then doing inference on the validation set. After all FL rounds have been finished, the best model is acquired and the test set is used to evaluate the final model generalization performance. Performing model validation and testing across the same sets in both the centralized and federated settings ensures that results are comparable.

Experiments were executed only once, due to the long waiting times required for the federated experiments to finish (especially with the CaraNet model). Due to the high cost of the cloud resource with GPU used ($3.75 hourly) and the number of experiments planned (especially in hyperparameter optimization), it was decided to draw conclusions based on single-run experiments. DL models are often sensitive to the initial choice of weights, which is often done randomly. For this reason, future investigations with repeated runs of experiments might be required to ensure the independence of results on model weight initializations and their statistical robustness.

# 4

# Experimental results & discussion

This chapter presents the results of the experiments carried out in this research. Section 4.1 presents the results that address each of the research questions defined in Section 1.2. Section 4.2 wraps up this chapter by providing an overview of the main discoveries found, the interpretation of the results, and the identification of potential research directions.

## 4.1 Experimental results

In this section, evidence of the main scientific findings gathered from the centralized and federated experiments is presented. This section is further organized into 8 subsections that address the specific research questions from Section 1.2. Subsection 4.1.1 provides an overview of the data distributions, Subsection 4.1.2 presents the best hyperparameters found, Subsections 4.1.3 and 4.1.4 present the centralized and federated results, Subsection 4.1.5 shows an analysis on FL client size, Subsection 4.1.6 reviews the convergence behavior of the models, and finally Subsections 4.1.7 and 4.1.8 provide an overview of the execution time and resources consumption of the experiments. All the results presented in the next sections can also be found in an interactive public Tableau dashboard *.

*Results Dashboard: https://public.tableau.com/app/profile/andresespinalh/viz/FLAnalysisinHealthcare/Main

### 4.1.1 Exploratory data analysis (EDA)

#### Centralized datasets

**EEG dataset.** The distribution of classes across patients and partitioning approaches are explored in Figures 4.1 and 4.2 respectively. As it is evident from the first figure, most signals contain normal brain activity, indicating only a few of the signals correspond to an upcoming seizure. The partitioning strategy described in Section 3.1.1 is also shown here. As seen in this visualization, the stratified sampling strategy outputs IID data with cross-contamination of samples from different patients (notice how data from all the patients exist in every partition), while the patient-aware approach generates non-IID data but separates patient data across splits (one particular patient exists only in one partition at the time). The original dataset manifests variable proportions of examples and class distributions across patients as shown in the second figure, which indicates varying lengths of EEG activity being sampled across subjects.

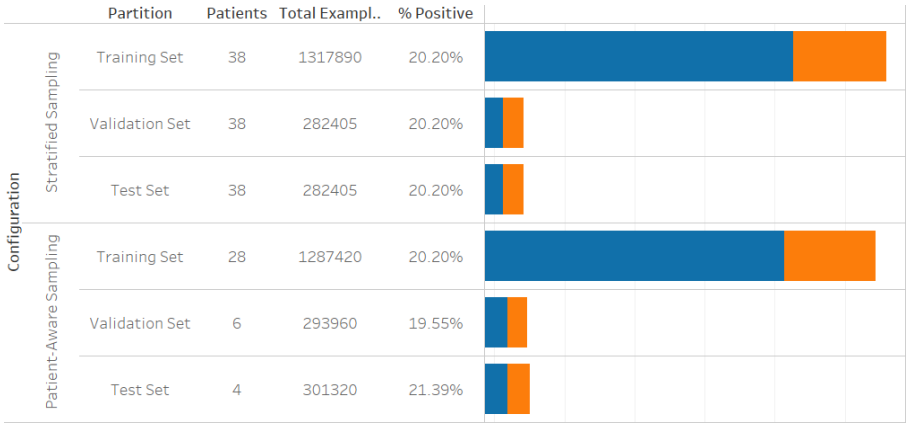


**Figure 4.1:** Class distribution for the EEG dataset. Label 0 (blue) indicates a period of regular brain activity (no seizure) and label 1 (orange) indicates an atypical surge in brain activity (seizure incoming).

**Polyp dataset.** An analogous data distribution analysis is shown in Figure 4.3. By interpreting black pixels in the colonoscopy images as the negative class (non-polyp regions) and white pixels as the positive class (which contains the polyp's ROI) a binary label is produced. The total pixel count per label across all images is analyzed per partition. On average, samples drawn from the Kvasir dataset exhibit double the count of positive pixels (approximately 16%) compared to the other datasets (approximately 8%). This behavior is expected across splits due to the bigger sizes of polyps found in the Kvasir dataset (as seen in Table 3.2). Additionally, polyp

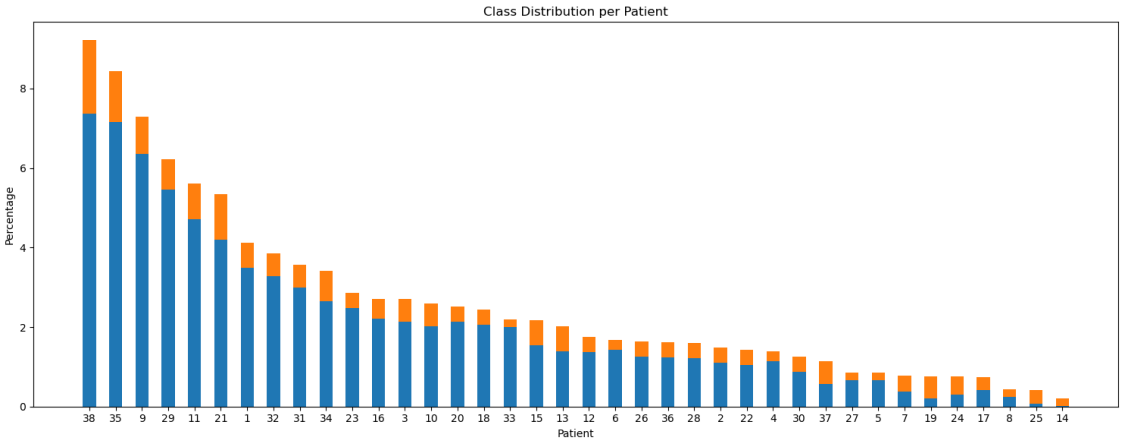


**Figure 4.2:** Per patient class distribution for the EEG dataset. Label 0 (blue) indicates a period of regular brain activity (no seizure) and label 1 (orange) indicates an atypical surge in brain activity (seizure incoming).

regions are relatively small when compared to the complete source image so having small pixel counts for the positive labels is expected. Regardless of the differences in class labels, all of the splits manifest class imbalances similar to the ones found in the EEG dataset.

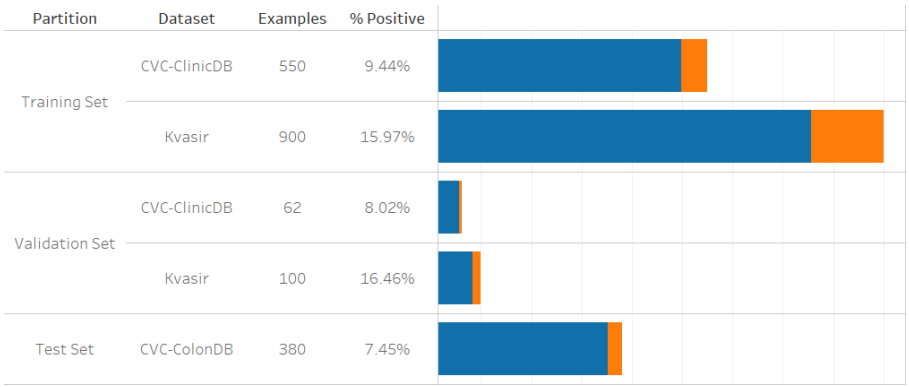


**Figure 4.3:** Class distribution for each of the polyp datasets across centralized partitioning strategy. Label 0 (Blue) is assigned to black pixels that do not contain the ROI, and 1 (orange) represents white pixels that contain the ROI containing the polyp.

## Federated datasets

**EEG dataset.** The FL distribution of samples for the seizure prediction task is displayed in Figure 4.4. All FL clients across all FL configurations (2, 4, 8, 16 clients) contain the same proportion of positive classes in the stratified sampling setup, while samples are uniformly distributed across FL clients. Each client holds the same percentage of data from the original train-

ing set. This is not the case in the patient-aware sampling setup, in which each client might have slightly different proportions of positive labels and a variable number of examples assigned to each FL client. It is worth noting that as a side-effect of increasing client size, local models have less data to learn from.

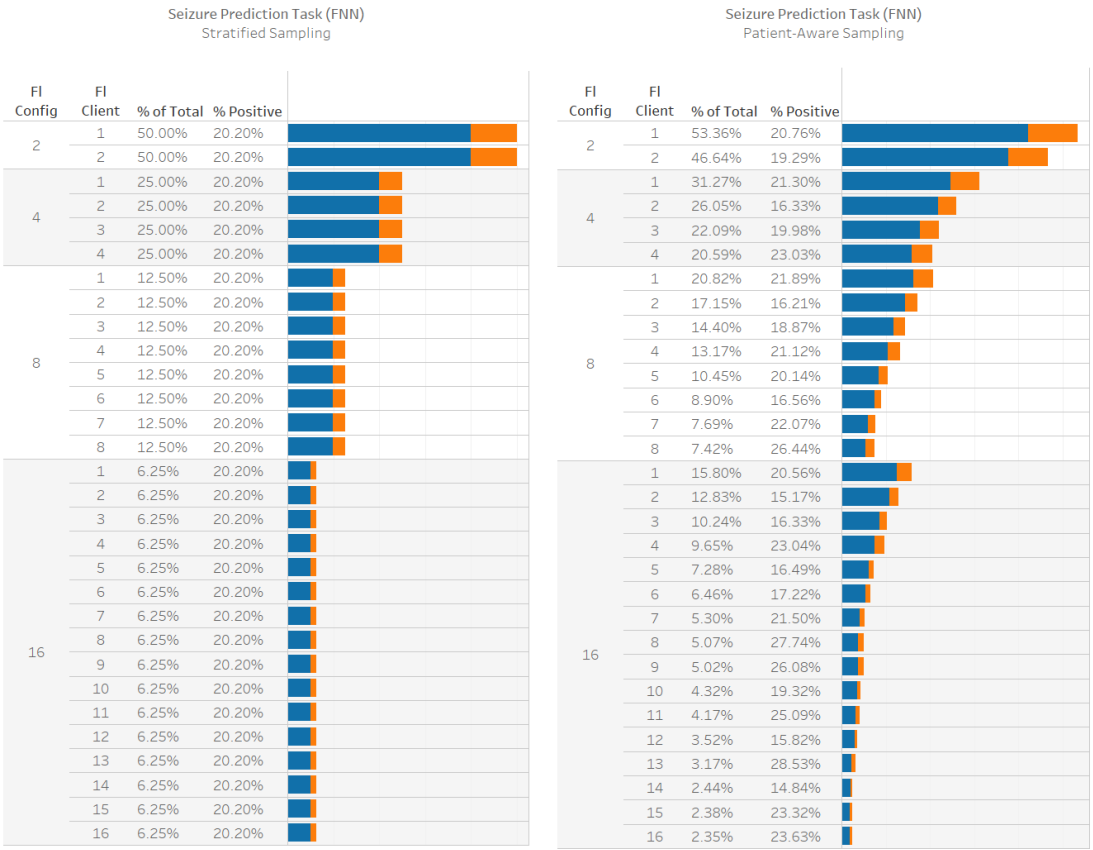


**Figure 4.4:** Distribution of class labels in the EEG dataset per FL configuration (2, 4, 8, and 16 clients), and FL client for the stratified sampling and patient-aware datasets. The blue portion of the bar indicates the proportion of the negative labels, and the orange portion the proportion of positive labels.

**Polyp dataset.** The polyp segmentation dataset showcases the same behavior as the patient-aware EEG dataset as it is shown in Figure 4.5. Each FL client contains a variable set of images and positive pixels (polyp ROI). The data was partitioned in a way that samples from the Kvasir and CVC-ClinicDB datasets didn't coexist in a particular federated node.

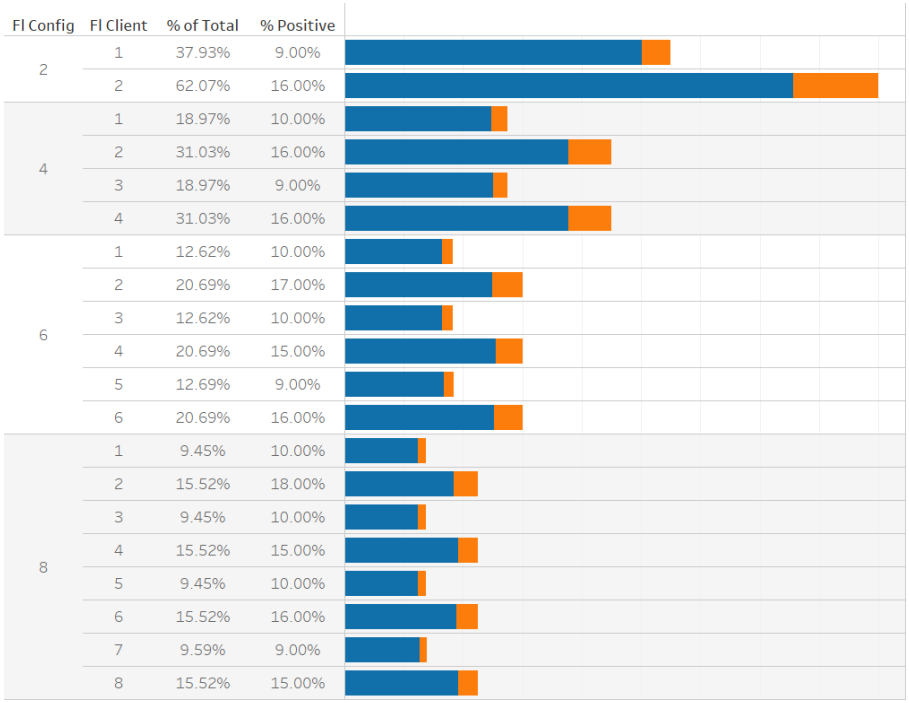


**Figure 4.5:** Distribution of class labels in the polyp dataset per FL configuration (2, 4, 6, and 8 clients) and FL client. The blue portion of the bar indicates the proportion of the negative labels (black pixels in the segmentation mask), and the orange portion is the proportion of positive labels (white pixels in the segmentation mask).

### 4.1.2 Hyperparameter tuning

#### Centralized models

**Seizure prediction task (FNN).** The final configuration used consisted of an FNN with 2 layers, containing 64 units each. Further increases in model complexity proved to be unnecessary, and even detrimental to model performance. Complex networks consisting of more than 2 layers and 64 units exhibited heavy overfitting without a significant improvement in evaluation metrics. The network was trained for a total of 3000 epochs using whole batches, and as a result, tuning mini-batch size wasn't required. Complete batches were used for training as the whole EEG dataset could be accommodated in memory. The network was left to train for a little longer than it took to converge, to be able to compare convergence trends across approaches. Both SGD and Adam optimizers were tried, but ultimately the former was chosen due to its faster convergence capabilities and robustness to the learning rate hyperparameter.

The optimizer was parametrized with a learning rate of 1e-2 and $\beta_1$, $\beta_2$ parameters set to 0.9 and 0.999 respectively. These hyperparameters were used across both the stratified and patient-aware datasets. Due to class imbalances inherent to EEG datasets, since most signals won't result in a seizure, weighted loss was implemented to balance the proportions of negative and positive labels. An equivalent approach was attempted by replacing loss weighting with positive class oversampling, and the results were very similar. To avoid adding more complexity to the data partitioning strategy in the federated setting the first strategy was adopted.

**Polyp segmentation task (CNN)**: For the CaraNet model, all of the original values from Lou et al. [9] except for the number of epochs and mini-batch sizes was adopted as the optimal hyperparameters. Mini-batches of size 6 containing images of $352 \times 352$ pixels resolution were used. Adam was used as the optimizer with a learning rate of 1e-4, a weight decay of 1e-4, a momentum of 0.9, and $\beta_1$, $\beta_2$ parameters set to 0.9 and 0.999 respectively. The training was done for a total of 15 epochs, clipping gradients by a value of 0.5 after each gradient descent step.

### Federated models

**Seizure prediction task (FNN)**: The amount of FL rounds and local epochs was tuned manually through trial-and-error experiments runs until similar results to the centralized setting on the validation performance metrics were achieved. The optimal values found were 500 FL rounds with 10 epochs of local work each. Hyperparameters specific to each federated strategy were explored through a grid search approach. The optimal combination of parameters that minimized evaluation loss against the two EEG datasets and averaging strategy configurations is presented in Table 4.1. The experiment grids containing the average evaluation loss per hyperparameter set can be found in Appendix A.2. Specifically, grids for FedAdam (Figure A.1), FedAdagrad (Figure A.2), FedYogi (Figure A.3) and FedProx (Figure A.4) were explored. FedAvg required no further tuning besides the number of averaging rounds and local epochs.

**Polyp segmentation task (CNN)**: Similar to the FNN, manual tuning was done for the number of epochs and FL rounds. The FL model was trained for a total of 10 FL rounds, performing 2 epochs of training in each FL client. Since each client loaded a batch of data concurrently to memory during training, adequate memory management was required to avoid running out of the 40GBs of available memory. To do this, the batch size used for each client configuration (2, 4, 6, 8) was calibrated manually to allow the simulated clients to fit the data in available VRAM (batch sizes of 16, 8, 4, and 2 respectively). Results for the federated strategy-

specific hyperparameters can be reviewed in Appendix A.2. Similarly, grids for FedAdam (A.5), FedAdagrad (A.6), FedYogi (A.7) and FedProx (A.8) were explored and the best hyperparameters can be found in Table 4.1.

| Task | Partitioning Strategy | Federated Strategy | $tau$ | $eta$ | $eta_l$ | $mu$ |
|---|---|---|---|---|---|---|
| **EEG Seizure Prediction (FNN)** | *Stratified* | FedAdam | 10e – 1 | 10e – 0 | 10e – 0 | |
| | | FedAdagrad | 10e – 2 | 10e – 2 | 10e – 0 | |
| | | FedYogi | 10e – 5 | 10e – 3 | 10e – 2 | |
| | | FedProx | | | | 10e – 0 |
| | *Patient-Aware* | FedAdam | 10e – 2 | 10e – 1 | 10e – 1 | |
| | | FedAdagrad | 10e – 0 | 10e – 3 | 10e – 3 | |
| | | FedYogi | 10e – 0 | 10e – 3 | 10e – 2 | |
| | | FedProx | | | | 10e – 3 |
| **Polyp Segmentation (CNN)** | *Standard* | FedAdam | 10e – 0 | 3e – 0 | 10e – 1 | |
| | | FedAdagrad | 10e – 0 | 10e – 0 | 10e – 0 | |
| | | FedYogi | 15e – 0 | 5e – 0 | 10e – 2 | |
| | | FedProx | | | | 10e – 4 |

**Table 4.1:** Best combination of federated hyperparameters found for the seizure prediction (FNN) and polyp segmentation (CNN) tasks across datasets and FL strategies.

### 4.1.3 Centralized model evaluation

**Seizure prediction task (FNN).** Training results for the FNN are shown in Figure 4.6, and the trajectory of the loss function is displayed for both training and validation datasets. The stratified sampling strategy exhibits a healthy learning curve where the training and validation loss decreases until reaching the convergence of the model, with little overfitting. The patient-aware approach on the other hand suffers from heavy overfitting from the beginning as can be seen from the divergence of training and validation loss curves.

A crosstab containing the final results of both models across all metrics can be found in Figure 4.7. The stratified sampling approach achieved an AUROC of 0.7 on the test set while keeping a balance in the performance of the sensitivity and specificity metrics and the model encountered its best performance at epoch 1962 out of 3000. The patient-aware approach outputted an AUROC of 0.53 which is in practical terms a chance level result, and the best validation loss was found at the beginning of the training process. Furthermore, the sensitivity

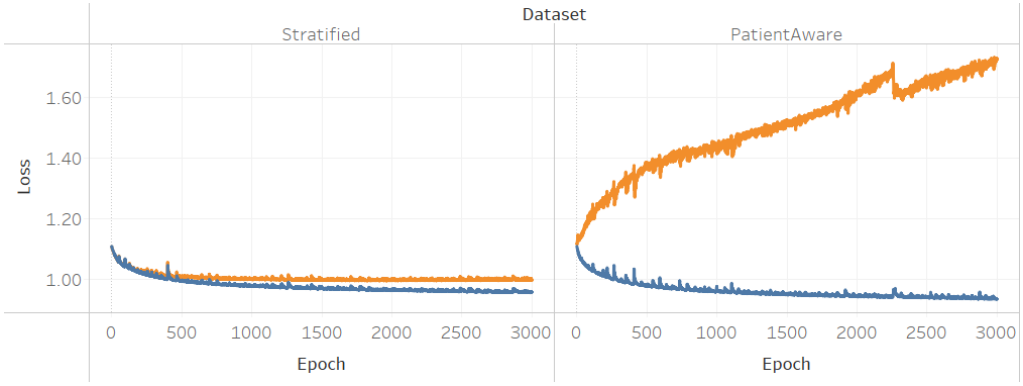


**Figure 4.6:** Training results for the centralized seizure prediction task (FNN). Training loss (blue) against validation loss (orange) is plotted against the two dataset sampling approaches.

and specificity metrics are imbalanced due to the model overfitting the majority class (label 0) as observed in Figure 4.6.

| Dataset | Elapsed Time (Seconds) | Epoch Best Model | Test Loss | Test Accuracy | Test Sensitivity | Test Specificity | Test AUROC | Train Loss | Train Accuracy | Train Sensitivity | Train Specificity | Val. Loss | Val. Accuracy | Val. Sensitivity | Val. Specificity |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Stratified | 170.23 | 1,962 | 1.01 | 0.63 | 0.65 | 0.62 | 0.70 | 0.96 | 0.64 | 0.67 | 0.64 | 0.99 | 0.62 | 0.66 | 0.61 |
| PatientAware | 167.22 | 1 | 1.09 | 0.65 | 0.21 | 0.77 | 0.53 | 1.11 | 0.63 | 0.33 | 0.71 | 1.12 | 0.71 | 0.16 | 0.84 |

**Figure 4.7:** Model results for the centralized seizure prediction task (FNN). Test metrics are displayed accompanied by a snapshot of the training and validation metrics in the training epoch that minimized validation loss.

**Polyp segmentation task (CNN)** The training results for this task are displayed in Figure 4.8. The network showcases an increase in the mean dice metrics through time with little overfitting. The detailed crosstab with the final results can be found in Figure 4.9. The network achieved a mean dice result of 0.75 in the test set with its best validation dice (0.88) in epoch 16.

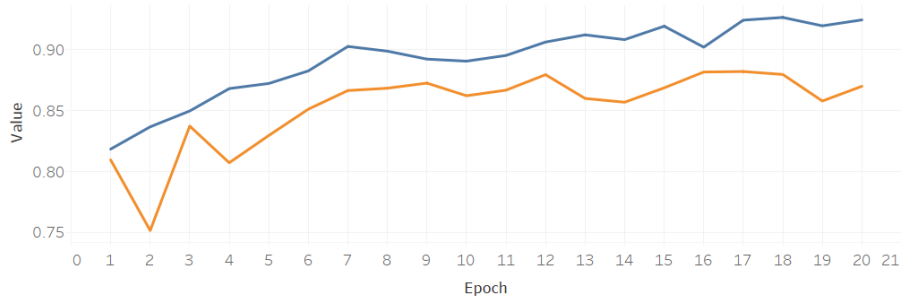


**Figure 4.8:** Training results for the polyp segmentation task (CNN). Mean Dice is plotted for training (blue) and validation (orange) datasets. Dice values closer to 1 indicate a closer match of inferred and ground truth segmentation masks.

| Elapsed Time (Seconds) | Epoch Best Model | Test Mean Dice | Train Loss | Train Mean Dice | Val. Mean Dice |
|---|---|---|---|---|---|
| 2,714.62 | 16 | 0.75 | 1.17 | 0.90 | 0.88 |

**Figure 4.9:** Model results for the polyp segmentation task (CNN). Test mean dice is displayed accompanied by a snapshot of the training and validation metrics in the training epoch that minimized validation loss.

### 4.1.4 Federated model evaluation

**Seizure prediction task (FNN).** Aggregated results across client configurations can be found in Figure 4.11. Overall, models trained with the stratified sampling strategy achieved a higher AUROC on average, with most of the strategies achieving a performance of 0.7, which is equivalent to the AUROC found in the centralized setting. Analogous to the results in the centralized setting, the patient-aware approach manifested a decrease in performance on the AUROC metric. However, FedAvg, FedAdam, and FedProx strategies scored higher AUROC scores (0.6, 0.59, 0.58) than the ones achieved in the centralized experiments (0.53). The mean AUROCs per federated strategy are relatively close to each other across datasets, indicating results across strategies at this level of aggregation are similar. A detailed table containing the results of all experiments across all the evaluation metrics tracked can be found in Appendix A.3, Figure A.10.

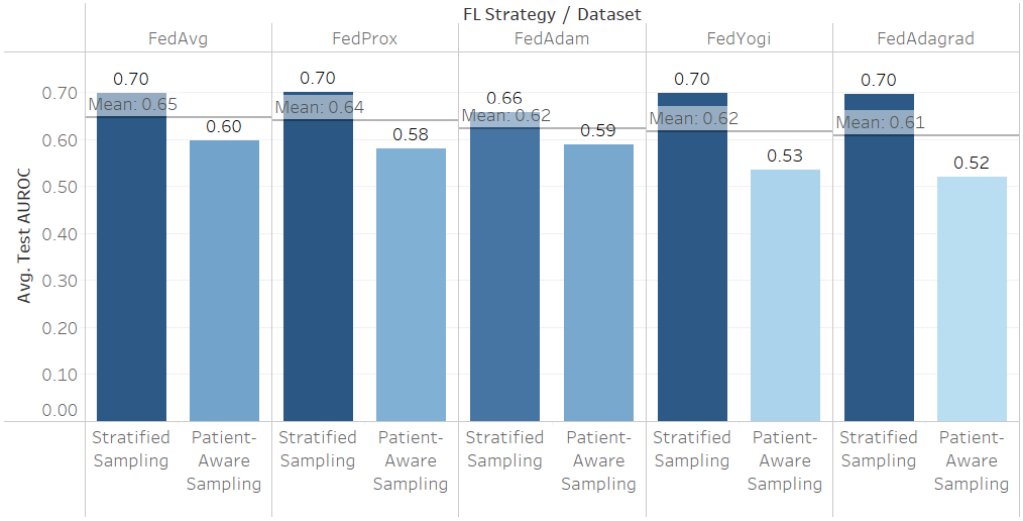


**Figure 4.10:** Mean test AUROC results for the federated seizure prediction task (FNN) across FL client sizes. Each panel represents one of the federated strategies evaluated, and each panel displays the results for the stratified and patient-aware datasets. Mean AUROC across datasets and FL client configurations is reported per federated strategy panel (line).

**Polyp segmentation task (CNN).** For this network, federated averaging done through FedAvg, FedProx, and FedAdagrad strategies resulted in the highest test dice coefficients (0.76, 0.75, 0.75). These values are similar to the ones encountered in the centralized setting, but unlike the seizure prediction task, there is a more significant difference between the performance of the averaging strategies. A crosstab detailing the results of all experiments across all the evaluation metrics tracked can be found in Appendix A.4, Figure A.9.

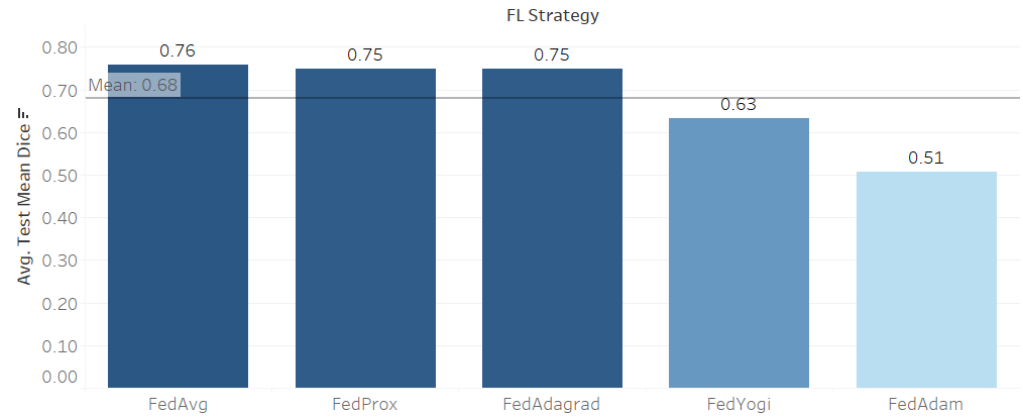


**Figure 4.11:** Mean test dice results for the federated polyp segmentation task (CNN) across FL client sizes. Each bar represents one of the federated strategies evaluated. Mean dice across strategies and FL client configurations are shown as a line.

### 4.1.5 Effect of FL client size

**Seizure prediction task (FNN).** Model evaluation metrics on the test set grouped by federated strategy, dataset, and client configuration are shown in Figure 4.12. Models trained with the stratified sampling dataset showcase a similar behavior in performance metrics across client configurations disregarding the averaging strategy used (FedAdam seems to be marginally an exception to this). For the models trained with the patient-aware strategy differences are more notorious across client configurations. There is a very slight decrease in model performance in some of the algorithms as the client size increases.

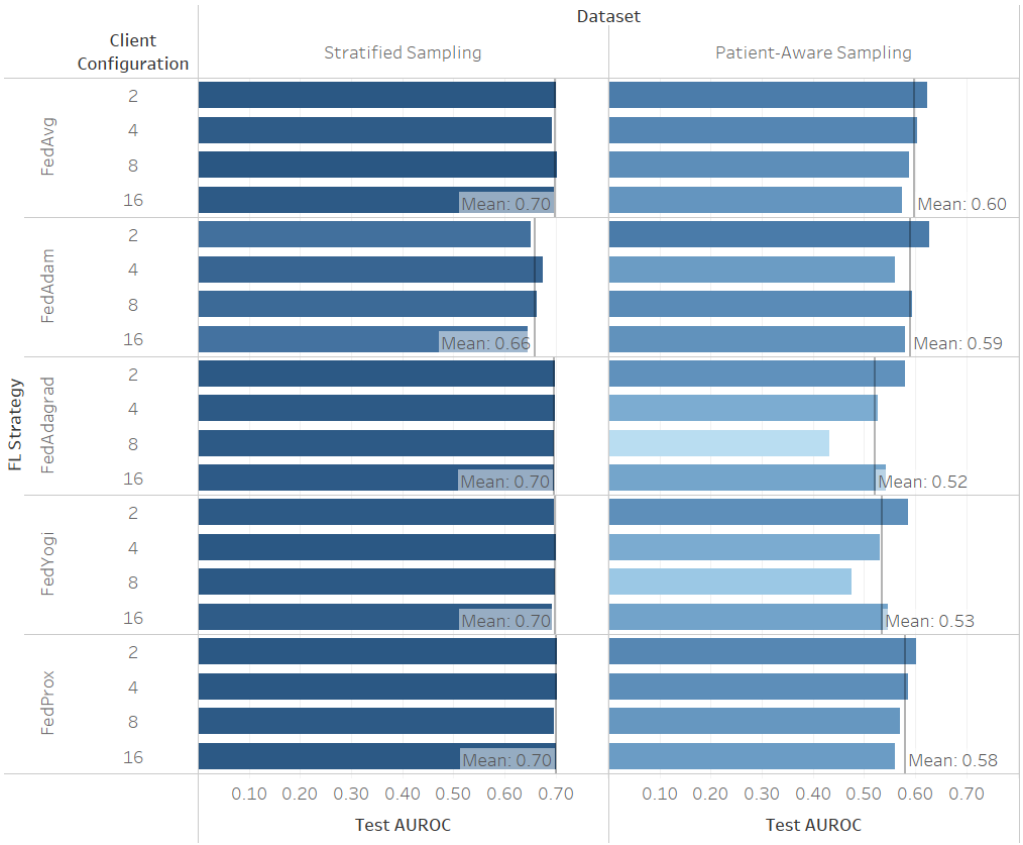


**Figure 4.12:** Model results for the federated seizure prediction task (FNN) partitioned by FL client sizes. Each bar represents one of the federated strategies and client configurations evaluated. The mean AUROC metric across strategies is shown as a line.

Similar to the centralized setting, models trained with the patient-aware strategy exhibit imbalances between the sensitivity and specificity metrics as illustrated in Figure 4.13. This graph outputs an overlap between the two areas when sensitivity and specificity are balanced as seen

across the algorithms in the stratified sampling column, with FedAdam being a subtle exception. The patient-aware approach results in imbalances, but curiously across configurations of FL clients, the class that is overfitted evolves. For instance, strategies like FedAdam and FedYogi favor the negative and positive classes in different FL configurations, while the other algorithms seem to favor one of these metrics across all client configurations.

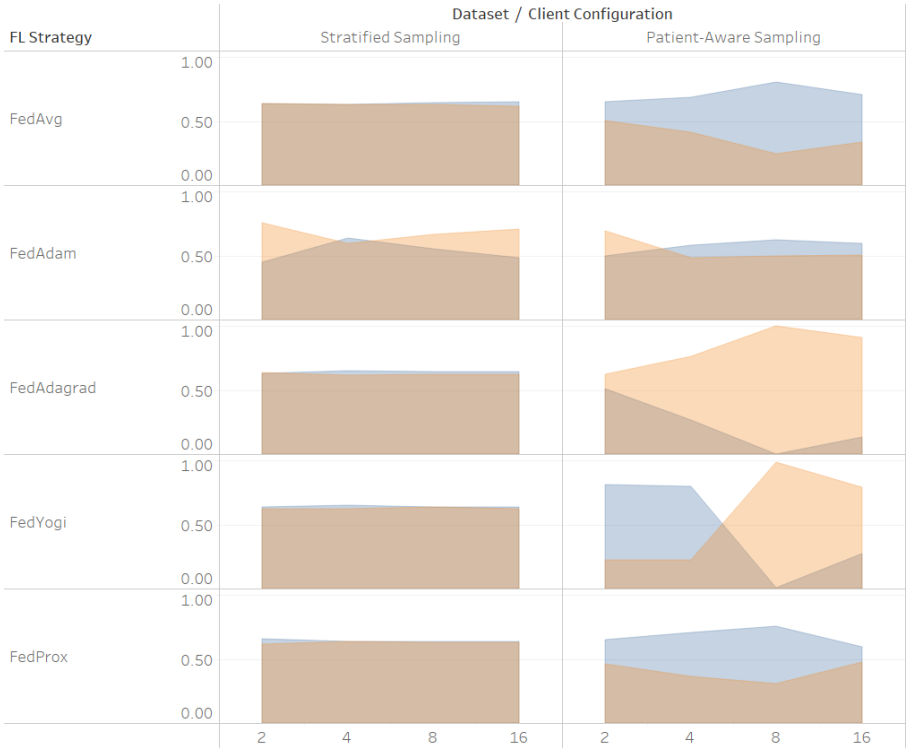


**Figure 4.13:** Graph showing the balance between sensitivity (blue) and specificity (orange) test metrics for the federated seizure prediction task (FNN). The vertical panels represent the stratified and patient-aware datasets, and the horizontal panels the federated strategy. Inside each cell, the horizontal axis displays the amount of FL clients, and the vertical axis the sensitivity/specificity values.

**Polyp segmentation task (CNN).** Evaluation metrics across client configurations for the polyp segmentation task can be found in Figure 4.14. A similar decrease in model performance is present here, with FedAdam producing the most dramatic decrease in the mean dice metric (from 0.64 to 0.38) when increasing the amount of FL clients (from 2 to 8), and FedYogi becoming unstable in configurations with a few federated clients (from 2 to 4). FedAvg and FedProx produced the best results with 4 and 2 FL clients respectively, both achieving a score of 0.78, which is 3% higher than the best score achieved in the centralized setting.

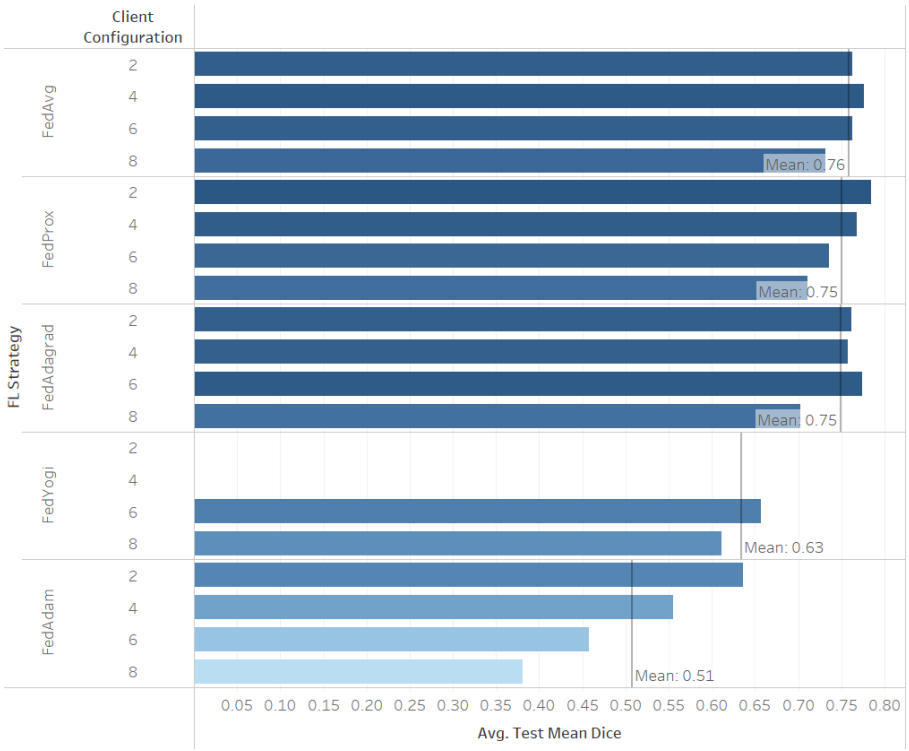


**Figure 4.14:** Mean test dice results for the federated polyp segmentation task (CNN) partitioned by FL client sizes. Each bar represents one of the federated strategies and client configurations evaluated. Mean dice across strategies are shown as a line. Empty bars (FedYogi with 2 and 4 clients) indicate numerical instabilities in the models that resulted in NaNs.

### 4.1.6 Federated model convergence

**Seizure prediction task (FNN).** The mean training loss function behavior across all client configurations can be found in the first two panels in Figure 4.15. Convergence curves for the stratified dataset for the FedAvg, FedAdagrad, and FedYogi algorithms look very similar, while FedAdam has a noisy training curve and FedProx a rather precipitated convergence behavior. In the patient-aware dataset, FedAvg has a behavior analogous to the stratified loss curve while FedAdaGrad and FedYogi continue on a decreasing trend by the end of training. Finally, FedProx manifests a more stable training curve in the heterogeneous setting.

The same trends for the training loss function for each of the models across client configurations can be found in Appendix A.4, Figure A.11. Overall, adding more FL clients doesn't seem to affect the shape of the convergence curve too much across algorithms, as is evident from the relatively similar convergence curve shapes across client configurations. However, in some cases like the FedProx algorithm in the stratified dataset, the loss starts ramping up as the amount of FL clients increases. FedAvg, FedAdagrad, and FedYogi on the other hand manifest a decrease in the loss function in the patient-aware dataset as the amount of FL clients increased, being the last 2 algorithms the ones with the most pronounced decrease. FedAdam is the only algorithm that exhibits unfavorable convergence curves across both datasets.

A comparison of the baseline FedAvg algorithm against the adaptive strategies (FedAdam, FedAdagrad, and FedYogi) is presented in the first two panels of Figure 4.16. When analyzing all the training curves on the same scale, it is evident that FedAvg, FedAdagrad, and FedYogi have relatively similar convergence behavior in the stratified dataset, but the former two exhibit different behavior in the patient-aware setting. FedAdam's loss curve starts diverging (and increasing) after the first rounds of training in both scenarios, while FedAvg appears to be the most consistent curve with the minimum average loss across datasets.

**Polyp segmentation task (CNN).** The mean convergence behavior across all client configurations can be found in the third panel in Figure 4.15. FedAvg, FedAdagrad, and FedProx manifest the greatest decreases in training loss, while FedAdam and FedYogi finish training with higher training loss values which coincides with the unfavorable performance metric results found in the federated evaluation.

Detailed convergence curves per client configuration are illustrated in Appendix A.4, Figure A.12. After inspection of the minimum loss values produced by each model, an increasing trend of training loss can be detected as the amount of FL clients increases across all algorithms. Overall, models trained with fewer FL clients performed relatively better in terms of training

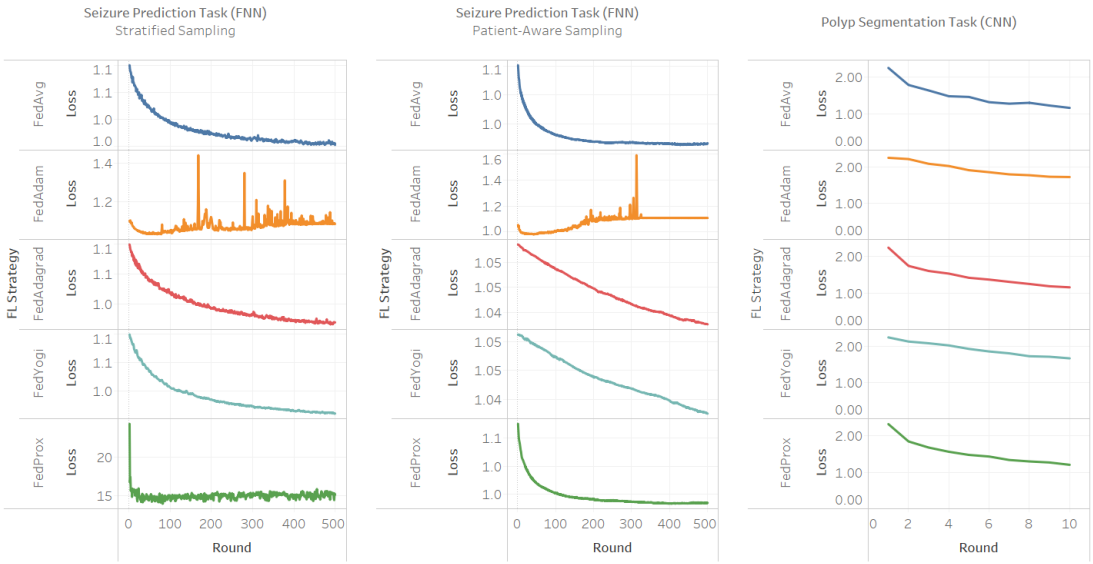


**Figure 4.15:** Training loss convergence curves for all tasks. The first two panels display the seizure prediction tasks curves with the stratified and patient-aware datasets respectively, while the third panel displays the polyp segmentation task curves. Rows represent each FL strategy employed (with different colors) and training loss is plotted per FL round in each cell. Each cell has an independent scale based on the loss function values for each configuration.

loss, which translates into better performance when reviewing the test mean dice values shown in Figure 4.14. Additionally, most of the models exhibited the best validation mean dice score in the last round of training (which differs from the results in the seizure prediction task).

FedAdam is compared to the adaptive algorithms (FedAdam, FedAdagrad, and FedYogi) in the third panel of Figure 4.16, for the seizure prediction task. Here, FedAvg and FedAdagrad produce the models with the fastest convergence, while FedAdam and FedYogi produce similar convergence curves with a higher average loss.

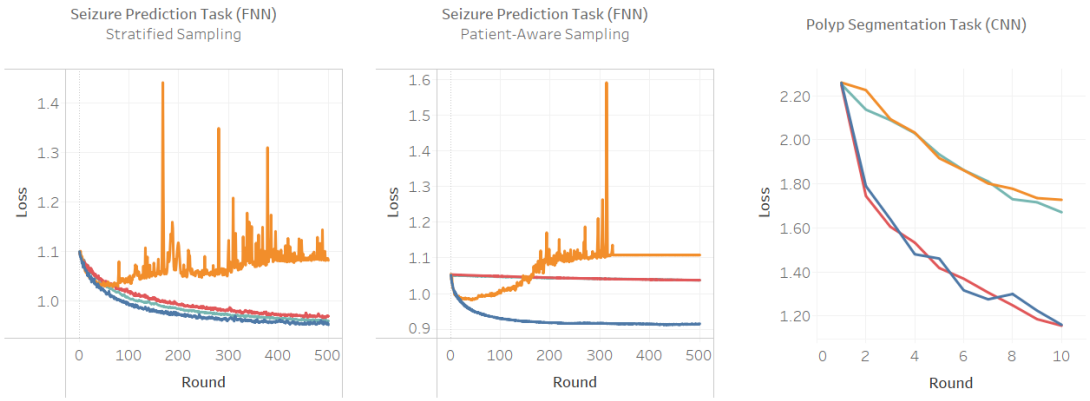


**Figure 4.16:** Comparison of training loss convergence curves for all tasks. FedAvg (dark blue) is compared against the adaptive strategies: FedAdam (orange), FedAdagrad (red), and FedYogi (sky blue). The first two panels display the seizure prediction tasks curves with the stratified and patient-aware datasets respectively, while the third panel displays the polyp segmentation task curves.

### 4.1.7 Execution time

**Seizure prediction task (FNN).** The time that each FL experiment took to run is shown in Figure 4.17. The values displayed in the graphs represent the duration of the federated learning process, which includes the instantiation of the FL clients, the I/O operations performed locally to load the federated datasets, the local training time per client, and the averaging steps of each federated round. The duration doesn't include the time that it takes to start up the federated server. There is a clear increasing trend in execution time as the amount of FL clients is increased. Finally, all algorithms take approximately the same time to run when averaging across client configurations, with FedProx being the exception taking a little bit longer.

**Polyp segmentation task (CNN).** Duration of the federated experiments for the polyp segmentation task is illustrated in Figure 4.18. Similar to the seizure prediction task, there is an increasing trend in execution time proportional to the number of FL clients. However, there is a slight dip in execution time when using 4 FL clients. Analogous to the seizure prediction task the FedProx algorithm takes longer to execute than the other strategies, but the difference is even more pronounced with this dataset.

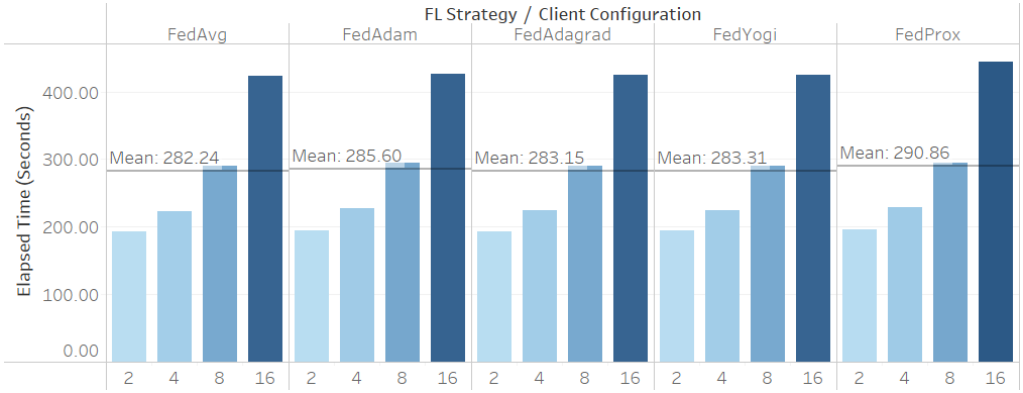


**Figure 4.17:** Execution time (seconds) of each of the models per federated strategy and client configuration, for the seizure prediction task (FNN).

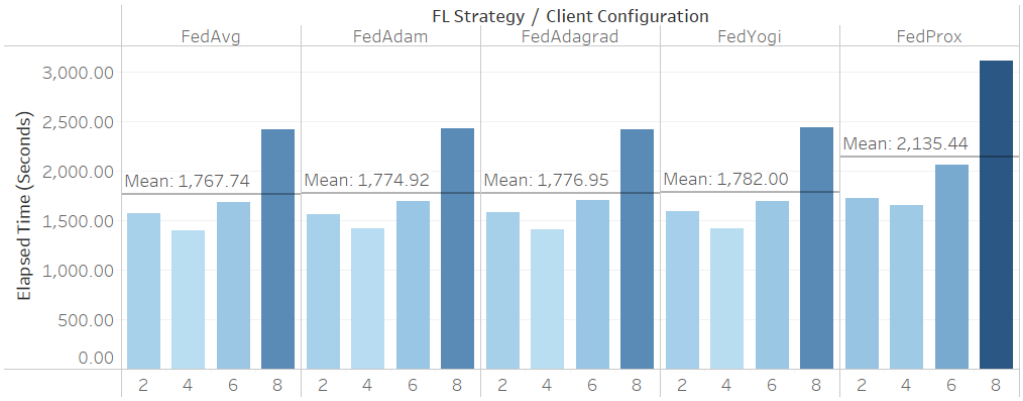


**Figure 4.18:** Execution time (seconds) of each of the models per federated strategy and client configuration, for the polyp segmentation task (CNN).

### 4.1.8 Resources consumption

**Centralized setting.** The amount of VRAM (MB) used by the models for the seizure prediction task (FNN) and polyp segmentation task (CNN) in the centralized setting is displayed in Figure 4.19. VRAM was recorded during each training epoch and averaged to acquire an aggregate of memory consumption for each model. The amount of VRAM consumed by the model is largely dependent on the amount of data that is loaded in memory at a specific training epoch which is defined by the minibatch size. The FNN was fed with a whole batch of data while the CNN was trained with batches of size 6 for each training epoch. The FNN consumed a total of 2.57 GB, while the CNN consumed 7.55 GB due to its larger size of parameters and type of input data (images).

**Federated setting.** The amount of VRAM (MB) used by the models for the seizure predic-

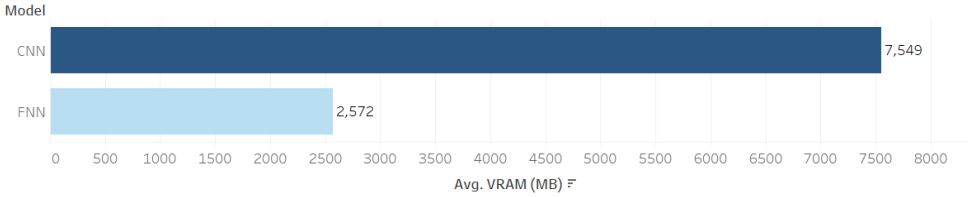


**Figure 4.19:** Centralized setting average VRAM Memory consumption (MB) per training epoch for the networks used.

tion task (FNN) and polyp segmentation task (CNN) in the federated setting is displayed in Figure 4.20. Similar to the centralized setting, VRAM usage was monitored per training unit (FL rounds in this case) and averaged to acquire an aggregate at the level of federated strategy and amount of FL clients. For the FNN, the amount of VRAM consumed increased proportionally to the amount of clients, due to the exposure to whole batches of data per local client at the time. For the CNN, the amount of VRAM varies due to the decreasing batch sizes used on increases to client configuration. Batch sizes were halved per increasing step in the number of FL clients (mini-batches of 16, 8, 4, and 2 items respectively). The amount of memory used across federated strategies remained the same, except for FedProx in the CNN model, which exhibits a slight increase in memory usage due to the overhead of parameter operations required to calculate the proximal term. This increase wasn't present in the FNN due to the smaller amount of parameters required for this network when compared to CaraNet.

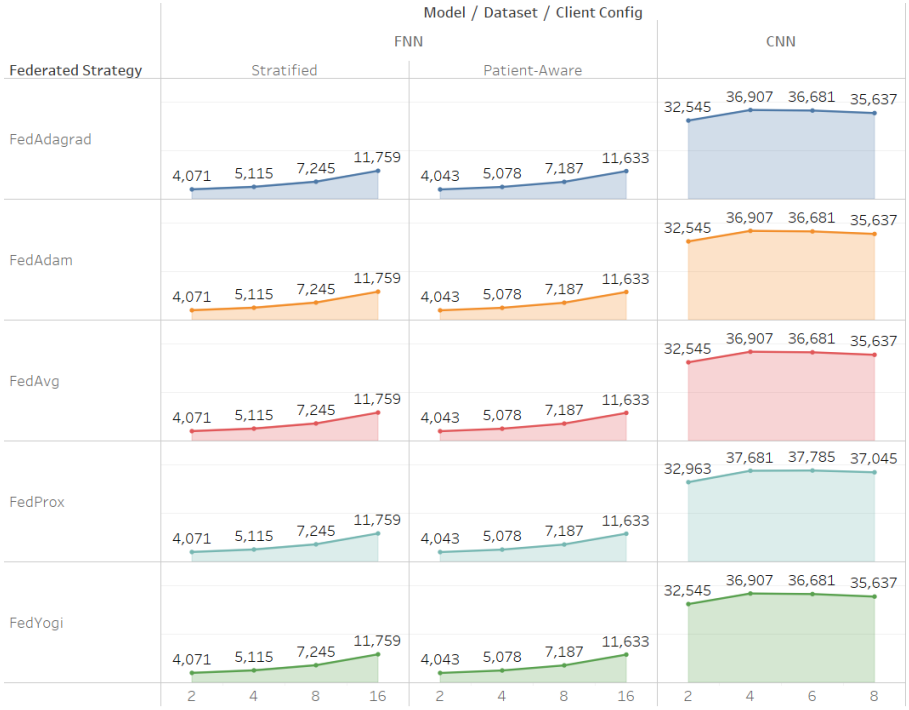


**Figure 4.20:** Federated setting average VRAM Memory consumption (MB) per FL algorithm, client configuration, dataset, and training round for the networks used.

## 4.2 Analysis & interpretation

In this section, the results presented in Section 4.1 are discussed and analyzed in terms of the research questions of this investigation.

**How do federated approaches fare against their centralized counterparts?** *(Q1).* *Centralized setting.* The model trained for the seizure prediction task produced good results for the stratified-sampling approach (AUROC of 0.7 in the test set), and chance level (AUROC of 0.53 on the test set) results for the patient-aware splitting strategy. The good results exhibited in the first case are very likely caused due to cross-contamination of correlated patient examples across evaluation sets. This cross-contamination violates the principle of independent splits in CV, which causes the model to learn spurious relationships in the data and inflates evaluation metrics. This rather frequent methodology mistake motivated the study of a separate dataset with independent patients per split, and the results are less favorable using this approach. The low performance of the model could be caused due to heterogeneous examples of seizures across patient splits. Another possible explanation is that features extracted from the

EEG signal might not be enough to infer patterns in the data, which might be the reason why only a few studies perform seizure prediction with calculated features, and more with RNNs and CNNs using images and raw signals [6]. In any case, both of these results are aligned with the findings explained in Shafiezadeh et al. [8] when using Leave-One-Patient-Out CV strategies. Both approaches were studied despite their low centralized performance, in an attempt to understand if FL can provide any performance benefits. The model trained for the polyp segmentation task achieves good levels of performance (mean dice of 0.75 in the test set) in line with Lou et al. [9] with the default hyperparameters from the original paper used.

*Federated setting.* Federated approaches were capable of producing equivalent results to the centralized experiments in terms of all the evaluation metrics across tasks and datasets as explained in Subsections 4.1.3 and 4.1.4. In some particular cases, the performance of the models even improved like in the patient-aware dataset for the seizure prediction task (from an AUROC of 0.53 to 0.63), and in the polyp segmentation task (from a mean dice score of 0.7 to 0.78). This improvement occurred in specific client configurations (FedAdam with 2 FL clients, and FedProx with 2 FL clients respectively) which might be a result of favorable data distributions occurring in each individual client as a product of random partitioning. However, federating the models also has a regularization effect on the patterns learned in the local nodes which might also lead to more favorable evaluation metrics than the centralized approach when averaged. This indicates there is an opportunity to improve the performance of centralized models trained with heterogeneous datasets by implementing them in a federated setting.

**What is the effect that the number of FL clients has on evaluation metrics?** *(Q2).*
An increase in the amount of FL clients results in a proportional increase in execution time as explained in Subsection 4.1.7. FedProx takes on average more time than the rest of the algorithms due to the overhead of performing matrix operations between the local and global weights for the local loss regularization term. This is more evident in the CaraNet network which has a substantially larger amount of parameters than the FNN trained for the seizure prediction task. The sudden decrease in experiment duration found with 4 FL clients in the polyp segmentation task is an interesting result, and it might be caused due to favorable partitioning for this configuration and batch size. Whether these increases in experiment duration will be found in a distributed environment is a matter that needs to be investigated, as concurrent processes in simulated environments might be affected by context switching and scheduling from the operative system. Furthermore, Beutel et al. [18] recognized that the Virtual Client Engine (VCE) component from the Flower framework might cause configurations with few FL clients to underperform. In their findings, they discovered significant speedups from 35 FL

clients onwards, which suggests experiments with more FL clients should be performed.

Besides the downside of execution time, adding more FL clients resulted in a slight decrease in evaluation performance across tasks in the heterogeneous datasets, in all experiments except for the stratified EEG dataset. This decrease might be caused due to an insufficient number of examples being distributed in larger client configurations for the datasets tested. It is well known that datasets with a big number of examples are required to achieve good results in DL tasks, so the small number of examples present in each client in larger configurations might lead to poor training sets that generate suboptimal local solutions in some cases. When averaged, these noisy solutions might decrease the overall evaluation performance of the aggregated model. This effect is not as noticeable in the stratified approach likely due to the cross-contamination of training examples into the test set. The best models were found in configurations with fewer clients across tasks, which implies there is an optimal range of clients to use for a given data distribution. The amount of FL clients should probably be treated as a hyperparameter of its own, varying the number of clients involved in the learning process.

**Are some FL averaging strategies more robust to non-IID datasets than others?** *(Q3).*
FedAvg performed well on all tasks irrespective of whether the data was IID or non-IID which makes it a robust choice for both scenarios. FedProx which is originally designed for working with heterogeneous datasets also produced good results for the non-IID datasets but might require recalibration of the number of local epochs and FL rounds hyperparameters as validation loss starts increasing in the early stages of training. FedAdam and FedAdagrad had mixed results while FedYogi consistently appeared in the lower rankings of evaluation performance.

**Do adaptive FL averaging strategies improve the convergence performance of the models?** *(Q4).*
FedAvg achieved the lowest average levels of training loss across all tasks and dataset configurations, beating even the adaptive algorithms in most cases. Even though some adaptive strategies featured similar convergence behavior to FedAvg in some scenarios, there is no evidence that the use of adaptive algorithms in the configurations tested in this study resulted in improvements to model convergence. The only conclusion that can be drawn from the experiments is that the behavior of the convergence curves can vary in a case-by-case scenario, so experimentation needs to be performed to understand the applicability of these methods to the task at hand.

Adaptive algorithms didn't produce a benefit in model convergence for the tasks studied probably due to a combination of a) the relatively small number of client configurations tested (maximum 16 FL clients), b) the limited number of examples per federated dataset, and c) the favorable distribution of labels across nodes. Individual client partitions have fewer examples

to train on as the amount of FL clients increases, which makes local models more prone to overfitting when the learning rates are adjusted too aggressively. The benefit of using adaptive algorithms might be more evident in configurations with tens or hundreds of FL clients, each of which might contain bigger datasets and less similar data distributions than the ones studied. Even in the non-IID configurations studied the distribution of classes across nodes was relatively balanced, as seen in Subsection 4.1.1, with the patient-aware EEG dataset splits per client ranging between 14.84% to 28.53% of positive labels, and the polyp dataset splits per client between 9% to 18%. It is possible adaptive algorithms perform better in FL configurations with more FL clients, more data, and bigger differences in data distributions.

**What is the effect of the choice of hyperparameters on model behavior?** *(Q5).*
FedAvg, despite being the model that requires the least tuning of hyperparameters got consistently good results across all tasks, which is in line with its usage as the state-of-the-art algorithm for FL. The FedProx algorithm on the other hand appeared to be relatively robust to the choice of the $\mu$ hyperparameter, as evaluation metrics for different choices of $\mu$ were close to each other as illustrated in Appendix A.2, Figures A.4 and A.8. Both of these algorithms seemed relatively straightforward to tune, the most important parameters to tune are the amount of FL rounds and the number of local training epochs each FL client does.

For the adaptive algorithms, some combinations of the $\tau$, $\eta$, and $\eta_l$ hyperparameters caused the CNN to become numerically unstable, which resulted in the network evaluating to NaNs. In general, the CaraNet model required values of $\tau$ larger than the ones needed in the FNN for the seizure prediction task, as is evident by the large empty rectangular regions in Appendix A.2, Figures A.5, A.6 and A.7. There seems to be a trend toward higher evaluation values in all the algorithms at higher values of $\tau$, $\eta$, and $\eta_l$ for this model, which suggests increasing the range of values explored in the hyperparameter grid would be beneficial. However, discrete jumps on these hyperparameters often caused instabilities which implies a randomized exploration of value ranges would have had more success in finding the sweet spot values for the strategies. The CaraNet network was particularly sensitive to these hyperparameters to find the best models, probably due to a combination of model complexity and the relatively small number of examples used to train the model. The choice of these parameters in the more parsimonious FNN wasn't as important as seen in Appendix A.2, Figures A.1, A.2 and A.3. The choice of parameters for the seizure prediction task didn't generate numerical instabilities, and evaluation metrics were relatively close to each other across hyperparameter choices. The takeaway from this analysis is that networks with more parameters and trained with less data require more care and effort to tune.

# 5

# Conclusions & recommendations

In this work, seizure prediction on EEG signals and polyp segmentation on colorectal images were studied in an effort to gain a deeper understanding of the applicability of FL in healthcare. Through a series of experiments, both tasks were studied in a federated setting and compared against traditional centralized learning approaches. A series of IID and non-IID datasets were considered, and the application of state-of-the-art federated strategies was analyzed to understand their benefits in terms of convergence, adaptivity, and robustness to heterogeneous data. The effect of hyperparameter choice, FL client size, and other configurations were also reviewed and presented in this work. Overall, FL proved to be an effective alternative to CL in the aforementioned tasks, providing similar or even better results than CL in most scenarios. The main conclusions of this work and identified research directions are presented below:

- **Model performance & interpretation.** FL models achieved comparable, and even better results than the centralized models in this work for the medical tasks studied. This suggests that even models that might initially perform poorly in the centralized setting can be improved by simply federating them, leveraging a combination of favorable federated partitioning and regularization side effects. Further investigations are required to assess the degree of improvement that can be achieved in ML models through these side effects of FL. However, due to the private nature of data in real FL, it might be difficult for researchers to understand the underlying distributions of the data and their effect on local model performance. Developing new techniques that allow researchers to delve deeper into these distributions while preserving the privacy of the data is important for the development of the field.

- **Hyperparameter optimization.** As demonstrated in this study, hyperparameter choice has a significant impact on model performance and stability, especially on complex networks that use a larger amount of parameters. Hyperparameter optimization in the federated setting imposes challenges that might not exist in a centralized setting, like a heavier consumption of computational resources, communication costs, understanding of local node behavior, experiment duration, and the energy expenditure of clients in configurations with edge devices. All these challenges might deter FL parties from participating in the tuning process, so better strategies to explore hyperparameter grids in less time while ensuring the statistical significance of the results (through methods like federated CV) are of vital importance to the adoption of FL.

- **Federated strategies.** FedAvg proved to be a robust federated strategy that produces good results most of the time, even when facing heterogeneous datasets, which explains why it is usually depicted as the state-of-the-art algorithm for FL. In the results found in this research both adaptive strategies (FedAdam, FedAdagrad, FedYogi) and strategies designed to deal with heterogeneous data (FedProx) produced only marginal improvements in all the areas, and in some cases even performed worse than the FedAvg standard. The benefits of using these strategies might be more evident when using full-scale federated systems, which suggests further investigations are needed with a more significant number of FL clients and bigger and more diverse datasets.

- **FL in healthcare.** Overall, despite all the challenges, FL has proven to be a feasible alternative to leverage the benefits of ML in data silos. FL allows researchers to train ML models on data that would be otherwise inaccessible, which is increasingly valuable in fields like medicine where publicly available datasets are scarce. Furthermore, FL can motivate the study of rare diseases by naturally augmenting the number of available examples by simply recruiting the parties that hold such data. Further developments need to be performed in the area of incentives, to motivate data owners to contribute to the development of better and more robust ML models that improve the quality of life and survivability rates of patients.

# A

# Supplementary information

## A.1 Hyperparameter grids

- **Seizure prediction task (FNN)**
  - *FedAdam*:
    - $\tau : [10e-3, 10e-2, 10e-1, 10e-0]$
    - $\eta : [10e-3, 10e-2, 10e-1, 10e-0]$
    - $\eta_l : [10e-3, 10e-2, 10e-1, 10e-0]$
  - *FedAdagrad*:
    - $\tau : [10e-3, 10e-2, 10e-1, 10e-0]$
    - $\eta : [10e-3, 10e-2, 10e-1, 10e-0]$
    - $\eta_l : [10e-3, 10e-2, 10e-1, 10e-0]$
  - *FedYogi*:
    - $\tau : [10e-5, 10e-3, 10e-1, 10e-0]$
    - $\eta : [10e-3, 10e-2, 10e-1, 10e-0]$
    - $\eta_l : [10e-3, 10e-2, 10e-1, 10e-0]$
  - *FedProx*:

    * $\mu : [10e-4, 10e-3, 10e-2, 10e-1, 10e-0]$

- **Polyp segmentation task (CNN)**
  - *FedAdam*:
    * $\tau : [10e-2, 10e-1, 10e-0, 15e-0]$
    * $\eta : [10e-2, 10e-1, 3e-0, 5e-0, 10e-0]$
    * $\eta_l : [10e-2, 10e-1, 10e-0]$
  - *FedAdagrad*:
    * $\tau : [10e-2, 10e-1, 10e-0]$
    * $\eta : [10e-2, 10e-1, 10e-0, 12e-0, 15e-0]$
    * $\eta_l : [10e-2, 10e-1, 10e-0]$
  - *FedYogi*:
    * $\tau : [10e-2, 10e-1, 10e-0, 15e-0]$
    * $\eta : [10e-2, 10e-1, 3e-0, 5e-0, 10e-0]$
    * $\eta_l : [10e-2, 10e-1, 10e-0]$
  - *FedProx*:
    * $\mu : [10e-4, 10e-3, 10e-2, 10e-1, 10e-0]$

## A.2 Hyperparameter tuning results

| Dataset | Tau | Eta L | Eta 0.01 | Eta 0.1 | Eta 1 | Eta 10 |
|---|---|---|---|---|---|---|
| PatientAware | 0.0001 | 0.01 | 1.174 | 1.141 | 1.633 | 404.053 |
| | | 0.1 | 1.178 | 1.229 | 3.578 | 683.676 |
| | | 1 | 1.211 | 1.156 | 1.093 (x) | 815.725 |
| | | 10 | 1.202 | 1.162 | 1.321 | 470.572 |
| | 0.01 | 0.01 | 1.171 | 3.323 | 1.115 | 1,920.324 |
| | | 0.1 | 1.159 | 1.178 | 1.090 (x) | 449.866 |
| | | 1 | 1.184 | 1.097 | 1.147 | 1,611.172 |
| | | 10 | 1.207 | 1.267 | 1.212 | 762.524 |
| | 1 | 0.01 | 1.094 | 1.144 | 1.224 | 1.090 |
| | | 0.1 | 1.100 | 1.153 | 1.152 | 1.090 (x) |
| | | 1 | 1.096 | 1.142 | 1.170 | 1.090 |
| | | 10 | 1.098 | 1.139 | 1.230 | 1.090 (x) |
| | 10 | 0.01 | 1.092 | 1.091 | 1.144 | 1.164 |
| | | 0.1 | 1.089 (x) | 1.102 | 1.123 | 1.176 |
| | | 1 | 1.093 | 1.093 | 1.135 | 1.213 |
| | | 10 | 1.090 | 1.094 | 1.121 | 1.224 |
| Stratified | 0.0001 | 0.01 | 1.005 | 1.174 | 1.106 | 1.161 |
| | | 0.1 | 0.998 (x) | 3.423 | 1.438 | 665.304 |
| | | 1 | 1.000 | 8.974 | 2.694 | 3,218.832 |
| | | 10 | 1.001 | 1.570 | 1.278 | 841.368 |
| | 0.01 | 0.01 | 1.001 | 1.010 | 1.110 | 70.292 |
| | | 0.1 | 0.999 | 1.029 | 1.255 | 274.252 |
| | | 1 | 0.999 (x) | 1.005 | 1.132 | 531.808 |
| | | 10 | 1.001 | 1.032 | 1.162 | 762.765 |
| | 1 | 0.01 | 1.076 | 1.024 | 0.999 (x) | 1.106 |
| | | 0.1 | 1.075 | 1.024 | 0.999 | 1.106 |
| | | 1 | 1.076 | 1.025 | 1.006 | 1.106 |
| | | 10 | 1.077 | 1.021 | 1.003 | 1.106 |
| | 10 | 0.01 | 1.102 | 1.076 | 1.024 | 1.002 (x) |
| | | 0.1 | 1.100 | 1.075 | 1.023 | 1.005 |
| | | 1 | 1.100 | 1.074 | 1.022 | 1.004 |
| | | 10 | 1.100 | 1.074 | 1.019 | 1.006 |

**Figure A.1:** Hyperparameter grid results for **FedAdam** in the seizure prediction task (FNN). Evaluation loss over the last 125 rounds was averaged to acquire the values for each cell across a combination of $\tau$, $\eta$, and $\eta_l$ combinations. Each combination was explored against a *stratified* and a *patient-aware* data sampling strategy. Minimum evaluation loss across configurations is marked with an (x).

| Dataset | Tau | Eta L | Eta 0.01 | 0.1 | 1 | 10 |
|---|---|---|---|---|---|---|
| PatientAware | 0.01 | 0.01 | 1.153 | 1.179 | 1.092 | 51.760 |
| | | 0.1 | 1.128 | 1.141 | 1.092 | 28.711 |
| | | 1 | 1.147 | 1.187 | 1.090 (x) | 39.490 |
| | | 10 | 1.154 | 1.160 | 1.094 | 50.512 |
| | 0.1 | 0.01 | 1.119 | 1.203 | 1.090 | 10.803 |
| | | 0.1 | 1.120 | 1.234 | 1.168 | 9.564 |
| | | 1 | 1.127 | 1.188 | 1.090 (x) | 17.168 |
| | | 10 | 1.119 | 1.220 | 1.247 | 28.796 |
| | 1 | 0.01 | 1.093 (x) | 1.132 | 1.207 | 1.613 |
| | | 0.1 | 1.095 | 1.126 | 1.210 | 1.494 |
| | | 1 | 1.097 | 1.138 | 1.183 | 1.151 |
| | | 10 | 1.094 | 1.141 | 1.270 | 3.244 |
| | 10 | 0.01 | 1.089 (x) | 1.091 | 1.133 | 1.208 |
| | | 0.1 | 1.092 | 1.096 | 1.125 | 1.152 |
| | | 1 | 1.094 | 1.099 | 1.133 | 1.171 |
| | | 10 | 1.093 | 1.098 | 1.127 | 1.217 |
| Stratified | 0.01 | 0.01 | 1.032 | 0.998 | 1.108 | 1.117 |
| | | 0.1 | 1.035 | 0.999 | 1.108 | 1.117 |
| | | 1 | 1.030 | 1.003 | 1.107 | 20.724 |
| | | 10 | 1.034 | 0.998 (x) | 1.109 | 32.532 |
| | 0.1 | 0.01 | 1.046 | 1.000 | 1.106 | 29.015 |
| | | 0.1 | 1.047 | 0.997 | 1.106 | 8.856 |
| | | 1 | 1.045 | 0.997 | 1.106 | 65.357 |
| | | 10 | 1.050 | 0.993 (x) | 1.106 | 11.695 |
| | 1 | 0.01 | 1.081 | 1.022 | 1.008 | 1.336 |
| | | 0.1 | 1.080 | 1.023 | 0.997 (x) | 2.170 |
| | | 1 | 1.080 | 1.020 | 0.999 | 3.615 |
| | | 10 | 1.079 | 1.022 | 0.999 | 2.389 |
| | 10 | 0.01 | 1.101 | 1.078 | 1.021 | 0.998 |
| | | 0.1 | 1.100 | 1.075 | 1.019 | 0.999 |
| | | 1 | 1.100 | 1.074 | 1.022 | 0.997 |
| | | 10 | 1.100 | 1.076 | 1.017 | 0.995 (x) |

**Figure A.2:** Hyperparameter grid results for **FedAdagrad** in the seizure prediction task (FNN). Evaluation loss over the last 125 rounds was averaged to acquire the values for each cell across a combination of $\tau$, $\eta$, and $\eta_l$ combinations. Each combination was explored against a *stratified* and a *patient-aware* data sampling strategy. Minimum evaluation loss across configurations is marked with an (x).

| Dataset | Tau | Eta L | Eta 0.01 | 0.1 | 1 | 10 |
|---|---|---|---|---|---|---|
| PatientAware | 0.0001 | 0.01 | 1.174 | 1.141 | 1.633 | 404.053 |
| | | 0.1 | 1.178 | 1.229 | 3.578 | 683.676 |
| | | 1 | 1.211 | 1.156 | 1.093 (x) | 815.725 |
| | | 10 | 1.202 | 1.162 | 1.321 | 470.572 |
| | 0.01 | 0.01 | 1.171 | 3.323 | 1.115 | 1,920.324 |
| | | 0.1 | 1.159 | 1.178 | 1.090 (x) | 449.866 |
| | | 1 | 1.184 | 1.097 | 1.147 | 1,611.172 |
| | | 10 | 1.207 | 1.267 | 1.212 | 762.524 |
| | 1 | 0.01 | 1.094 | 1.144 | 1.224 | 1.090 |
| | | 0.1 | 1.100 | 1.153 | 1.152 | 1.090 (x) |
| | | 1 | 1.096 | 1.142 | 1.170 | 1.090 |
| | | 10 | 1.098 | 1.139 | 1.230 | 1.090 (x) |
| | 10 | 0.01 | 1.092 | 1.091 | 1.144 | 1.164 |
| | | 0.1 | 1.089 (x) | 1.102 | 1.123 | 1.176 |
| | | 1 | 1.093 | 1.093 | 1.135 | 1.213 |
| | | 10 | 1.090 | 1.094 | 1.121 | 1.224 |
| Stratified | 0.0001 | 0.01 | 1.005 | 1.174 | 1.106 | 1.161 |
| | | 0.1 | 0.998 (x) | 3.423 | 1.438 | 665.304 |
| | | 1 | 1.000 | 8.974 | 2.694 | 3,218.832 |
| | | 10 | 1.001 | 1.570 | 1.278 | 841.368 |
| | 0.01 | 0.01 | 1.001 | 1.010 | 1.110 | 70.292 |
| | | 0.1 | 0.999 | 1.029 | 1.255 | 274.252 |
| | | 1 | 0.999 (x) | 1.005 | 1.132 | 531.808 |
| | | 10 | 1.001 | 1.032 | 1.162 | 762.765 |
| | 1 | 0.01 | 1.076 | 1.024 | 0.999 (x) | 1.106 |
| | | 0.1 | 1.075 | 1.024 | 0.999 | 1.106 |
| | | 1 | 1.076 | 1.025 | 1.006 | 1.106 |
| | | 10 | 1.077 | 1.021 | 1.003 | 1.106 |
| | 10 | 0.01 | 1.102 | 1.076 | 1.024 | 1.002 (x) |
| | | 0.1 | 1.100 | 1.075 | 1.023 | 1.005 |
| | | 1 | 1.100 | 1.074 | 1.022 | 1.004 |
| | | 10 | 1.100 | 1.074 | 1.019 | 1.006 |

**Figure A.3:** Hyperparameter grid results for **FedYogi** in the seizure prediction task (FNN). Evaluation loss over the last 125 rounds was averaged to acquire the values for each cell across a combination of $\tau$, $\eta$, and $\eta_l$ combinations. Each combination was explored against a *stratified* and a *patient-aware* data sampling strategy. Minimum evaluation loss across configurations is marked with an (x).

| Dataset | Prox Mu 0.001 | 0.01 | 0.1 | 1 | 10 |
|---|---|---|---|---|---|
| PatientAware | 1.237 | 1.206 (x) | 1.214 | 1.243 | 1.208 |
| Stratified | 0.998 | 1.001 | 1.002 | 1.000 | 0.995 (x) |

**Figure A.4:** Hyperparameter grid results for **FedProx** in the seizure prediction task (FNN). Evaluation loss over the last 125 epochs was averaged to acquire the values for each cell $\mu$ choice. Each choice was explored against a *stratified* and a *patient-aware* data sampling strategy. Minimum evaluation loss across configurations is marked with an (x).

| Tau | Eta L | Eta 0.1 | 1 | 3 | 5 | 10 |
|---|---|---|---|---|---|---|
| 0.1 | 0.1 | | | | | |
| | 1 | | | | | |
| | 10 | | | | | |
| 1 | 0.1 | 0.288 | | | | |
| | 1 | 0.296 (x) | | | | |
| | 10 | 0.278 | | | | |
| 10 | 0.1 | 0.153 | 0.293 | 0.520 | | |
| | 1 | 0.215 | 0.277 | 0.534 (x) | | |
| | 10 | 0.130 | 0.254 | 0.512 | | |
| 15 | 0.1 | 0.204 | 0.257 | 0.330 | 0.508 | |
| | 1 | 0.244 | 0.246 | 0.345 | 0.521 (x) | |
| | 10 | 0.158 | 0.282 | 0.347 | 0.514 | |

**Figure A.5:** Hyperparameter grid results for **FedAdam** in the polyp segmentation task (CNN). Evaluation loss over the last 3 rounds was averaged to acquire the values for each cell across a combination of $\tau$, $\eta$, and $\eta_l$ combinations. Maximum evaluation mean dice coefficient across configurations is marked with an (x).

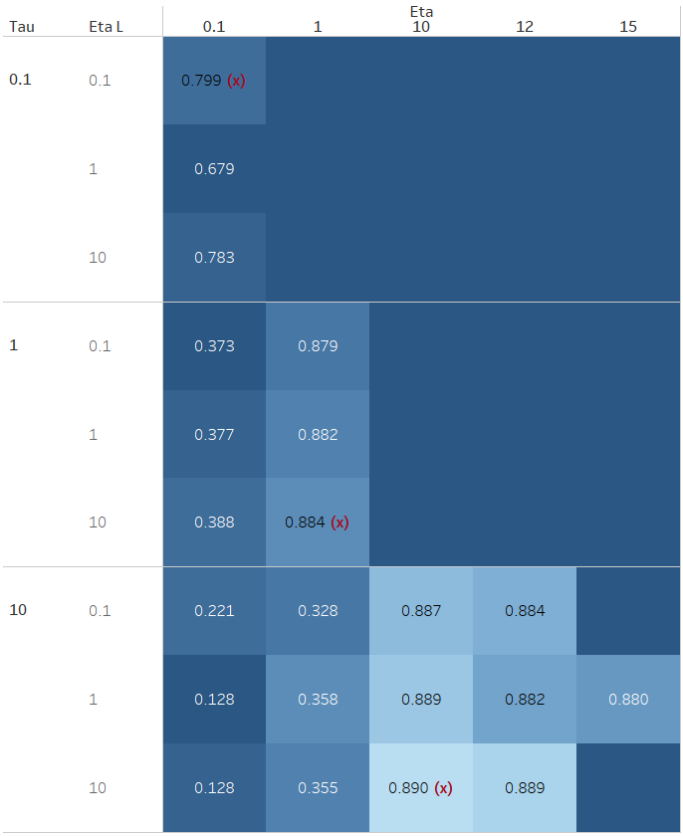


**Figure A.6:** Hyperparameter grid results for **FedAdagrad** in the polyp segmentation task (CNN). Evaluation loss over the last 3 rounds was averaged to acquire the values for each cell across a combination of $\tau$, $\eta$, and $\eta_l$ combinations. Maximum evaluation mean dice coefficient across configurations is marked with an (x).

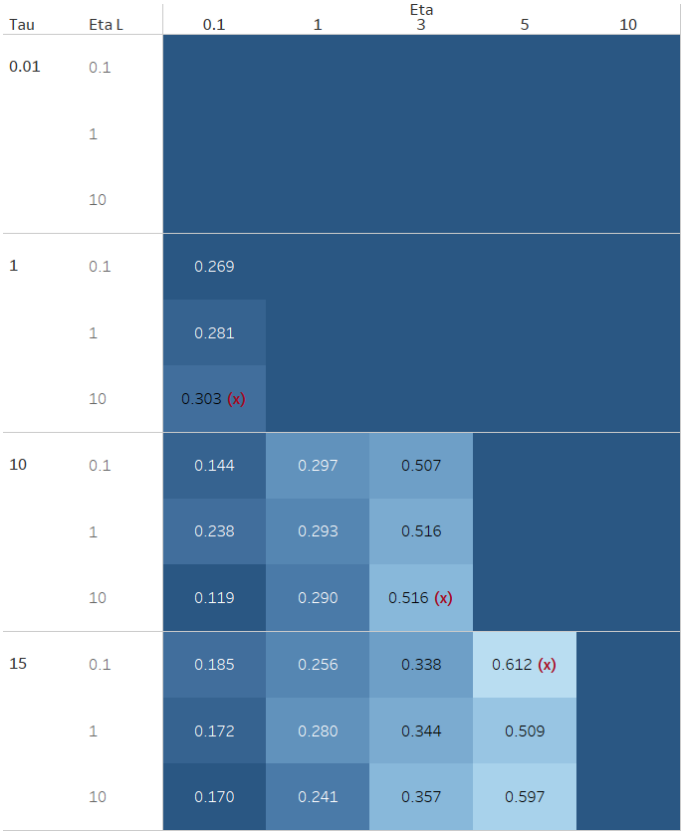


**Figure A.7:** Hyperparameter grid results for **FedYogi** in the polyp segmentation task (CNN). Evaluation loss over the last 3 rounds was averaged to acquire the values for each cell across a combination of $\tau$, $\eta$, and $\eta_l$ combinations. Maximum evaluation mean dice coefficient across configurations is marked with an (x).

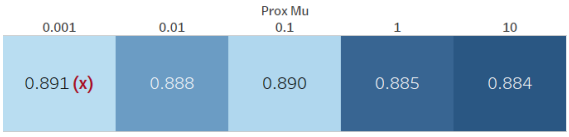


**Figure A.8:** Hyperparameter grid results for **FedProx** in the polyp segmentation task (FNN). Evaluation loss over the last 3 rounds was averaged to acquire the values for each cell $\mu$ choice. Maximum evaluation mean dice coefficient across configurations is marked with an (x).

## A.3 Federated results

| FL Strategy | Client Configuration | Elapsed Time (Seconds) | Round Best Model | Test Mean Dice | Train Loss | Train Mean Dice | Val. Mean Dice |
|---|---|---|---|---|---|---|---|
| FedAvg | 2 | 1,569.33 | 6 | 0.76 | 1.05 | 0.92 | 0.89 |
| | 4 | 1,400.88 | 8 | 0.78 | 1.20 | 0.90 | 0.89 |
| | 6 | 1,682.93 | 9 | 0.76 | 1.29 | 0.89 | 0.88 |
| | 8 | 2,417.83 | 10 | 0.73 | 1.42 | 0.87 | 0.86 |
| FedAdam | 2 | 1,564.22 | 10 | 0.64 | 1.38 | 0.88 | 0.77 |
| | 4 | 1,415.70 | 10 | 0.56 | 1.59 | 0.85 | 0.69 |
| | 6 | 1,695.78 | 10 | 0.46 | 1.83 | 0.81 | 0.61 |
| | 8 | 2,423.96 | 10 | 0.38 | 2.11 | 0.78 | 0.54 |
| FedAdagrad | 2 | 1,580.58 | 10 | 0.76 | 0.91 | 0.93 | 0.90 |
| | 4 | 1,404.60 | 10 | 0.76 | 1.06 | 0.91 | 0.89 |
| | 6 | 1,702.87 | 6 | 0.77 | 1.44 | 0.87 | 0.88 |
| | 8 | 2,419.73 | 9 | 0.70 | 1.43 | 0.87 | 0.86 |
| FedYogi | 2 | 1,590.30 | 10 | 0.00 | 1.33 | 0.88 | 0.00 |
| | 4 | 1,413.94 | 10 | 0.00 | 1.56 | 0.85 | 0.00 |
| | 6 | 1,691.43 | 10 | 0.66 | 1.78 | 0.82 | 0.77 |
| | 8 | 2,432.32 | 10 | 0.61 | 2.01 | 0.79 | 0.73 |
| FedProx | 2 | 1,725.14 | 10 | 0.78 | 0.92 | 0.94 | 0.90 |
| | 4 | 1,653.55 | 10 | 0.77 | 1.09 | 0.92 | 0.90 |
| | 6 | 2,056.75 | 10 | 0.74 | 1.34 | 0.89 | 0.88 |
| | 8 | 3,106.33 | 10 | 0.71 | 1.47 | 0.88 | 0.86 |

**Figure A.9:** Results of the federated models on the polyp segmentation task (CNN). All combinations of FL strategies and client configurations are displayed in this table. Test metrics are accompanied by the training and validation metrics encountered in the FL round that maximized the validation mean dice metric (Round best model column). Dice values for models with instabilities are filled in with 0's (FedYogi with 2 and 4 clients)

| Dataset | FL Strategy | Client Configuration | Elapsed Time (Seconds) | Round Best Model | Test Loss | Test Accuracy | Test Sensitivity | Test Specificity | Test AUROC | Train Loss | Train Accuracy | Train Sensitivity | Train Specificity | Val. Loss | Val. Accuracy | Val. Sensitivity | Val. Specificity |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Stratified Sampling | FedAvg | 2 | 194.51 | 260 | 0.99 | 0.63 | 0.64 | 0.63 | 0.70 | 0.96 | 0.64 | 0.67 | 0.63 | 0.99 | 0.63 | 0.64 | 0.63 |
| | | 4 | 225.72 | 160 | 1.00 | 0.63 | 0.63 | 0.63 | 0.69 | 0.98 | 0.63 | 0.66 | 0.62 | 1.00 | 0.63 | 0.63 | 0.63 |
| | | 8 | 288.84 | 500 | 1.00 | 0.63 | 0.64 | 0.63 | 0.70 | 0.95 | 0.64 | 0.68 | 0.63 | 1.00 | 0.63 | 0.64 | 0.63 |
| | | 16 | 423.90 | 361 | 1.00 | 0.62 | 0.65 | 0.62 | 0.70 | 0.96 | 0.64 | 0.67 | 0.63 | 1.00 | 0.62 | 0.65 | 0.62 |
| | FedAdam | 2 | 196.19 | 125 | 1.03 | 0.69 | 0.45 | 0.75 | 0.65 | 1.03 | 0.68 | 0.47 | 0.73 | 1.03 | 0.69 | 0.45 | 0.75 |
| | | 4 | 229.33 | 159 | 1.01 | 0.61 | 0.64 | 0.60 | 0.67 | 1.00 | 0.63 | 0.62 | 0.63 | 1.01 | 0.61 | 0.64 | 0.60 |
| | | 8 | 292.99 | 68 | 1.03 | 0.64 | 0.55 | 0.66 | 0.66 | 1.02 | 0.63 | 0.59 | 0.64 | 1.03 | 0.64 | 0.55 | 0.66 |
| | | 16 | 430.22 | 30 | 1.04 | 0.66 | 0.48 | 0.70 | 0.64 | 1.04 | 0.65 | 0.52 | 0.68 | 1.04 | 0.66 | 0.48 | 0.70 |
| | FedAdagrad | 2 | 195.84 | 445 | 0.99 | 0.64 | 0.63 | 0.64 | 0.70 | 0.98 | 0.64 | 0.65 | 0.63 | 0.99 | 0.64 | 0.63 | 0.64 |
| | | 4 | 227.53 | 479 | 1.00 | 0.62 | 0.65 | 0.62 | 0.70 | 0.98 | 0.63 | 0.65 | 0.63 | 1.00 | 0.62 | 0.65 | 0.62 |
| | | 8 | 289.14 | 343 | 1.00 | 0.63 | 0.64 | 0.62 | 0.70 | 0.97 | 0.63 | 0.66 | 0.63 | 1.00 | 0.63 | 0.64 | 0.62 |
| | | 16 | 426.75 | 450 | 1.00 | 0.63 | 0.64 | 0.62 | 0.70 | 0.96 | 0.64 | 0.67 | 0.64 | 1.00 | 0.63 | 0.64 | 0.62 |
| | FedYogi | 2 | 195.79 | 448 | 0.99 | 0.63 | 0.64 | 0.63 | 0.70 | 0.97 | 0.65 | 0.64 | 0.65 | 0.99 | 0.63 | 0.64 | 0.63 |
| | | 4 | 226.49 | 441 | 1.00 | 0.63 | 0.65 | 0.62 | 0.70 | 0.96 | 0.63 | 0.68 | 0.62 | 1.00 | 0.63 | 0.65 | 0.62 |
| | | 8 | 288.52 | 491 | 1.00 | 0.64 | 0.63 | 0.64 | 0.70 | 0.96 | 0.65 | 0.66 | 0.64 | 1.00 | 0.64 | 0.63 | 0.64 |
| | | 16 | 427.81 | 325 | 1.00 | 0.63 | 0.64 | 0.62 | 0.69 | 0.96 | 0.64 | 0.67 | 0.63 | 1.00 | 0.63 | 0.64 | 0.62 |
| | FedProx | 2 | 199.00 | 458 | 0.99 | 0.63 | 0.65 | 0.62 | 0.70 | 12.30 | 0.64 | 0.68 | 0.63 | 0.99 | 0.63 | 0.65 | 0.62 |
| | | 4 | 232.57 | 490 | 1.00 | 0.63 | 0.64 | 0.63 | 0.70 | 15.86 | 0.64 | 0.68 | 0.64 | 1.00 | 0.63 | 0.64 | 0.63 |
| | | 8 | 300.83 | 240 | 0.99 | 0.63 | 0.64 | 0.63 | 0.70 | 14.99 | 0.64 | 0.65 | 0.64 | 0.99 | 0.63 | 0.64 | 0.63 |
| | | 16 | 448.84 | 397 | 1.00 | 0.63 | 0.64 | 0.63 | 0.70 | 17.60 | 0.65 | 0.67 | 0.64 | 1.00 | 0.63 | 0.64 | 0.63 |
| Patient-Aware Sampling | FedAvg | 2 | 192.27 | 17 | 1.05 | 0.53 | 0.65 | 0.50 | 0.62 | 1.02 | 0.59 | 0.67 | 0.57 | 1.05 | 0.53 | 0.65 | 0.50 |
| | | 4 | 223.32 | 7 | 1.07 | 0.47 | 0.69 | 0.41 | 0.60 | 1.02 | 0.56 | 0.72 | 0.51 | 1.07 | 0.47 | 0.69 | 0.41 |
| | | 8 | 290.02 | 2 | 1.08 | 0.36 | 0.80 | 0.24 | 0.59 | 1.04 | 0.56 | 0.68 | 0.53 | 1.08 | 0.36 | 0.80 | 0.24 |
| | | 16 | 423.35 | 7 | 1.08 | 0.41 | 0.71 | 0.33 | 0.57 | 0.97 | 0.62 | 0.71 | 0.60 | 1.08 | 0.41 | 0.71 | 0.33 |
| | FedAdam | 2 | 193.96 | 56 | 1.05 | 0.65 | 0.50 | 0.69 | 0.63 | 1.02 | 0.64 | 0.58 | 0.66 | 1.05 | 0.65 | 0.50 | 0.69 |
| | | 4 | 227.47 | 3 | 1.08 | 0.50 | 0.58 | 0.48 | 0.56 | 1.06 | 0.52 | 0.71 | 0.47 | 1.08 | 0.50 | 0.58 | 0.48 |
| | | 8 | 294.32 | 26 | 1.08 | 0.52 | 0.62 | 0.50 | 0.59 | 0.97 | 0.62 | 0.71 | 0.59 | 1.08 | 0.52 | 0.62 | 0.50 |
| | | 16 | 426.68 | 3 | 1.08 | 0.52 | 0.59 | 0.50 | 0.58 | 1.00 | 0.57 | 0.74 | 0.53 | 1.08 | 0.52 | 0.59 | 0.50 |
| | FedAdagrad | 2 | 193.73 | 500 | 1.08 | 0.60 | 0.51 | 0.62 | 0.58 | 1.08 | 0.57 | 0.60 | 0.56 | 1.08 | 0.60 | 0.51 | 0.62 |
| | | 4 | 224.36 | 500 | 1.09 | 0.66 | 0.26 | 0.76 | 0.53 | 1.05 | 0.55 | 0.68 | 0.52 | 1.09 | 0.66 | 0.26 | 0.76 |
| | | 8 | 289.96 | 1 | 1.09 | 0.79 | 0.00 | 1.00 | 0.43 | 1.05 | 0.60 | 0.61 | 0.60 | 1.09 | 0.79 | 0.00 | 1.00 |
| | | 16 | 424.54 | 500 | 1.09 | 0.74 | 0.13 | 0.91 | 0.54 | 0.99 | 0.63 | 0.67 | 0.61 | 1.09 | 0.74 | 0.13 | 0.91 |
| | FedYogi | 2 | 194.29 | 500 | 1.08 | 0.35 | 0.81 | 0.22 | 0.59 | 1.07 | 0.55 | 0.63 | 0.53 | 1.08 | 0.35 | 0.81 | 0.22 |
| | | 4 | 224.54 | 500 | 1.09 | 0.35 | 0.80 | 0.23 | 0.53 | 1.05 | 0.54 | 0.70 | 0.50 | 1.09 | 0.35 | 0.80 | 0.23 |
| | | 8 | 289.70 | 1 | 1.09 | 0.78 | 0.01 | 0.99 | 0.48 | 1.05 | 0.61 | 0.60 | 0.61 | 1.09 | 0.78 | 0.01 | 0.99 |
| | | 16 | 424.69 | 500 | 1.09 | 0.68 | 0.28 | 0.79 | 0.55 | 0.99 | 0.62 | 0.68 | 0.60 | 1.09 | 0.68 | 0.28 | 0.79 |
| | FedProx | 2 | 195.71 | 19 | 1.06 | 0.50 | 0.65 | 0.47 | 0.60 | 1.05 | 0.59 | 0.66 | 0.58 | 1.06 | 0.50 | 0.65 | 0.47 |
| | | 4 | 228.53 | 11 | 1.07 | 0.44 | 0.71 | 0.37 | 0.59 | 1.04 | 0.56 | 0.73 | 0.52 | 1.07 | 0.44 | 0.71 | 0.37 |
| | | 8 | 294.20 | 6 | 1.08 | 0.40 | 0.76 | 0.30 | 0.57 | 1.03 | 0.57 | 0.72 | 0.53 | 1.08 | 0.40 | 0.76 | 0.30 |
| | | 16 | 445.00 | 4 | 1.08 | 0.50 | 0.59 | 0.48 | 0.56 | 1.00 | 0.61 | 0.71 | 0.58 | 1.08 | 0.50 | 0.59 | 0.48 |

**Figure A.10:** Results of the federated models on the seizure prediction task (FNN). All combinations of FL strategies, client configurations, and datasets are displayed in this table. Test metrics are accompanied by the training and validation metrics encountered in the FL round that minimized validation loss (Round best model column).

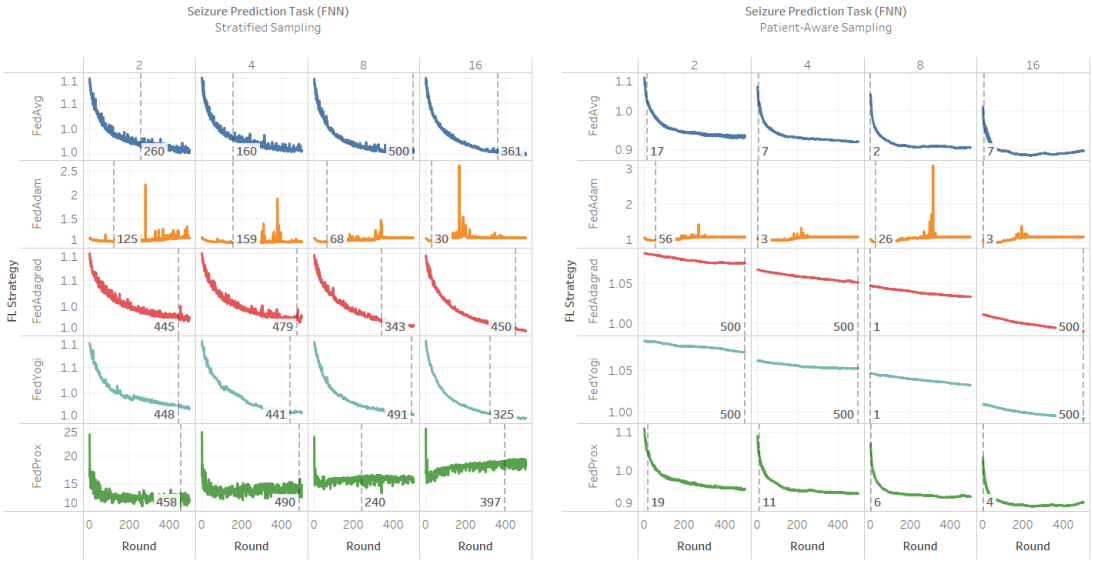


**Figure A.11:** Training loss convergence curves for the federated seizure prediction task (FNN). The left panel contains results for the stratified sampling dataset, while the right panel contains results for the patient-aware dataset. Rows represent each FL strategy employed (with different colors), and columns are divided by the number of FL clients. Each cell represents a model trained with a specific FL strategy, FL client configuration, and dataset. Training loss is plotted per FL round in each cell, and the best round (the one that minimizes evaluation loss) is highlighted using a dotted line. Every cell has an independent scale based on the configuration loss values.

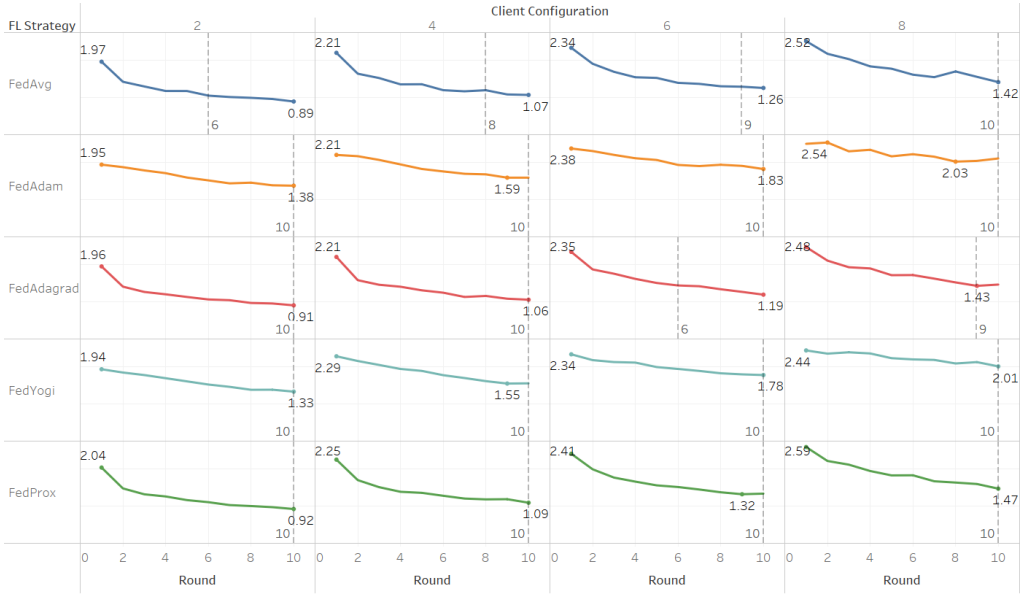


**Figure A.12:** Training loss convergence curves for the federated polyp segmentation task (CNN). Rows represent each FL strategy employed (with different colors), and columns are divided by the number of FL clients. Each cell represents a model trained with a specific FL strategy, FL client configuration, and dataset. Training loss is plotted per FL round in each cell, and the best round (the one that minimizes evaluation loss) is highlighted using a dotted line. Every cell has an independent scale based on the configuration loss values.

## A.4 Software settings

```
python=3.9.16
pytorch-cuda=11.8
torchvision=0.15.2
scikit-learn=1.2.0
flwr=1.4.0
torchmetrics=0.11.4
torchinfo=1.8.0
imbalanced-learn=0.10.1
pillow=9.4.0
py-opencv=4.7.0
```

**Figure A.13:** Conda Environment: List of the main Python packages used and their versions.

## A.5 EEG Extracted features

| Feature Type | API Names |
|---|---|
| Univariate features | compute_mean, compute_variance, compute_std, compute_ptp_amp, compute_skewness, compute_kurtosis, compute_rms, compute_quantile, compute_decorr_time, compute_pow_freq_bands, compute_hjorth_mobility_spect, compute_hjorth_complexity_spect, compute_hjorth_mobility, compute_hjorth_complexity, compute_higuchi_fd, compute_katz_fd, compute_zero_crossings, compute_line_length, compute_spect_slope, compute_spect_entropy, compute_energy_freq_bands, compute_spect_edge_freq, compute_wavelet_coef_energy, compute_teager_kaiser_energy |
| Bivariate features | compute_max_cross_corr, compute_phase_lock_val, compute_nonlin_interdep |

**Figure A.14:** List of the APIs (Python MNE package, version 3.8.5) used to extract the signal features from the EEG dataset by Shafiezadeh et al. [8].

# References


[1] N. Rieke, J. Hancox, W. Li, F. Milletarì, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, S. Ourselin, M. Sheller, R. M. Summers, A. Trask, D. Xu, M. Baust, and M. J. Cardoso, "The future of digital health with federated learning," *npj Digital Medicine*, vol. 3, no. 1, p. 119, 2020.

[2] D. C. Nguyen, Q.-V. Pham, P. N. Pathirana, M. Ding, A. Seneviratne, Z. Lin, O. Dobre, and W.-J. Hwang, "Federated learning for smart healthcare: A survey," *ACM Comput. Surv.*, vol. 55, no. 3, feb 2022. [Online]. Available: https://doi.org/10.1145/3501296

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[4] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *CoRR*, vol. abs/2003.00295, 2020. [Online]. Available: https://arxiv.org/abs/2003.00295

[5] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," *CoRR*, vol. abs/1812.06127, 2018. [Online]. Available: http://arxiv.org/abs/1812.06127

[6] A. Craik, Y. He, and J. L. Contreras-Vidal, "Deep learning for electroencephalogram (eeg) classification tasks: a review," *Journal of Neural Engineering*, vol. 16, no. 3, p. 031001, apr 2019. [Online]. Available: https://dx.doi.org/10.1088/1741-2552/ab0ab5

[7] L. F. Sánchez-Peralta, L. Bote-Curiel, A. Picón, F. M. Sánchez-Margallo, and J. B. Pagador, "Deep learning to find colorectal polyps in colonoscopy: A systematic litera-

ture review," *Artificial Intelligence in Medicine*, vol. 108, p. 101923, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0933365719307493

[8] S. Shafiezadeh, G. M. Duma, G. Mento, A. Danieli, L. Antoniazzi, F. Del Popolo Cristaldi, P. Bonanni, and A. Testolin, "Methodological issues in evaluating machine learning models for eeg seizure prediction: Good cross-validation accuracy does not guarantee generalization to new patients," *Applied Sciences*, vol. 13, no. 7, 2023.

[9] A. Lou, S. Guan, H. Ko, and M. Loew, "Caranet: Context axial reverse attention network for segmentation of small medical objects," 2022.

[10] Q. Li, W. Z. Wen, Zeyi and, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, pp. 3347–3366, 2021.

[11] D. Jha, P. H. Smedsrud, M. A. Riegler, P. Halvorsen, T. de Lange, D. Johansen, and H. D. Johansen, "Kvasir-seg: A segmented polyp dataset," in *MultiMedia Modeling*, Y. M. Ro, W.-H. Cheng, J. Kim, W.-T. Chu, P. Cui, J.-W. Choi, M.-C. Hu, and W. De Neve, Eds. Cham: Springer International Publishing, 2020, pp. 451–462.

[12] S. Banabilah, M. Aloqaily, E. Alsayed, N. Malik, and Y. Jararweh, "Federated learning review: Fundamentals, enabling technologies, and future applications," *Information Processing and Management*, vol. 59, no. 6, p. 103061, 2022.

[13] A. Rauniyar, D. H. Hagos, D. Jha, J. E. Håkegård, U. Bagci, D. B. Rawat, and V. Vlassov, "Federated learning for medical applications: A taxonomy, current trends, challenges, and future research directions," 2022.

[14] F. AI, "Fate," Federated AI, 2023. [Online]. Available: https://github.com/FederatedAI/FATE

[15] TensorFlow, "Tensorflow federated," TensorFlow, 2023. [Online]. Available: https://www.tensorflow.org/federated

[16] OpenMined, "Pysyft," OpenMined, 2023. [Online]. Available: https://github.com/OpenMined/PySyft

[17] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, X. Zhu, J. Wang, L. Shen, P. Zhao, Y. Kang, Y. Liu, R. Raskar, Q. Yang, M. Annavaram, and S. Avestimehr, "Fedml: A research library and benchmark for federated machine learning," 2020.

[18] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. Hei Li, T. Parcollet, P. Porto Buarque de Gusmão, and N. D. Lane, "Flower: A friendly federated learning research framework," 2022.

[19] H. R. Roth, Y. Cheng, Y. Wen, I. Yang, Z. Xu, Y.-T. Hsieh, K. Kersten, A. Harouni, C. Zhao, K. Lu, Z. Zhang, W. Li, A. Myronenko, D. Yang, S. Yang, N. Rieke, A. Quraini, C. Chen, D. Xu, N. Ma, P. Dogra, M. Flores, and A. Feng, "Nvidia flare: Federated learning from simulation to real-world," 2023.

[20] P. Foley, M. J. Sheller, B. Edwards, S. Pati, W. Riviera, M. Sharma, P. N. Moorthy, S. han Wang, J. Martin, P. Mirhaji, P. Shah, and S. Bakas, "OpenFL: the open federated learning library," *Physics in Medicine &amp Biology*, vol. 67, no. 21, p. 214001, oct 2022. [Online]. Available: https://doi.org/10.1088%2F1361-6560%2Fac97d9

[21] H. Ludwig, N. Baracaldo, G. Thomas, Y. Zhou, A. Anwar, S. Rajamoni, Y. Ong, J. Radhakrishnan, A. Verma, M. Sinn, M. Purcell, A. Rawat, T. Minh, N. Holohan, S. Chakraborty, S. Whitherspoon, D. Steuer, L. Wynter, H. Hassan, S. Laguna, M. Yurochkin, M. Agarwal, E. Chuba, and A. Abay, "Ibm federated learning: an enterprise framework white paper v0.1," 2020.

[22] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. van der Laak, B. van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, pp. 60–88, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361841517301135

[23] K. ELKarazle, V. Raman, P. Then, and C. Chua, "Detection of colorectal polyps from colonoscopy using machine learning: A survey on modern techniques," *Sensors*, vol. 23, no. 3, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/3/1225

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

# Acknowledgments

My sincerest thanks to my supervisors, Prof. Alessandro Sperduti and Prof. Alberto Testolin from the Department of Mathematics at the University of Padova, for their valuable guidance in this academic endeavor, and for facilitating the cloud infrastructure and tools required to perform this study. Last but not least, my heartfelt gratitude to the Erasmus Mundus BDMA program staff for granting me the honor of participating in this life-changing international master's program.