

D9.1 Report on algorithms & strategies for federated training, distillation, and fine-tuning of foundational models

Lead Author: Alexandru Serban, Cosmin Hatfaludi

Reviewer: Cecilio Angulo [UPC], Jan Ramon and Mouad Blej [INRIA]

Deliverable nature	Report (R)
Dissemination level	Public Use
Delivery date	31-10-2024
Version	2.1
Total number of pages	42 pages
Keywords	Foundational models, Federated Learning, Pre-training, Healthcare applications, self-supervised learning, pretrained models, fine tuning, distillation, transfer learning

EXECUTIVE SUMMARY

This report presents a comprehensive literature survey exploring the intersection of foundational models (FMs) and Federated learning (FL), detailing and characterising existing methods for training, customizing, and deploying FMs using FL. It provides extensive comparisons between the methods, highlighting their advantages and disadvantages, and offers practical perspectives on adopting and developing the methods presented by comparing their complexity, efficiency, and scalability. Moreover, the survey explores the application of FMs using FL in the healthcare domain, an area currently underserved by existing literature, thereby highlighting an opportunity for future developments and impact for FLUTE.

DOCUMENT INFORMATION

Grant agreement No.	101095382	Acronym	FLUTE
Full title	Federate Learning and mUlti-party computation Techniques for prostatE cancer		
Call	HORIZON-HLTH-2022-IND-13-02		
Project URL	https://cordis.europa.eu/project/id/101095382		
EU project officer	Mrs. Serena Battaglia		

Deliverable	Number	D9.1	Title	Report on algorithms and strategies for Federated Learning
Work package	Number	WP9	Title	Scalable federated learning using pre-trained models
Task	Number	T9.1	Title	Solution developer API validation and identification of algorithms for federated distillation and fine-tuning

Date of delivery	Contractual	M18	Actual	31 October 2024
Status	version 2.1	<input checked="" type="checkbox"/> Final version		
Nature	<input checked="" type="checkbox"/> R <input type="checkbox"/> DEM <input type="checkbox"/> DMP <input type="checkbox"/> DEC <input type="checkbox"/> ETHICS <input type="checkbox"/> OTHER			
Dissemination level	<input checked="" type="checkbox"/> Public <input type="checkbox"/> Sensitive			

Authors (partners)	SR
Responsible author	Alexandru Serban alexandru.serban@siemens.com

Summary (for dissemination)	<p>This report presents a comprehensive literature survey exploring the intersection of foundational models (FMs) and Federated learning (FL), detailing and characterising existing methods for training, customizing, and deploying FMs using FL. It provides extensive comparisons between the methods, highlighting their advantages and disadvantages, and offers practical perspectives on adopting and developing the methods</p>
-----------------------------	--

presented by comparing their complexity, efficiency, and scalability.

VERSION LOG			
Issue Date	Rev. No.	Author	Change
01/10/2024	v.0.1	Alex Serban, Cosmin Hatfaludi	First draft including the classification and description of all methods
15/10/24	v 1.0	Alex Serban, Cosmin Hatfaludi	First internally reviewed version
21/10/24	v 1.1	Alex Serban, Cosmin Hatfaludi	Final version after suggested changes
25/10/24	v 2.0	Alex Serban, Cosmin Hatfaludi	Final version after second review
30/10/30	v 2.1	Lisa Hanselmann	Final version after formatting and final proofread

TABLE OF CONTENTS

LIST OF FIGURES	7
LIST OF TABLES	7
1. Introduction	8
2. Background	8
2.1. Methodology	8
2.2. Taxonomy	9
3. Methods	11
3.1. Train	11
3.2. Customize	15
3.3. Deploy	23
4. FMs and FL in healthcare applications	23
5. Practical perspectives	24
6. Insights for the FLUTE project	26
7. Conclusions	27
APPENDIX A - PREREQUISITES	29
A.1 Federated Learning	29
A.2 Pre-training Foundational Models	29
A.3 Fine-tuning FMs and LoRA	30
A.4 Knowledge distillation	31
APPENDIX B – LIST OF TERMS FOR LITERATURE SEARCH	33
REFERENCES	34

LIST OF FIGURES

Figure 1 - Selection of articles using the filtering criteria.....	9
Figure 2 - Taxonomy of methods for training or using foundational models using federating learning.....	10
Figure 3 - Overview of pre-training methods for FMs using FL.....	11
Figure 4 - Overview of additive fine-tuning methods, where the grey boxes mean that the added parameters can be heterogeneous between clients.....	16
Figure 5 - Overview of KD methods.....	20
Figure 6 - Illustration of pre-training FMs using contrastive learning.....	30
Figure 7 - Illustration of LoRA.....	31
Figure 8 - Illustration of knowledge distillation.....	31

LIST OF TABLES

Table 1 - Classification of methods based on their complexity, efficiency, and scalability.....	25
Table 2 - List of terms for the queries. Queries were formed by first combining the first and second terms using the boolean AND operator, and afterwards combining the first, second, and third terms using the same operator.	33

1. Introduction

Federated Learning (FL) allows multiple parties to collaboratively train a machine learning (ML) model without exchanging or transferring private data [89]. Its main goal is to improve ML algorithms by integrating siloed data, which would otherwise remain isolated and underused. For example, using patient data from different hospitals to improve diagnostic ML algorithms would be challenging without FL [59].

As ML models and datasets grow, particularly with the development of Foundational Models (FM) [7], training them requires more computational resources and diverse data sources. However, high quality data is often siloed due to privacy concerns. In the context of the FLUTE project, WP9 explores how FL can enable new data sources for training FMs, improve resource sharing, or use pre-trained FMs, to develop robust collaborative ML models. While both FMs and FL aim to improve ML models, their approaches are orthogonal. FMs are trained using large datasets integrating diverse sources and fine-tuned for various tasks [2], [11], [87]. In contrast, FL allows multiple parties with similar data to train specific models without centralizing or sharing private data, thereby preserving privacy.

The large size of FMs and the overhead of FL present challenges such as managing bandwidth, minimizing training latency, and ensuring efficient, private data transmission [93]. Furthermore, there are higher-level concerns regarding the trade-offs between privacy and performance, fairness among participants, and unclear ownership models for integrating FMs in FL [98]. This integration requires addressing technical, legal, and organizational challenges, such as model compression and quantization to reduce overhead [63] and incentivization structures to define benefits and ownership for participating nodes [73].

In this report, we present a comprehensive literature review focused on identifying, categorizing, and describing technical methods that integrate FMs and FL. The document is organised as follows. We present background information about the methodology used to discover the algorithms presented and their classification using a common taxonomy (SECTION 2) followed by a comparison of the algorithms (SECTION 3), a discussion about the use of FMs and FL in healthcare (SECTION 4), and a discussion about practical perspectives on adopting and developing novel methods (SECTION 5). Insights and recommendations for the FLUTE project are introduced afterwards (SECTION 6), followed by conclusions (SECTION 7). We also discuss prerequisite information about FL, FMs, fine-tuning and knowledge distillation (KD) in APPENDIX A.

2. Background

This section outlines the methodology used for the literature survey and presents the taxonomy used to classify the reviewed methods for integrating FMs and FL.

2.1. Methodology

To identify potentially all methods at the intersection of FL and FMs, we performed a systematic literature review [64], aimed at identifying, classifying, and evaluating the strengths and weaknesses of various methods. The study protocol was designed following the guidelines of Okoli[56] and Yannascoli et al.[85].

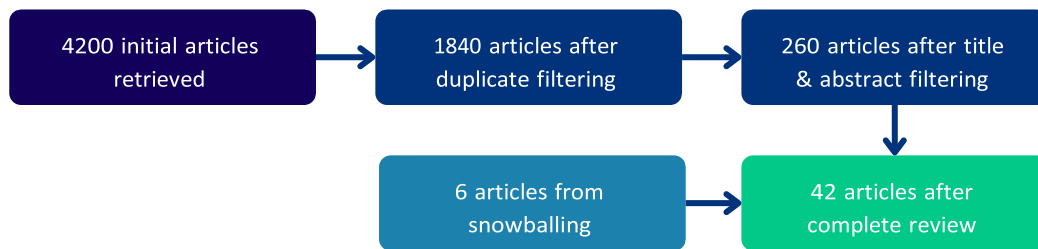


Figure 1 - Selection of articles using the filtering criteria.

To ensure comprehensive coverage of academic and potential non-academic studies, the search strategy included multiple information sources. Initially, we used the Google and Google Scholar search engines. While these engines are known to encompass many other information sources commonly used in literature reviews [65], we also used the ScienceDirect and Scopus search engines to further broaden the search. For each information source we used the first five pages of answers. ScienceDirect and Scopus were queried using the APIs, limiting the number of answers to fifty for each query.

To define the queries, we followed several guidelines [56], [85] and initially defined terms that comprised of two elements. For the first element, we used the term *federated learning*, and for the second element we used the term *foundational model* along with various synonyms for the techniques used to develop or use FMs. These included *self-supervised learning*, *pretrained models*, *fine tuning*, *distillation*, *transfer learning*. Additionally, we created two variations of these queries to make them more specific. First, we added an element suggesting the medical domain using the words *medical* and *healthcare*. Second, we added an element suggesting the practical application of these methods using the words *application*, *implementation*, *development*, and *deployment*. A complete list of keywords is provided in APPENDIX B.

To filter out irrelevant documents, we first removed duplicates based on the article titles, and restricted the search to articles published from 2021 onwards, as this year corresponds with the first developments in FMs. Afterwards, we reviewed the titles and abstracts of the remaining articles and discarded those that did not align with our goals. The remaining articles were fully read and the initial list of articles was complemented from their references using a snowballing strategy [33]. An illustration of the article filtering process is provided in FIGURE 1.

2.2. Taxonomy

When attempting to classify the articles discussed in this report using taxonomies presented in the literature, we found that no existing work could categorize them into distinct, nonoverlapping classes based on the algorithms presented. For example, the taxonomy proposed by [45], while compelling, exhibited significant overlap between classes and provided only representative examples for each technique. Upon trying to scale up their taxonomy, we found that the fine granularity of their approach made it difficult to fit some methods from our study. Therefore, to classify the methods, we developed a broader taxonomy that is compatible with all proposals in the literature but adheres to more stringent classification criteria.

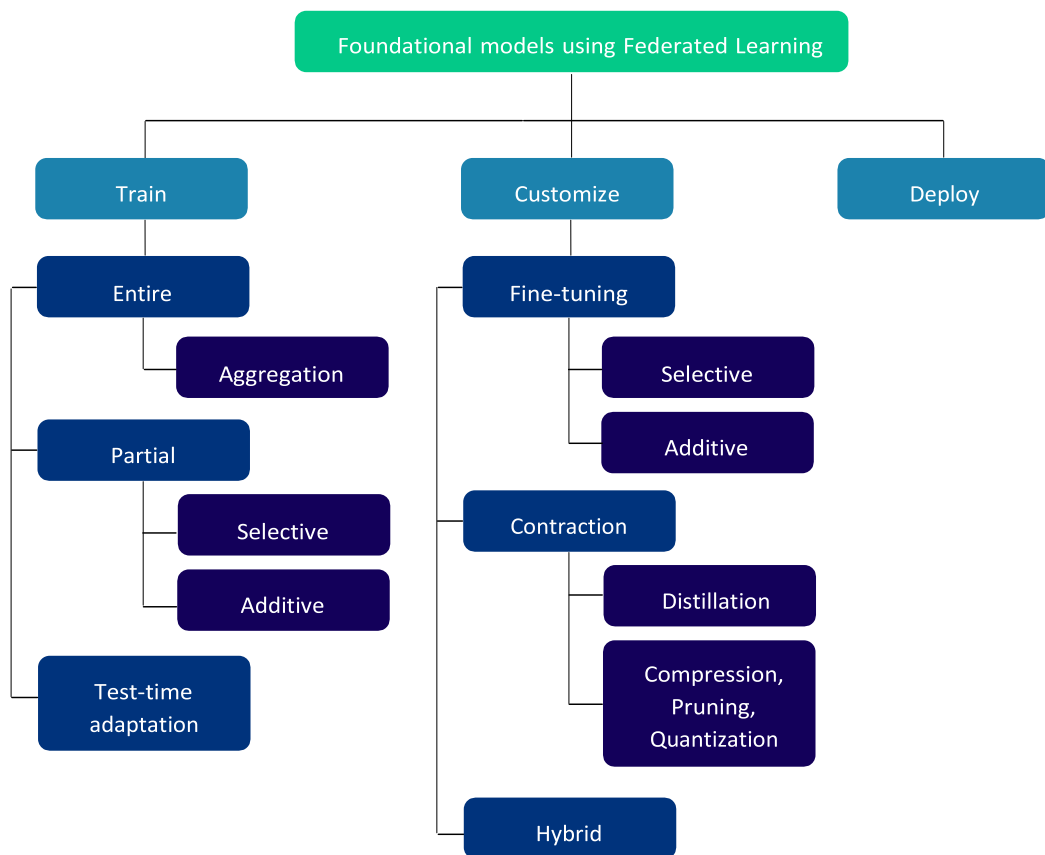


Figure 2 - Taxonomy of methods for training or using foundational models using federating learning.

Initially, we classified the articles based on the stage of the development life-cycle into three categories: (i) *train* – focusing on techniques for pre-training FMs using FL, (ii) *customize* – focusing on the adaptation of FMs for specific tasks using FL, and (iii) *deploy* – focusing on the use of FL to run inference for a pre-trained or customised FM. Within each of these categories, we further defined sub-classes based on the core algorithmic technique used to develop the methods. An illustration of this taxonomy is provided in [FIGURE 2](#). We observe that for certain classes, such as

those used to customize FMs, it was possible to define more fine-grained sub-classes due to the diversity of the algorithms used. Conversely, for other classes like the deploy class, where fewer techniques have been developed, the taxonomy is coarser. Comprehensive details about the class definitions are discussed in SECTION 3, where the methods belonging to each class are introduced.

3. Methods

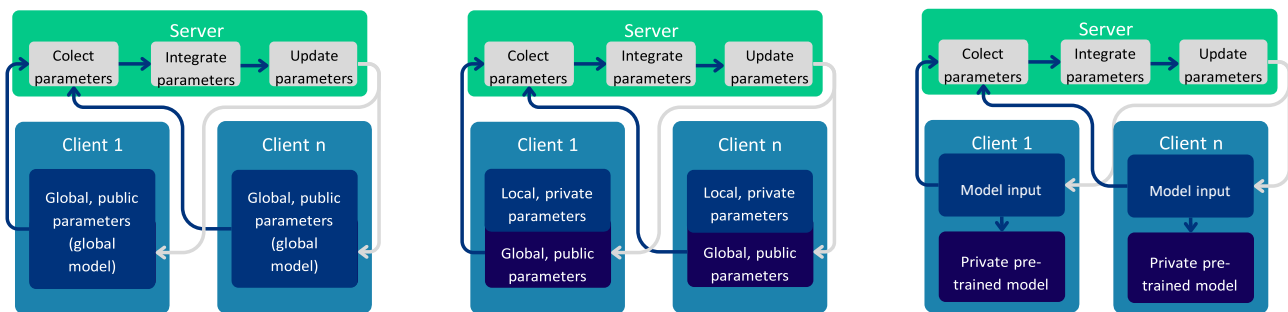


Figure 3 - Overview of pre-training methods for FMs using FL

3.1. Train

Although collaboratively training FMs using FL can offer access to high-quality siloed data, it is also the most challenging use-case as communicating large model updates leads to substantial communication overhead. Managing large datasets and models requires innovative communication-efficient algorithms and compression methods [52]. Additionally, handling heterogeneous data distributions and varied computational resources across nodes poses further challenges [37]. Due to these challenges, few publications attempt to pre-train entire FMs, with most focusing on adding parameters or pre-training only a subset of the FMs collaboratively. This technique is also known as continuous domain adaptation, where off-the-shelf FMs are further pre-trained on more specific data [35].

To initially categorize the algorithms in this class, we first consider the parts of the model used for pre-training: (i) *entire* – where the whole model is pre-trained using FL, (ii) *partial* – where only parts of the model are pre-trained using FL, and (iii) *test time adaptation* – where the pre-training phase adjusts specific parameters or inputs for customization or deployment using FL – such as modifying the prompts used to better suit the context or specific user requirements. An illustration of these methods is provided in FIGURE 3, focusing on the parts of the models that are trained collaboratively. On the server side, most algorithms discussed in this section use Federated Averaging (FedAVG) (APPENDIX A.1) or its variants for parameter aggregation, but implement novel techniques to minimize the parameters transmitted over the network. Within these classes, several sub-categories could be identified, detailed in the following sections.

3.1.1. Entire model pre-training

The methods presented in this section can be classified into methods that aim to pre-train FMs using FL, discussed below, and those that improve the aggregation algorithms used on the server, detailed in the *aggregation* paragraph. This distinction was made because novel aggregation methods can, in principle, be combined with other pre-training techniques.

Bernal et al.[6] conducted one of the first empirical studies on pre-training a FM in FL, training a Word2Vec [53] model collaboratively with a small number of clients holding large text corpora. They used FederatedSGD [52], where the only changes were context-specific decisions like merging client vocabularies. While the final model did not improve performance significantly, the study showed the feasibility of training FMs using FL. Sani et al.[63] further optimized training by combining efficient local and global gradient updates with data parallelism techniques typically used for large-scale FMs (e.g., distributed data parallelism). Besides these optimizations, the authors used a larger set of local updates before updating the global model, which reduced communication costs and proved effective when scaling the dataset size.

While these articles focused on using existing techniques without introducing novel methods, they demonstrate the viability of pre-training FMs in FL settings. Zhuang et al.[96] first adapted pre-training techniques for FMs by modifying Bootstrap Your Own Latent (BYOL) (APPENDIX A.2) to run in FL. In their approach, each client trains a BYOL-like model locally using contrastive learning, with one encoder updated via gradient descent and the other via Exponential Moving Average (EMA). Every n epochs, clients upload their EMA encoders to the server, which merges them using FedAVG and sends the updates back to the clients. The clients can then dynamically choose to replace their EMA models with the global one while maintaining consistency with their local data. The novelty of this approach is that only the EMA encoder is updated globally, reducing communication overhead. Li et al.[44] also introduced a method using contrastive learning in FL, allowing clients to adapt their local models on demand. Unlike Zhuang et al., they performed contrastive learning at the model level, aiming to minimize the distance between local and global model features by contrasting them at each training step. The global model is updated using FedAVG every epoch, and after the update the contrastive loss is used to update the local model. This removes the need for an EMA encoder, as the global model acts like it. Nevertheless, the communication costs increase as the global model is updated at every epoch.

Zhuang et al.[97] examined the advantages and disadvantages of choosing different pretraining techniques, and evaluated their impact of running them in FL. They conducted an empirical study comparing SimCLR, MoCo, BYOL, and SimSiam (APPENDIX A.2) pre-training methods in FL. The authors found that non-contrastive methods such as BYOL perform better, and while EMA is not essential, it improves performance. To further investigate EMA's influence, they replaced the update of the local EMA encoder with the global model, as performed in [96], with an EMA update on the global model where the decay rate is dynamically set by all clients. This technique led to performance improvements, as the EMA model is more adapted to local data.

Makhija et al.[50] presented a method where each client first pre-trains an individual model, and then a global model is adapted using a separate alignment dataset. At each epoch, the server sends a subset of this dataset and the corresponding global model embeddings to all clients. Clients then compute the similarity between their local embeddings and the global embeddings, and send their local embeddings back to the server. The server aggregates these representations and sends the results back to the clients, which continues training by maximizing the embeddings similarity. While this approach outperforms previously discussed methods such as Zhuang et al.[97], it incurs high communication costs due to the exchange of data and embeddings. Some of these costs are alleviated by using a smaller alignment dataset. However, the need for a small dataset is not guaranteed by the method. Agarwal et al.[1] run an empirical study on pre-training FMs using FL, investigating distinct factors such as data heterogeneity, the client’s contribution to the final metrics, as well as the influence of the client’s dataset size in relation to performance and speed. They found that using FMs can have both a positive and a negative impact on the local models, depending on how skewed the data distribution is between clients. The authors also found that having more clients with less data, instead of less clients with more data, can impact the performance of the FMs, and note that performing less updates from clients with less data can improve the communication efficiency without loss of performance.

Aggregation To improve FedAVG and address potential scaling issues with the embedding norms during local updates, Kim et al.[39] introduced an additional L2 feature normalization, moderated by a client-specific scaling factor. This technique demonstrates improved performance when applied to both pre-training and fine-tuning, and could potentially be integrated with the methods discussed earlier.

Rehman et al.[62] introduced a new aggregation method aimed at reducing bias from heterogeneous clients during model aggregation. In their approach, each client performs local pretraining and uploads its models to the server. The server then computes a layer-wise weight for each model’s contribution to the final global model, using a weight metric measured as the cosine similarity between the client’s model and the global model from the previous iteration. The method can be interpreted as a weighted, layer-wise FedAVG, which improves both the performance and mitigates some of the biases introduced by clients with more skewed data distributions.

Recasens et al.[61] proposed a novel averaging technique based on the Fisher matrix, which merges local models in the parameter space. The method assumes that each client performs pre-training locally on all data, until convergence, and only upon convergence merges the models into a global model, significantly reducing the communication costs during training. The Fisher matrix is used in the aggregation process to weigh the importance of each parameter, providing information for preserving parameters that contribute most significantly to the model’s performance. This method is particularly effective in scenarios where client data distributions are highly heterogeneous, ensuring that the global model benefits from the most informative local updates.

3.1.2. Partial model training

The methods discussed in this section pre-train only a portion of the FMs using FL and can be classified based on the parameters used for pre-training. These include methods that add new parameters for pre-training (additive) and methods that select existing parameters for pre-training (selective).

Additive [Tan et al.\[68\]](#) assume that the clients already have independent pre-trained models, obtained either from pre-training on their own data or adopting off-the-shelf FMs, and use these to pre-train a global FM. In this setting, each client encodes their data with the local model and shares the embeddings with all other clients. The clients then concatenate all embeddings and project them through a learnable projection layer to obtain client-specific prototypes for the concatenated embeddings. These prototypes are shared with the server, which aggregates them using FedAVG, and sends the results back to the clients. For supervised or semi-supervised tasks, multiple class-wise prototypes can be used. After receiving the prototypes from the server, each client minimizes the distance between the local and the global prototype, or can perform contrastive learning if multiple prototypes are used. Sharing and training only the prototypes significantly reduces communication costs while enabling each client to contribute more effectively to the global model. This method improves both performance and communication efficiency. However, it assumes the existence of pre-trained models for all clients and does not explore the scenario where all clients use the same FM (e.g., using an off-the-shelf FM). [Fani et al. \[21\]](#) extended this method by introducing a regularization term based on the magnitude of the prototype weights to address concerns arising from heterogeneous data distributions across clients. This addition resulted in further performance improvements.

[Chen et al. \[14\]](#) further used the prototype mechanism to achieve a balance between pretrained models that provide generic features and more personalized, client-specific features. The adaptation consists in increasing the training period for clients with more complex data, and simplifying the local parameters and the training period for clients with limited or noisy data. This dynamic adjustment helps optimize the training process for each client. Their experiments demonstrated improved performance in pre-training methods for personalizing the final models, while being more resilient to bias added by specific clients.

[Kim et al. \[38\]](#) further simplified this method demonstrating that removing the embedding concatenation and projecting only the features from the local models is also effective. The clients send only the projection layers to the server, which aggregates them using FedAVG. However, the authors used projection layers with many more parameters, making them more similar to adapter layers [\[58\]](#) rather than simple projection layers.

[Lu et al. \[49\]](#) further extended this strategy by also incorporating learnable adapters for the inputs, in addition to the projections (prototypes) used for the outputs of a FM. The authors trained both an auto-encoder to map the inputs to a FM (a stage also called input surgery), and use the outputs of the FM to project them to a common space through a learnable layer. The server receives both the input auto-encoder and the output projection, performs FedAVG, and returns the aggregated results to the clients, who then replace their corresponding local components with the global ones. This approach improves the adaptability of the FM by integrating both input and output

adjustments. However, it also increases communication costs compared with previously mentioned methods, as the input auto-encoder is additionally trained using FL.

Selective Lit et al. [46] introduced a method to pre-train the BERT language model in FL by splitting the model into two parts: one trained locally by each client and one trained jointly by all clients. The novelty lies in splitting the model to have more encoder layers trained locally and fewer globally. To optimize communication, only a subset of clients participates in each global update, using the FedAVG algorithm to train the global layers.

Yu et al. [88] introduced a method to select a subset of parameters from the main model to be used in FL using saliency maps. The L1-norm is used to rank the original model's weights, and only a small percentage of these weights, determined by thresholding the L1 score, are sent to the server. The server performs FedAVG on the received features and sends only these features back to the clients. The authors demonstrate that this approach accelerates training of FLs in FL by nearly halving the computational time, while maintaining and even improving performance. This suggests that performing FedAVG on a subset of parameters can act as a form of regularization, similar to stochastic weight averaging [26].

3.1.3. Test-time adaptation for training

The methods in this class only optimize the input to the FMs, by collaboratively adjusting it using FL [80].

Guo et al. [27] introduced a method for collaboratively training prompts instead of the entire model. Their approach assumes that each client pre-trains or uses an off-the-shelf FM on their local data, and uses FL to optimize continuous prompts, which are learnable vectors directly integrated into the model's embedding space. The method introduces a new set of learnable input parameters representing the continuous prompt and optimizes only these parameters in FL using FedAVG. While this significantly reduces communication costs, it presents additional challenges, as it assumes clients can pre-train models independently on their data. Additionally, adjustments are needed, as the number of clients involved in optimizing the prompts directly affects performance. **Su et al. [66]** proposed a similar method where prompts are trained collaboratively, but the contribution of each client's prompt is modulated by a learnable parameter called a key. This technique mirrors the aggregation methods used in training entire models, but is specifically applied to prompts. Using the learnable weight (key) enables a more personalized aggregation process, where clients with similar domains, but different distributions, can customize their contribution to the final prompt. **Zhao et al. [95]** extended the experiments with continuous prompts to evaluate whether the exchange of prompts can compromise the privacy of FL, and found that introducing prompt tuning in FL does not inherently breach privacy.

3.2. Customize

Customizing existing FMs using FL is one of the most attractive use cases, as it is more efficient to train a FM in a centralized manner or to use an off-the-shelf pre-trained FM and then customize it using FL. This approach aligns well with the ML development life-cycle, where starting from pre-trained models has become a standard practice for most tasks. Therefore, the algorithms in this class are also the most numerous. To initially classify them, we first consider the type of method used for customization: (i) *fine-tuning* – where parts or the entire model are fine-tuned in FL using supervised or semi-supervised learning, (ii) *contraction* – where the size of the model is reduced using various techniques, and (iii) *hybrid* methods – combining fine-tuning and contraction. On the server side, most algorithms discussed in this section use FedAVG or its variants for parameter aggregation. Within these classes, several sub-categories can be identified, which are detailed in the following sections.

3.2.1. Fine-tuning

Fine-tuning methods involve continuing to train a FM in FL using supervised or semi-supervised learning. Given that fine-tuning the entire model is both computationally and communication intensive, most methods in this class focus on fine-tuning only a selection of existing or new parameters. To further distinguish between these methods, we classify them into two categories: methods that select only parts of the model for fine-tuning (selective) and methods that add extra parameters for fine-tuning (additive), both of which are presented in the following paragraphs.

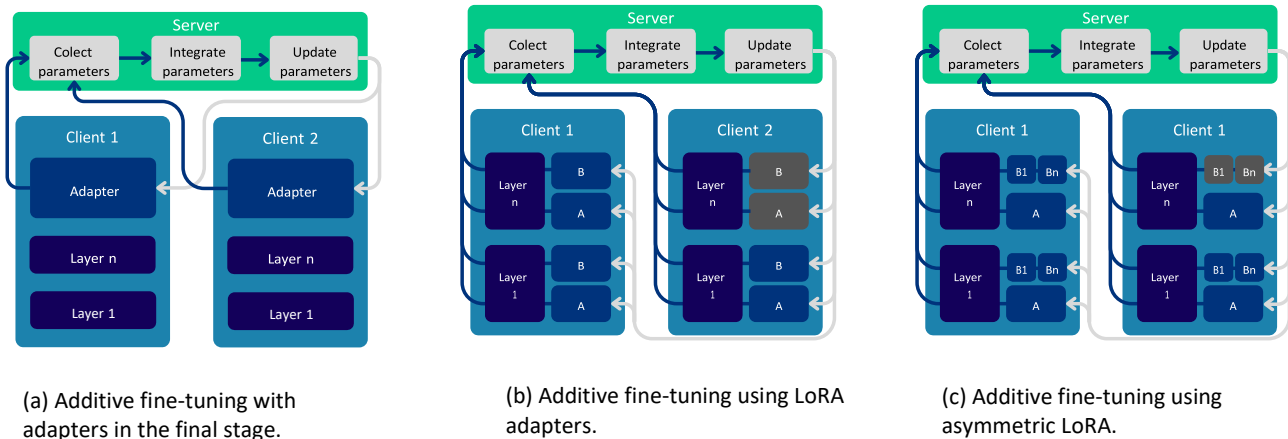


Figure 4 - Overview of additive fine-tuning methods, where the grey boxes mean that the added parameters can be heterogeneous between clients.

Selective Selective methods fine-tune only a subset of the existing parameters of the FM using FL. Unlike pre-training with selective methods (APPENDIX A.2), the methods in this class use supervised or semi-supervised learning and primarily focus on fine-tuning the bias parameters of a model in FL, while keeping the rest of the parameters frozen. Sun et al.[67] are among the firsts to evaluate the performance of fine-tuning only the bias parameters (also known as bias tuning) and compare it to

adding extra adapters under various settings, including distinct client stability, data distributions, and differential privacy settings. During training, each client performs a forward pass through the model, sends the bias parameters to the server, which aggregates them using FedAVG and returns the outcome to all clients. The authors find that bias tuning consistently outperforms additive fine-tuning, across distinct use cases involving both language and vision. This performance gap is maintained across various settings, such as when differential privacy is used for training, or when all clients operate in low-data regimes. [Chen et al.\[13\]](#) conduct a similar study and arrive at comparable conclusions using different types of adapters.

[Tsouvalas et al.\[71\]](#) present a novel technique that selects a subset of network parameters based on the L1 norm of the weight matrices at each layer. The algorithm ranks these matrices using the L1 norm and aggregates only the top n matrices, with n being determined by the specific use case. This approach significantly reduces the number of parameters involved in the aggregation process. The authors demonstrate a reduction of more than 60% in the number of parameters for large NLP architectures such as BERT-Large. However, they observe a less pronounced decrease for smaller models, indicating that the effectiveness of this technique may vary depending on the model size and complexity.

Additive Additive methods introduce extra parameters that are fine-tuned while keeping the FM frozen. These methods focus on adding extra parameters in the final layers, typically called adapters, which project the final embeddings of the FM and adapt them to new tasks, or on using parameter-efficient fine tuning (PEFT) (particularly low-rank adaptation (LoRA) – [APPENDIX A.3](#)) to add parameters at each layer. When using LoRA, a common question arises: should the added LoRA ranks be homogeneous or heterogeneous, given that some clients may have better computational resources or larger datasets. Additionally, within LoRA, there is the possibility of adding asymmetric parameters between the low-rank matrices. An illustration of these methods is provided in [FIGURE 4](#), where [FIGURE 4A](#) shows the addition of extra adapters, [FIGURE 4B](#) illustrates the use of LoRA with either homogeneous or heterogeneous (gray) parameters, and [FIGURE 4C](#) illustrates the use of asymmetric LoRA parameters.

[Houlsby et al. \[32\]](#) are the first to analyze which parameters of a FM should be fine-tuned for better precision by comparing fine-tuning directly the last layers of a FM with adding one extra layer to fine-tune ([FIGURE 4a](#)). The authors find that adding extra adapters not only improves the performance, but also leads to more stable training. [Chen et al. \[13\]](#) test this idea in FL settings and find out that using an adapter does improve the performance but only in some cases. In other cases, selecting and fine-tuning only the bias parameters of a transformer leads to better results (discussed in the selective section above).

[Legate et al. \[42\]](#) extend this idea and propose to first fine-tune only the extra adapter, followed by some steps in which the complete model is fine-tuned using FedAVG. This second step is shown to further improve performance; however, it increases the communication costs significantly.

When using LoRA in FL, a key consideration is whether to make the added low-rank parameters homogeneous (identical) across clients or heterogeneous. Several studies use homogeneous LoRA, as discussed in [APPENDIX A.3](#), for various tasks. [Nguyen et al. \[55\]](#) used it to fine-tune a vision-

language model where the encoders for either the vision or language components are extended using LoRA. [Jiang et al. \[34\]](#) used LoRA to fine-tune large language models (LLMs), while [Zhang et al. \[90\]](#) used it for instruction-tuning LLMs in FL. [Yi et al. \[86\]](#) applied it in scenarios where clients own heterogeneous FMs, but use homogeneous parameters for fine-tuning, by manually selecting the layers for parameter addition. In all these cases, the aggregation of global (homogeneous) LoRA parameters is performed using FedAVG, with the mentioned studies showing consistent performance improvements across different benchmarks and modalities.

However, using homogeneous LoRA ranks presents a trade-off between overfitting and slow convergence, especially for clients with more heterogeneous datasets or models (e.g., skewed distributions) or when personalization is required. In such cases, the definition and aggregation of local parameters for LoRA can be made more dynamic by introducing heterogeneous parameters. For example, [Guo et al.\[28\]](#) fine-tuned a multi-language LLM by introducing heterogeneous local adapters for each language family (e.g., Germanic or Italic), and aggregating only the parameters relevant to a specific language family globally. This approach helps to mitigate inter-language bias and offers flexibility for more challenging languages, at the expense of increased communication costs.

Additionally, [Cho et al.\[17\]](#) proposed a method to assign and aggregate heterogeneous ranks to all clients based on their system capabilities (e.g., distributing higher ranks to more capable clients and vice versa). The challenge then becomes aggregating the heterogeneous parameters into global LoRA parameters. To achieve this, the server pads all ranks to the same size and applies weighting based on the norm of the singular value vectors for the local parameters. The padded parameters are then aggregated using weighted FedAVG. This not only improves communication efficiency, as smaller parameter exchanges occur for some clients, but also acts as a form of regularization that enhances final performance compared to homogeneous LoRA. [Byun and Lee\[8\]](#) further investigated the padding-based aggregation of heterogeneous ranks and discovered that it introduces instabilities during training. To mitigate these instabilities, the authors used replication-based padding instead of zero-padding, which led to more robust and efficient fine-tuning.

More complex approaches, which combine multiple sets of homogeneous and heterogeneous adapters have also been explored. [Yang et al.\[83\]](#) proposed using two sets of LoRA parameters: a homogeneous set that is aggregated and updated globally, and a heterogeneous that is kept local. This technique can address skewed distributions between the client datasets or facilitate personalization. During each training step, the forward pass uses the frozen FM, the global adapter, and the local adapter. To balance the contribution of the global and local adapters, a weighting mechanism can be additionally employed. The global adapter is aggregated by the server using FedAVG, while the local adapter is kept individual for each system. This technique provides further adaptability without compromising the global performance.

[Ping et al.\[60\]](#) explore this concept in a multi-task setting by defining multiple (heterogeneous) LoRA parameters for each task. During each training step, only the parameters specific to a task are aggregated using FedAVG. In scenarios where task labels are not explicitly defined, the authors introduce a mechanism that employs k-means clustering to group and aggregate the parameters.

This approach automatically clusters similar adapters from the clients, which are assumed to contribute to the same tasks, and aggregates them using FedAVG. The method demonstrates improved robustness in multi-task settings by mitigating cross-task drift. However, it incurs high computational costs.

Tian et al.[70] introduces additional heterogeneity by modifying one of the low-rank matrices from LoRA (A , B , APPENDIX A.3) to add multiple asymmetric matrices for B (illustrated in FIGURE 4c). This approach allows specialization for particular tasks in multi-task settings, similar to mixture of experts (MOE) [9]. During each training step, the matrix A is aggregated globally using FedAVG, while each part of matrix B is aggregated only with its corresponding part from other clients, also using FedAVG. The method demonstrates consistent performance improvements even when the combined sizes of the asymmetric matrices are equal to the initial size of B , providing evidence that specializing these matrices for particular tasks can improve overall performance without increasing the parameter count.

In addition to parameter homogeneity, Babakniya et al.[3] find that the initialization of the LoRA parameters can introduce instabilities and affect performance, as a single initialization may not suit all client data distributions. To mitigate these issues, the authors propose to initially fine-tune the entire model for a few steps, followed by fine-tuning only the LoRA parameters. This approach provides initialization and momentum from the entire data distribution from all clients, which represents a better initialization. However, fine-tuning the entire model comes with high computational costs and may not always be feasible. Wu et al.[77] extend this procedure by alternating between full fine-tuning and fine-tuning only the LoRA parameters. This method aims to balance global and local knowledge exchange, demonstrating performance improvements but at an even higher cost.

Yang et al.[84] further advance this idea by introducing a process similar to simulated annealing. They propose initially fine-tuning the entire model using FedAVG while applying the L2 norm to all parameters to mitigate potential client drifts. In the second stage, fine-tuning is limited to the LoRA parameters, but the ranks are adaptively decreased during training. This stage begins with more parameters in the early epochs, gradually reducing the number of parameters (ranks) as training progresses, similar to a learning rate scheduler. However, the method adds overhead for adjusting the ranks and only improves communication efficiency in the final stages when the number of parameters decreases.

Instead of fine-tuning the entire FM, Yan et al.[79] suggest initializing the LoRA ranks using the results of singular value decomposition on the pre-trained weight matrices. This approach introduces only a minor initialization step which helps stabilize the entire training process.

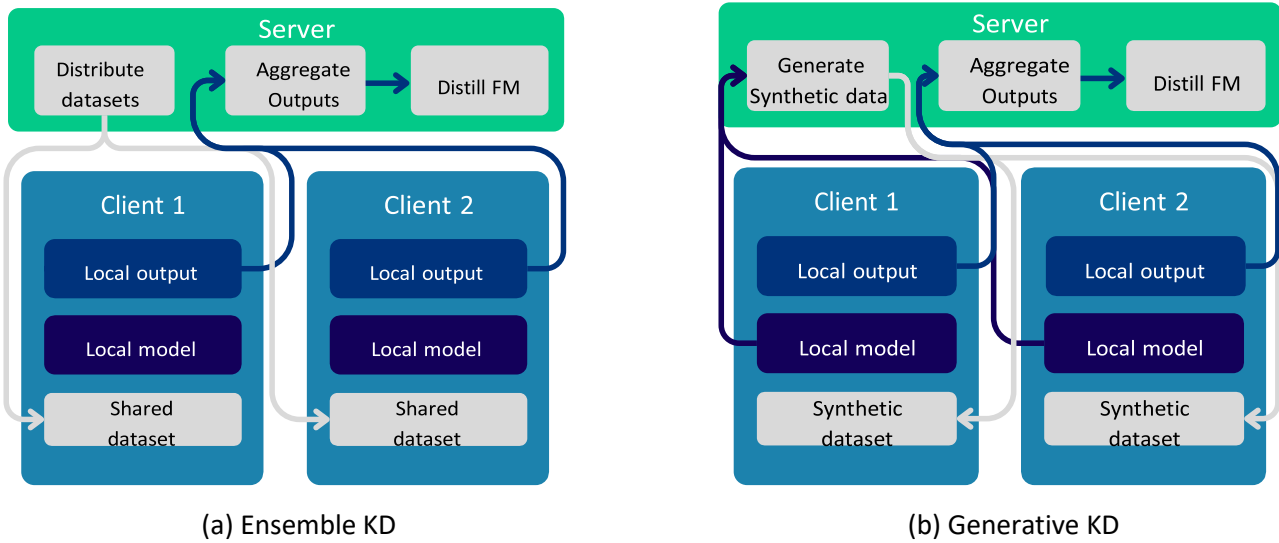


Figure 5 - Overview of KD methods

Zhang et al.[94] conducted an empirical study on additive fine-tuning in FL and found that, compared to fine-tuning the entire model, additive fine-tuning offers better defence against certain threat models, such as data reconstruction attacks. Additionally, they discovered that adaptive fine-tuning generally performs on par with complete fine-tuning while significantly reducing communication costs, by factors ranging from 12 to 190 times. However, additive fine-tuning is more sensitive to data heterogeneity.

3.2.2. Contraction

Contraction methods aim to decrease the size of the models to optimize communication and computation efficiency. The methods in this class are further classified in methods that transfer the knowledge from one model to another (distillation) and methods that aim compress the FM (through compression, pruning, or quantization).

Distillation As discussed in APPENDIX A.4, KD in FL involves transferring knowledge from local models to a global, in this case, FM. This process is equivalent to training local models on local datasets and then distilling this knowledge into the FM using a shared dataset [74]. Additionally, it is possible to distill the local datasets by training synthetic data generators.

This synthetic data can either be shared directly with the global model or used to distill the knowledge from the local models to the global model. An illustration of these techniques is provided in FIGURE 8 , FIGURE 5A illustrates training local models on private data and distilling the knowledge on public data and FIGURE 5B illustrates the scenario where a synthetic data generator is trained using locally trained models. While the intersection of KD and FL has been explored, we focus on presenting papers that use FMs or aim to customize pre-trained models. For a more comprehensive overview of KD and FL, we refer readers to the work of Li et al.[43].

Gong et al.[23] are among the first to customize a FM using ensemble KD (FIGURE 5A). In the first stage of their approach, each client trains a local model on their private datasets. In the second stage, the local datasets are disconnected, and a central public dataset is used for distillation. At each training step, the server distributes data to each node, which runs inference on the local model and sends the results back to the server. The server then aggregates these results using FedAVG and uses them to distill the knowledge to the FM using the KLdivergence as described in SECTION A.4. Additionally, beyond performing KD only with the final outputs, the authors introduce a mechanism to also distill the attention from transformer layers, which improves the overall effectiveness of the knowledge transfer by transferring intermediate features as well. While the method demonstrates performance gains, especially when compared to training the model using fine-tuning with FedAVG, it introduces a dependency on the availability of a dataset, which should be sufficiently large to enable KD between the local nodes and the global FM. While the method demonstrates performance gains, particularly when compared to fine-tuning with FedAVG, it introduces a dependency on the availability of a sufficiently large public dataset, that can facilitate KD between the local nodes and the global FM.

Wu et al.[76] assume the existence of a FM for each client node, assuming the clients can pre-train their own models or use off-the-shelf FMs. In this setting, each client keeps the FM private and trains a small-scale proxy model used to exchange the knowledge. The proxy models are aggregated by the server using FedAVG. Wu et al.[76] assume the existence of a FM for each client node, either by allowing clients to pre-train their own models or by using off-the-shelf FMs. In this setting, each client keeps the FM private and trains a smallscale proxy model used to exchange knowledge. At each training step, KD is used between the FM and the proxy. The proxy models from each client are then aggregated by the server using FedAVG. This approach ensures that the knowledge from the private FMs is shared and integrated into the global model without sharing the models. However, it assumes the existence of FMs at each client, which may not always be realistic.

Zhang et al.[92] introduce a method where the server trains a synthetic data generator used for KD, as illustrated in FIGURE 5B. In this framework, each client trains their own local model and shares it with the server. The server then trains a synthetic data generator using these local models, aiming to match the semantics of the outputs from the local models with synthetic data. Subsequently, the generator is used to produce synthetic data, perform inference on the local models, and distill this knowledge to the global model using KD (as described above). This method ensures that the knowledge from the local models is transferred to the global model without the need to share the original data. However, it introduces additional failure dependencies on the quality of the synthetic data generated, which can impact performance in different scenarios Zhang et al.[91] present a similar framework where the server trains a data generator. However, in their approach, the generator is also trained using KD, instead of using a semantic loss.

For language-specific tasks, Deng et al.[18] do not train a new generator but instead use the FM directly to generate new data. In this framework, each client trains a small model on their local, private data. The clients then transfer their models to the server, which uses them to calibrate the

generation process and synthesize a new dataset. This dataset is subsequently used for KD, as described above.

Compression, Pruning, and Quantization While techniques such as compression, pruning, and quantization have been extensively studied for reducing model size and implicitly lowering communication costs in both FL and FMs, their intersection has received relatively little attention in the literature. Compression involves either compressing the weights of local models (e.g., using singular-value decomposition [81]) before transmission to the server, or compressing the gradients for back-propagation (e.g., using stochastic sign-based methods [69]) and aggregating them on the server. Pruning involves iteratively removing model parameters with small values during collaborative training [36], while quantization reduces the number of bits used to represent the weight matrices [16].

As these concerns can be considered together with other customization techniques such as fine-tuning techniques, most of the works for FMs using compression, pruning, or quantization will be discussed in the *hybrid* section. The only study on Foundation Models (FMs) without additional techniques is by Yang et al.[82], who employ quantization to compress the weight matrices (excluding other parameters like biases) in a large FM from 32 to 16 bits. To mitigate the effects of quantization, the authors propose applying a linear (learnable) transformation on the server side when integrating the aggregated parameters. Additionally, they suggest alternating and quantizing only a subset of parameters for each client, varying the selection across clients to ensure the server receives precise updates from clients that did not quantize those parameters. Using these techniques, they achieve communication cost reductions of over 60% in some cases, assuming that clients can perform inference on the FM locally and the server can run back-propagation for the quantized parameters.

3.2.3. Hybrid

Hybrid methods use both fine-tuning and contraction or multiple contraction methods methods to further improve communication costs.

Chen et al.[16] use selective fine-tuning with quantization to reduce communication costs. Instead of selecting individual weights, the authors propose partitioning the FM into blocks containing multiple layers and selecting only some of these blocks for fine-tuning in FL. At each training step, each block is assigned an importance score using the average L2 norm between the block values from the last two epochs. The score is used to select only the top n blocks, where n is determined based on the task. To further reduce communication costs, the selected blocks are quantized using stochastic quantization [20] before being sent to and aggregated on the server (using FedAVG). This method demonstrates consistent performance improvements and reduced communication costs compared to other selective fine-tuning approaches.

Chen et al.[12] introduce a hybrid approach that combines adapter-based fine-tuning with knowledge distillation. Similar to the dual adapter methods discussed in the additive finetuning section, each client maintains a global adapter and a local client-specific adapter. During each

training epoch, clients share the global adapter, which is aggregated by the server and then redistributed to the clients. This global adapter is used to distill its knowledge into the local adapter using the client’s local dataset. The proposed method demonstrates consistent performance improvements over traditional adaptive fine-tuning or distillation techniques. However, it introduces additional computational constraints due to the local distillation process. [Kuo et al.\[41\]](#) prune the LoRA parameters before uploading and aggregating them on the server by applying a magnitude-based threshold. Using the shared parameters, the server generates a global sparsification mask, averages the parameters using FedAVG, and then distributes the parameters back to the clients. The clients update only the parameters corresponding to the sparsification mask, retaining all other parameters as they were before sparsification, enabling dense training. This method demonstrates an order of magnitude improvement in communication costs while maintaining competitive performance with full LoRA fine-tuning. [Yadav et al.\[78\]](#) improve this method by incorporating quantization of the sparsified parameters, resulting in further size reduction.

[Wu et al.\[75\]](#) propose a method that combines compression with distillation and adaptive fine-tuning. Initially, the server compresses the FM into a more compact model by extracting a sub-model using layer dropout. This sub-model acts as a student model to distill information from the FM using a public dataset. The compressed model is then used with LoRA and trained within in FL, as described in the adaptive fine-tuning section.

3.3. Deploy

Deploying models through FL involves performing inference at each client using their local models, rather than relying on a final global model. This approach can be viewed as delivering an inference service leveraging the FL infrastructure [\[29\]](#).

The only study that addresses the deployment of FMs using FL is the work by [Liu et al.\[48\]](#). The authors proposed projecting multi-scale embeddings from intermediate layers and accumulating these projections across multiple clients to perform predictions. For example, in a classification task using transformer architectures, the class token responsible for classification is extracted from intermediate layers. The choice of layer is determined by the available resources at each client. Clients with fewer resources can use the token from earlier layers, while those with more resources can use the token from later layers. The final prediction is an average of the client predictions.

4. FMs and FL in healthcare applications

Healthcare and medical applications are among the most compelling use-cases for FL [\[59\]](#), and represent a significant area where FMs can make a substantial impact [\[54\]](#). However, despite the potential, relatively few articles have explored the use of FMs and FL in medical applications.

[Manoel et al.\[51\]](#) fine-tuned a multilingual FM for medical transcript analysis. At each epoch, the clients fine-tune the entire model using their own datasets for a specific number of batches and

upload the entire model to the server. The only optimization introduced is to train the model locally for a larger number of batches, to decrease the communication costs (similar to [63]). Wang et al.[72] introduced a new dataset with eight distinct medical tasks, including classification, anomaly detection, and generative tasks, designed for fine-tuning FMs using FL. The authors conduct a comparative benchmark using this dataset. In their setup, each client trains a local model on their private data and uploads it to a server, which employs FedAVG to aggregate these models into a global model. The global model is then used to incorporate local knowledge into a FM by concatenating the global model’s output with the input of an FM. Using a public dataset, the FM is fine-tuned together with the local knowledge. Using this method, the authors demonstrate performance improvements across all tasks in the dataset. Liu et al.[47] used FL to develop a foundational segmentation model based on the segment anything method [40]. The authors perform additive fine-tuning, where each client fine-tunes additional lightweight adapters for the segment anything model using their local data. The adapters are then aggregated on the server using FedAVG. Empirical experiments show that the performance achieved using FL is comparable to that of centralized training, while significantly reducing the resources required for training.

5. Practical perspectives

From a practical standpoint, choosing the right algorithms to experiment with can be challenging. This is because many algorithms are interconnected, and their trade-offs beyond performance are not always evident. For example, it can be difficult to differentiate between various additive fine-tuning methods and decide which one to prototype first or trace its incremental development. Furthermore, comparing the algorithms’ performance is not always relevant, as they are tested on distinct datasets and modalities.

To assist with this issue, we present an initial classification of the methods discussed in this report, using three criteria. First, we evaluate the complexity of the method, which is estimated based on the effort required to implement it. For incremental methods, we consider the base method to have the lowest complexity, with any additional features or modifications increasing its complexity. The extent of this increase depends on the nature and complexity of the additions. After classifying the methods based on their inherent complexity, we compared and scaled the base methods accordingly. Second, we assess the efficiency of the method, which is measured by the amount of information transmitted from the clients to the server. Here, we follow a similar approach to reduce iterative methods to their base method and first compare the base methods. Third, we consider the scalability of the method, which is estimated based on the resources required for both the client and the server. For example, if a method requires clients to run a FM on their premises, it will have lower scalability compared to methods that require inference on smaller models. To facilitate comparison, we classify each criterion on a scale ranging from 1 to 3, denoted as * to *** in TABLE 1. This classification system offers initial guidance in choosing methods to experiment with first, based on project constraints, and then progressing to more complex or resource-intensive methods later on.

Table 1 - Classification of methods based on their complexity, efficiency, and scalability

Technique		Complexity			Efficiency			Scalability		
		*	**	***	*	**	***	*	**	***
Pretrain	Entire	[6, 63, 62]	[96, 44]	[50, 61]	[6, 44, 50, 62]	[63, 61]	[96]	[6, 44, 50, 62, 61]	[63, 96]	–
	Partial	[68, 14, 38, 46, 88]	[49]		[49]	[68, 38, 46]	[14, 88]	[49]	[68, 14, 38, 46, 88]	–
	Test-time ad.		[27, 66, 95]			[27, 66, 95]			[27, 66, 95]	
Customize	Fine-tune	[67, 13, 71, 32]	[42, 55, 34, 90, 86, 84, 79, 94]	[28, 17, 8, 83, 60, 70, 3, 77]	[60, 3, 77, 84]	[67, 13, 42, 55, 34, 90, 86, 28, 17, 8, 83, 70, 79, 94]	[71, 32]	[42, 60, 70, 3, 77, 84]	[67, 13, 55, 34, 90, 86, 28, 8, 83, 79, 94]	[71, 32, 17]
	Contraction	[82]	[23]	[76, 92, 91, 18]	[76, 92, 91]	[23, 18]	[82]	[23, 76, 91]	[92, 18]	[82]
	Hybrid	[16]	[12, 41, 78]	[75]		[12, 75]	[16, 41, 78]	[75]	[12]	[16, 41, 78]
Deploy		[48]			[48]			[48]		

We observe that, for pre-training FMs with FL, the methods used for pre-training the entire model exhibit the lowest scalability but also low complexity. This is because basic algorithms can be employed, but the resources required to run training at each client and aggregate the results are substantial. Similarly, methods that assume clients can independently train FMs, such as those described in [Makhija et al.\[50\]](#), have higher complexity and low scalability. In general, pretraining FMs in FL demands significant resources, as either the clients or the server must perform forward passes on the model, which are generally computationally intensive. Therefore, from a practical standpoint, it is best to begin with partial model training, particularly using selective methods that can be both efficient and scalable. Test-time adaptation methods are suitable only for certain types of models, such as large language models or text-based models.

For customizing FMs in FL, we observe that methods using adapter fine-tuning with small adapters, such as adding a final layer, are the least complex and the most scalable and efficient. As discussed in [SECTION 3.2.1](#), these methods also provide good performance, making them a recommended starting point. More complex additive methods, such as those using LoRA, require forward passes of the entire model at the clients while aggregating only the additive parameters. This introduces additional complexity in managing the extra parameters. Similarly, more advanced LoRA methods that employ heterogeneous or dual adapters further increase complexity, although some, like those described in [\[17\]](#), may offer scalability improvements.

From the contraction methods, compression and quantization algorithms are the fastest to prototype, as they have lower complexity and high efficiency and scalability. Moreover, these algorithms can be used in conjunction with any other types of algorithms, making them highly versatile. The KD algorithms are relatively complex. They assume either the existence of an independent dataset or the ability to train independent FMs or data generators for each client. These constraints make KD algorithms less practical for many use cases, as they require additional resources and infrastructure.

6. Insights for the FLUTE project

The lack of healthcare studies using FL and FMs ([SECTION 4](#)) offers FLUTE a unique opportunity to innovate. Furthermore, the majority of the articles identified in this study use core FL algorithms, with almost all relying on plain FedAVG. As FLUTE plans to integrate Privacy Enhancement Technologies (PET) on top of FL and FMs to further strengthen privacy protections, this presents another opportunity to develop secure and efficient FMs tailored for healthcare.

Nevertheless, PET technologies add additional overhead, both in terms of complexity, efficiency and scalability. Therefore, some methods discussed in this study can present challenges when integrating in FLUTE. Besides the classification introduced in [SECTION 5](#), we present a series of recommendations for the short and long-time development and impact of FLUTE.

For the collaborative training of FMs using FL, partial model pre-training with additive methods emerges as an optimal choice for integration with PET [\[14\]](#). This approach offers lower complexity and higher efficiency, making it more adaptable to the increased overhead from technologies like

encryption for shared parameters, while methods for entire model pre-training are still difficult to run even without PET. Similarly, test-time adaptation methods are tailored to specific models like LLMs, which have limited applicability to FLUTE’s vision-driven objectives and are generally less suitable due to their limited applicability.

As discussed in [SECTION 3.2](#), customizing FMs using FL is the most compelling use case, particularly for practical applications like the FLUTE study. Here, fine-tuning, hybrid fine-tuning, and contraction methods such as compression or quantization offer significant benefits in terms of efficiency and scalability. Additionally, these methods allow for the integration of PET. Therefore, the near-future roadmap of FLUTE should prioritize the development and adaptation of fine-tuning methods for PET, particularly selective fine-tuning using bias parameters [\[67\]](#), [\[13\]](#) or additive fine-tuning methods where only final layers are added [\[13\]](#). Studying the trade-offs between integrating PET technologies with these methods and their impact on efficiency and scalability will likely lead to the development of improved finetuning methods for FL. LoRA technologies are also very popular, despite being more complex, and should be considered as well, especially heterogeneous LoRA methods where client resources can be balanced, such as [\[17\]](#). Furthermore, LoRA methods are suited for clientbased personalization, as local adapters can be trained together with global adapters [\[83\]](#).

KD methods have high complexity and often require additional resources, such as clients training local FMs or using public datasets for distillation. Therefore, a reasonable use-case for KD is to synthetic data generators [\[92\]](#), [\[91\]](#) which can be used to train a global model or generate more data, and can be integrated with other WPs and goals of FLUTE such as generating synthetic data. In this scenario, integrating PET technologies would occur after local models are trained, presenting an opportunity as the server is more powerful and can alleviate some of the overhead of PET. If public datasets are available, ensemble KD methods [\[23\]](#) can be prototyped, as they are relatively scalable and can accommodate the integration of PET technologies. However, the performance benefits over fine-tuning are not yet obvious.

Based on these observations, we believe that in the short term, fine-tuning technologies should be prioritized for integration with PET and FLUTE, and prototyped for FLUTE’s use cases. By analyzing diverse trade-offs encountered during the integration, novel optimized methods will emerge. Similarly, KD technologies should be implemented only if specific use cases, such as collaboratively generating synthetic data, justify their complexity and resource requirements.

7. Conclusions

We conducted a comprehensive literature review on FMs using FL, covering all stages of the development life cycle, including training, using, and deploying models. Over 260 articles were manually inspected, with more than 40 identified as relevant for our study. To classify the methods presented, we developed a custom taxonomy and classification criteria based on complexity, efficiency, and scalability. This review enables us to propose recommendations and insights for the FLUTE project, such as prioritizing fine-tuning techniques using selective or additive parameters and

integrating them with PET technologies developed in other WPs. The relatively low number of articles using FMs with FL in the healthcare domain opens new innovation opportunities for FLUTE, and confirms the need for novel approaches in this area.

APPENDIX A - PREREQUISITES

This section provides a brief overview of the prerequisite knowledge for FL, FMs, fine-tuning and KD that is used most frequently by the articles discussed in [SECTION 3](#).

A.1 Federated Learning

In FL, the objective is to collaboratively develop a global model by merging local model updates from participating clients without exchanging the private data stored on their devices. The training process usually involves a central server that orchestrates communication among multiple clients and aggregates local updates into a global model. Each client performs local updates using its own data and transmits these updates to the central server, which then integrates them using various techniques.

In the most basic implementation, during each training round, every client conducts a forward pass using a shared model architecture on its local data and computes an update for all parameters using a shared loss function. The clients then send their model updates to the server, which integrates them to form a global model and sends the updated parameters back to the clients. This iterative process continues, with each client replacing its local model with the global parameters and resuming training as described.

The majority of articles presented in this study use FedAVG to integrate the local client updates, which is defined as:

$$w^{t+1} = \sum_{k=1}^K \frac{n_k}{n} w_k^{t+1} \quad (1)$$

where n_k represents the number of samples on client k , and n is the total number of samples across all clients. This process weights each client's contribution to the global model according to the size of its local dataset.

A.2 Pre-training Foundational Models

The main method for pre-training FMs is self-supervised learning (SSL), which involves creating pretext tasks from unlabeled data. This approach eliminates the need for annotated data, allowing to significantly expand the training datasets. Some SSL methods are further extended to incorporate semi-supervised learning, allowing them to leverage any available annotated data. This is particularly useful when the annotations are not related to the downstream tasks or are noisy.

SSL methods can be broadly classified into three categories (i) contrastive learning methods (e.g., [\[30\]](#), [\[15\]](#)), where the pretext task aims to generate similar embeddings for inputs representing the same concepts (e.g., images of the same organ) and distinct (contrastive) embeddings for inputs representing distinct concepts (e.g., prostate vs. kidney); (ii) correlationbased methods (e.g., [\[5\]](#)), where the pretext task aims to decorrelate embeddings of distinct objects while preserving the variance of embeddings for similar concepts, and (iii) masked input modelling (e.g., [\[31\]](#), [\[4\]](#)), where

the task involves reconstructing original inputs from masked versions of it (e.g., masking words in sentences or patches of images). Recent studies suggest that, despite their differences in methodology and training objectives, SSL methods are closely related and may produce similar embeddings [22]. This is because they minimize criteria that are equivalent under certain conditions. These methods are independent of the architecture used, being universally applied to transformer or convolutional based architectures.

The predominant methods used in FL, as discussed in SECTION 3.1, use contrastive learning techniques that involve passing two augmented versions of an input through identical decoders, known as Siamese networks, but updating the decoders differently. The primary decoder, often referred as the student network, is updated through standard backpropagation. The secondary decoder, called the teacher network, is updated using different techniques aimed at preventing the encoders from generating overly the same embeddings, a phenomenon known as feature collapse. These techniques include the use of momentum or EMA as in BYOL [25], [10]. Simpler variants that use a single encoder for both input augmentations are possible but generally less effective e.g., as in SimCLR [15]. The teacher network is usually the only one updated in FL, using global aggregation. An illustration of these techniques is provided in FIGURE 6.

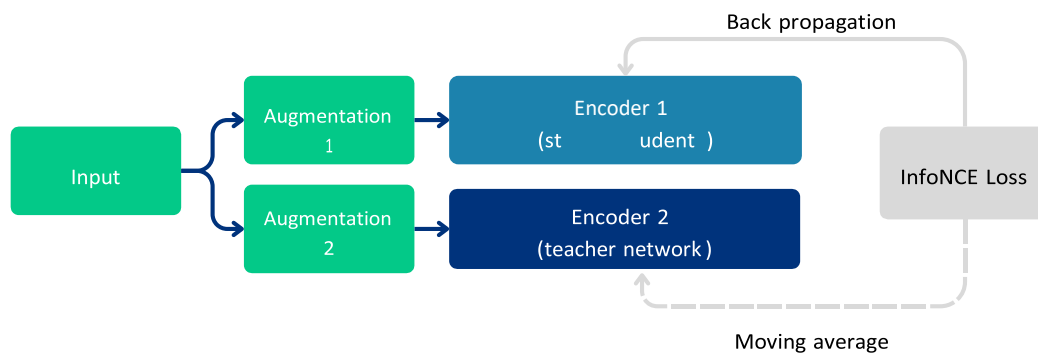


Figure 6 - Illustration of pre-training FMs using contrastive learning

In all cases, the loss used is based on the InfoNCE contrastive loss [57].

A.3 Fine-tuning FMs and LoRA

Fine-tuning follows the pre-training the FMs on unlabeled data by continuing the training process using labeled data in a supervised learning manner. This step can be seen as specializing the FMs for a specific task, allowing the model to leverage the general knowledge gained during pre-training and adapt it to particular nuances of the task at hand. While all model parameters can be updated during fine-tuning, resource constraints may induce a more selective approach. One strategy is to update only a subset of the parameters, such as the final layers (also called adapter tuning), which can be effective in projecting and adapting the models' outputs for specific tasks. Alternatively, one can introduce additional, smaller sized parameters at each layer, allowing the model to learn task-specific features while keeping most of the pre-train parameters fixed.



Figure 7 - Illustration of LoRA

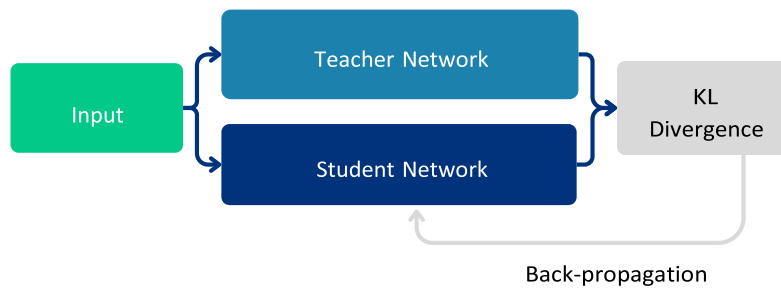


Figure 8 - Illustration of knowledge distillation

The most prevalent method for introducing additional parameters, frequently used in FL fine-tuning is known as LoRA. This method assumes that all model parameters have an intrinsic low rank that can be adapted during fine-tuning. Therefore, each model parameter (e.g., weight matrix) can be decomposed into the sum of the initial parameters and a low rank update expressed as $W_0x + \Delta Wx$. Here W_0 represents the frozen initial parameters, and Wx is the low-rank update optimized during fine-tuning. This low-rank is defined by two trainable matrices $Wx = BAx$, where A encodes the input to a lower dimensional rank and B recovers the output dimension of the original W_0 . An illustration of LoRA is provided in Figure 7. In transformer architectures, LoRA is applied to the attention weights rather than the multi-layer perceptrons (MLPs) layers.

LoRA can speed up fine-tuning of large-scale FMs with more than 25%, while decreasing the memory requirements by more than 103. Other LoRA variants, such as quantized LoRA further decrease the memory requirements by up to 90% [19].

A.4 Knowledge distillation

KD aims to transfer the knowledge from a large FM (called teacher) to a smaller, more efficient model (called student). This process involves training the student model to mimic the behavior of the teacher model by matching its outputs, such as logits or soft targets, rather than relying only on supervised signals such as labels. The most common method for minimizing the distance between the teacher's and student's outputs is the Kullback-Leibler (KL) divergence [24]. In the context of FL, KD is used to transfer knowledge from local models to a global model by minimizing the divergence

between their outputs. This approach allows the global model to benefit from the collective knowledge of the local models. An illustration is provided in Figure 8.

APPENDIX B – LIST OF TERMS FOR LITERATURE SEARCH

The complete list of terms used to develop queries for the literature review is illustrated in [TABLE 2](#).

Table 2 - List of terms for the queries. Queries were formed by first combining the first and second terms using the boolean AND operator, and afterwards combining the first, second, and third terms using the same operator.

First terms	Second terms	Third terms
Federated Learning	Foundational Models	Healthcare
	Self-supervised learning	Medical
	Pre-training	Applications
	Pre-trained models	Implementation
	Fine-tuning	Development
	Parameter-efficient fine-tuning	Deployment
	Adapter tuning	Inference
	Distillation	
	Transfer learning	
	Compression	
	Quantization	
	Pruning	

REFERENCES

- [1] Ankur Agarwal, Mehdi Rezagholizadeh, and Prasanna Parthasarathi. Practical takes on federated learning with pretrained language models. In *Findings of the Association for Computational Linguistics: EACL 2023*, 2023.
- [2] Muhammad Awais, Muzammal Naseer, Salman Khan, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, and Fahad Shahbaz Khan. Foundational models defining a new era in vision: A survey and outlook. *arXiv:2307.13721*, 2023.
- [3] Sara Babakniya, Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Qingfeng Liu, Kee-Bong Song, Mostafa El-Khamy, and Salman Avestimehr. Slora: Federated parameter efficient fine-tuning of language models. *arXiv:2308.06522*, 2023.
- [4] Alexei Baeovski, Arun Babu, Wei-Ning Hsu, and Michael Auli. Efficient self-supervised learning with contextualized target representations for vision, speech and language. In *Proc. of the 40th Int. Conf. on Machine Learning, ICML'23*, 2023.
- [5] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *Proc. of the 10th Int. Conf. on Learning Representations*, 2022.
- [6] Daniel Garcia Bernal, Lodovico Giaretta, Sarunas Girdzijauskas, and Magnus Sahlgren. Federated word2vec: Leveraging federated learning to encourage collaborative representation learning. *arXiv:2105.00831*, 2021.
- [7] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv:2108.07258*, 2021.
- [8] Yuji Byun and Jaeho Lee. Towards federated low-rank adaptation with rankheterogeneous communication. *arXiv:2406.17477*, 2024.
- [9] Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts. *arXiv:2407.06204*, 2024.
- [10] Zhaowei Cai, Avinash Ravichandran, Subhransu Maji, Charless Fowlkes, Zhuowen Tu, and Stefano Soatto. Exponential moving average normalization for self-supervised and semi-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 194–203, 2021.
- [11] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

- [12] Haokun Chen, Yao Zhang, Denis Krompass, Jindong Gu, and Volker Tresp. Feddat: An approach for foundation model finetuning in multi-modal heterogeneous federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.
- [13] Jinyu Chen, Wenchao Xu, Song Guo, Junxiao Wang, Jie Zhang, and Haozhao Wang. Fedtune: A deep dive into efficient federated fine-tuning with pre-trained transformers. *arXiv:2211.08025*, 2022.
- [14] Sijia Chen, Ningxin Su, and Baochun Li. Calibre: Towards fair and accurate personalized federated learning with self-supervised learning.
- [15] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *Proc. of the 37th Int. Conf. on Machine Learning*, ICML'20. JMLR.org, 2020.
- [16] Yuanyuan Chen, Zichen Chen, Pengcheng Wu, and Han Yu. Fedobd: Opportunistic block dropout for efficiently training large-scale neural networks through federated learning. *arXiv:2208.05174*, 2022.
- [17] Yae Jee Cho, Luyang Liu, Zheng Xu, Aldi Fahrezi, Matt Barnes, and Gauri Joshi. Heterogeneous lora for federated fine-tuning of on-device foundation models. In *International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023*, 2023.
- [18] Yongheng Deng, Ziqing Qiao, Ju Ren, Yang Liu, and Yaoyue Zhang. Mutual enhancement of large and small language models with cross-silo knowledge transfer. *arXiv:2312.05842*, 2023.
- [19] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 2024.
- [20] Yinpeng Dong, Renkun Ni, Jianguo Li, Yurong Chen, Hang Su, and Jun Zhu. Stochastic quantization for learning accurate low-bit deep neural networks. *International Journal of Computer Vision*, 127:1629–1642, 2019.
- [21] Eros Fani, Raffaello Camoriano, Barbara Caputo, and Marco Ciccone. Fed3r: Recursive ridge regression for federated learning with strong pre-trained models. In *International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023*.
- [22] Q. Garrido, Y. Chen, A. Bardes, L. Najman, and Y. LeCun. On the duality between contrastive and non-contrastive self-supervised learning. In *The 11th Int. Conf. on Learning Representations*, 2023.
- [23] Xuan Gong, Abhishek Sharma, Srikrishna Karanam, Ziyang Wu, Terrence Chen, David Doermann, and Arun Innanje. Ensemble attention distillation for privacy-preserving

- federated learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.
- [24] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.
- [25] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent—a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.
- [26] Hao Guo, Jiyong Jin, and Bin Liu. Stochastic weight averaging revisited. *Applied Sciences*, 13(5):2935, 2023.
- [27] Tao Guo, Song Guo, Junxiao Wang, Xueyang Tang, and Wenchao Xu. Promptfl: Let federated participants cooperatively learn prompts instead of models—federated learning in age of foundation model. *IEEE Transactions on Mobile Computing*, 2023.
- [28] Zhihan Guo, Yifei Zhang, Zhuo Zhang, Zenglin Xu, and Irwin King. Fedlfc: Towards efficient federated multilingual modeling with lora-based language family clustering. In *Findings of the Association for Computational Linguistics: NAACL 2024*, 2024.
- [29] Pengchao Han, Shiqiang Wang, Yang Jiao, and Jianwei Huang. Federated learning while providing model as a service: Joint training and inference optimization. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, pages 631–640. IEEE, 2024.
- [30] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735, 2020. doi: 10.1109/CVPR42600.2020.00975.
- [31] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 15979–15988, 2022. doi: 10.1109/CVPR52688.2022.01553.
- [32] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameterefficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.
- [33] Samireh Jalali and Claes Wohlin. Systematic literature studies: database searches vs. backward snowballing. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 29–38, 2012.

- [34] Jingang Jiang, Xiangyang Liu, and Chenyou Fan. Low-parameter federated learning with large language models. *arXiv:2307.13896*, 2023.
- [35] Lekang Jiang, Filip Svoboda, and Nicholas D Lane. Fdapt: Federated domain-adaptive pre-training for language models. *arXiv:2307.06933*, 2023.
- [36] Yuang Jiang, Shiqiang Wang, Victor Valls, Bong Jun Ko, Wei-Han Lee, Kin K Leung, and Leandros Tassiulas. Model pruning enables efficient federated learning on edge devices. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):10374–10386, 2022.
- [37] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 2021.
- [38] Gyunyeop Kim, Joon Yoo, and Sangwoo Kang. Efficient federated learning with pretrained large language model using several adapter mechanisms. *Mathematics*, 11(21): 4479, 2023.
- [39] Seongyeon Kim, Gihun Lee, Jaehoon Oh, and Se-Young Yun. Fedfn: Feature normalization for alleviating data heterogeneity problem in federated learning. *arXiv:2311.13267*, 2023.
- [40] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.
- [41] Kevin Kuo, Arian Raje, Kousik Rajesh, and Virginia Smith. Federated lora with sparse communication. *arXiv:2406.05233*, 2024.
- [42] Gwen Legate, Nicolas Bernier, Lucas Page-Caccia, Edouard Oyallon, and Eugene Belilovsky. Guiding the last layer in federated learning with pre-trained models. *Advances in Neural Information Processing Systems*, 36, 2024.
- [43] Haoran Li, Xinyuan Zhao, Dadi Guo, Hanlin Gu, Ziqian Zeng, Yuxing Han, Yangqiu Song, Lixin Fan, and Qiang Yang. Federated domain-specific knowledge transfer on large language models using synthetic data. *arXiv:2405.14212*, 2024.
- [44] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10713–10722, 2021.
- [45] Shenghui Li, Fanghua Ye, Meng Fang, Jiaxu Zhao, Yun-Hin Chan, Edith C-H Ngai, and Thiemo Voigt. Synergizing foundation models and federated learning: A survey. *arXiv:2406.12844*, 2024.
- [46] Zhengyang Lit, Shijing Sit, Jianzong Wang, and Jing Xiao. Federated split bert for heterogeneous text classification. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.

- [47] Yuxi Liu, Guibo Luo, and Yuesheng Zhu. Fedfms: Exploring federated foundation models for medical image segmentation. *arXiv:2403.05408*, 2024.
- [48] Zicheng Liu, Da Li, Javier Fernandez-Marques, Stefanos Laskaridis, Yan Gao, Łukasz Dudziak, Stan Z Li, Shell Xu Hu, and Timothy Hospedales. Federated learning for inference at anytime and anywhere. *arXiv:2212.04084*, 2022.
- [49] Wang Lu, Hao Yu, Jindong Wang, Damien Teney, Haohan Wang, Yiqiang Chen, Qiang Yang, Xing Xie, and Xiangyang Ji. Zoopfl: Exploring black-box foundation models for personalized federated learning. *arXiv:2310.05143*, 2023.
- [50] Disha Makhija, Nhat Ho, and Joydeep Ghosh. Federated self-supervised learning for heterogeneous clients. *arXiv:2205.12493*, 2022.
- [51] Andrea Manoel, Mirian del Carmen Hipolito Garcia, Tal Baumel, Shize Su, Jialei Chen, Robert Sim, Dan Miller, Danny Karmon, and Dimitrios Dimitriadis. Federated multilingual models for medical transcript analysis. In *Conference on Health, Inference, and Learning*. PMLR, 2023.
- [52] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 2017.
- [53] Tomas Mikolov. Efficient estimation of word representations in vector space. *arXiv:1301.3781*, 2013.
- [54] Michael Moor, Oishi Banerjee, Zahra Shakeri Hossein Abad, Harlan M Krumholz, Jure Leskovec, Eric J Topol, and Pranav Rajpurkar. Foundation models for generalist medical artificial intelligence. *Nature*, 616(7956):259–265, 2023.
- [55] Duy Phuong Nguyen, J Pablo Munoz, and Ali Jannesari. Flora: Enhancing visionlanguage models with parameter-efficient federated learning. *arXiv:2404.15182*, 2024.
- [56] Chitu Okoli. A guide to conducting a standalone systematic literature review. *Communications of the Association for Information Systems*, 37, 2015.
- [57] Advait Parulekar, Liam Collins, Karthikeyan Shanmugam, Aryan Mokhtari, and Sanjay Shakkottai. Infonce loss provably learns cluster-preserving representations. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 1914–1961. PMLR, 2023.
- [58] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv:2005.00247*, 2020.
- [59] Bjarne Pfitzner, Nico Steckhan, and Bert Arnrich. Federated learning in a medical context: a systematic literature review. *ACM Transactions on Internet Technology (TOIT)*, 21 (2):1–31, 2021.

- [60] Siqi Ping, Yuzhu Mao, Yang Liu, Xiao-Ping Zhang, and Wenbo Ding. Fl-tac: Enhanced fine-tuning in federated learning via low-rank, task-specific adapter clustering. *arXiv:2404.15384*, 2024.
- [61] Pol G Recasens, Jordi Torres, Josep Lluís Berral, Søren Hauberg, and Pablo MorenoMuñoz. Beyond parameter averaging in model aggregation. In *International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023*.
- [62] Yasar Abbas Ur Rehman, Yan Gao, Pedro Porto Buarque De Gusmão, Mina Alibeigi, Jiajun Shen, and Nicholas D Lane. L-dawa: Layer-wise divergence aware weight aggregation in federated self-supervised visual representation learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2023.
- [63] Lorenzo Sani, Alex Iacob, Zeyu Cao, Bill Marino, Yan Gao, Tomas Paulik, Wanru Zhao, William F Shen, Preslav Aleksandrov, Xinchu Qiu, et al. The future of large language model pre-training is federated. *arXiv:2405.10853*, 2024.
- [64] Robert T Sataloff, Matthew L Bush, Rakesh Chandra, Douglas Chepeha, Brian Rotenberg, Edward W Fisher, David Goldenberg, Ehab Y Hanna, Joseph E Kerschner, Dennis H Kraus, et al. Systematic and other reviews: Criteria and complexities, 2021.
- [65] Mojtaba Shahin, Muhammad Ali Babar, and Liming Zhu. Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE Access*, 5:3909–3943, 2017.
- [66] Shangchao Su, Mingzhao Yang, Bin Li, and Xiangyang Xue. Cross-domain federated adaptive prompt tuning for clip. *arXiv:2211.07864*, 3, 2022.
- [67] Guangyu Sun, Matias Mendieta, Taojiannan Yang, and Chen Chen. Exploring parameter-efficient fine-tuning for improving communication efficiency in federated learning. 2022.
- [68] Yue Tan, Guodong Long, Jie Ma, Lu Liu, Tianyi Zhou, and Jing Jiang. Federated learning from pre-trained models: A contrastive learning approach. *Advances in neural information processing systems*, 2022.
- [69] Zhiwei Tang, Yanmeng Wang, and Tsung-Hui Chang. z-signfedavg: A unified stochastic sign-based compression for federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 15301–15309, 2024.
- [70] Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. Hydralora: An asymmetric lora architecture for efficient fine-tuning. *arXiv:2404.19245*, 2024.
- [71] Vasileios Tsouvalas, Yuki M Asano, and Aaqib Saeed. Federated fine-tuning of vision foundation models via probabilistic masking. In *ICML 2024 Workshop on Foundation Models in the Wild*.

- [72] Jiaqi Wang, Xiaochen Wang, Lingjuan Lyu, Jinghui Chen, and Fenglong Ma. Fedmeki: A benchmark for scaling medical foundation models via federated knowledge injection. *arXiv:2408.09227*, 2024.
- [73] Herbert Woisetschläger, Alexander Isenko, Shiqiang Wang, Ruben Mayer, and HansArno Jacobsen. A survey on efficient federated learning methods for foundation model training. *arXiv:2401.04472*, 2024.
- [74] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Yongfeng Huang, and Xing Xie. Communication-efficient federated learning via knowledge distillation. *Nature communications*, 13(1):2032, 2022.
- [75] Feijie Wu, Zitao Li, Yaliang Li, Bolin Ding, and Jing Gao. Fedbiot: Llm local fine-tuning in federated learning without full model. *arXiv:2406.17706*, 2024.
- [76] Xidong Wu, Wan-Yi Lin, Devin Willmott, Filipe Condessa, Yufei Huang, Zhenzhen Li, and Madan Ravi Ganesh. Leveraging foundation models to improve lightweight clients in federated learning. *arXiv:2311.08479*, 2023.
- [77] Xinghao Wu, Xuefeng Liu, Jianwei Niu, Haolin Wang, Shaojie Tang, and Guogang Zhu. Fedlora: When personalized federated learning meets low-rank adaptation. 2024.
- [78] Prateek Yadav, Leshem Choshen, Colin Raffel, and Mohit Bansal. Compeft: Compression for communicating parameter efficient updates via sparsification and quantization. *arXiv:2311.13171*, 2023.
- [79] Yuxuan Yan, Shunpu Tang, Zhiguo Shi, and Qianqian Yang. Federa: Efficient finetuning of language models in federated learning leveraging weight decomposition. *arXiv:2404.18848*, 2024.
- [80] Fu-En Yang, Chien-Yi Wang, and Yu-Chiang Frank Wang. Efficient model personalization in federated learning via client-specific prompt generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19159–19168, 2023.
- [81] He Yang. H-fl: A hierarchical communication-efficient and privacy-protected architecture for federated learning. *arXiv:2106.00275*, 2021.
- [82] Tien-Ju Yang, Yonghui Xiao, Giovanni Motta, Françoise Beaufays, Rajiv Mathews, and Mingqing Chen. Online model compression for federated learning with large models. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.
- [83] Yiyuan Yang, Guodong Long, Tao Shen, Jing Jiang, and Michael Blumenstein. Dualpersonalizing adapter for federated foundation models. *arXiv:2403.19211*, 2024.

- [84] Yuning Yang, Xiaohong Liu, Tianrun Gao, Xiaodong Xu, and Guangyu Wang. Safedlora: Adaptive parameter allocation for efficient federated learning with lora tuning. *arXiv:2405.09394*, 2024.
- [85] Sarah M Yannascoli, Mara L Schenker, James L Carey, Jaimo Ahn, and Keith D Baldwin. How to write a systematic review: A step-by-step guide. *University of Pennsylvania Orthopaedic Journal*, 23:64–69, 2013.
- [86] Liping Yi, Han Yu, Gang Wang, and Xiaoguang Liu. Fedlora: Model-heterogeneous personalized federated learning with lora tuning. *arXiv:2310.13283*, 2023.
- [87] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *arXiv:2306.13549*, 2023.
- [88] Sixing Yu, J Pablo Muñoz, and Ali Jannesari. Bridging the gap between foundation models and heterogeneous federated learning. *arXiv:2310.00247*, 2023.
- [89] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning. *Knowledge-Based Systems*, 216:106775, 2021.
- [90] Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Tong Yu, Guoyin Wang, and Yiran Chen. Towards building the federatedgpt: Federated instruction tuning. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6915–6919. IEEE, 2024.
- [91] Jie Zhang, Chen Chen, Bo Li, Lingjuan Lyu, Shuang Wu, Shouhong Ding, Chunhua Shen, and Chao Wu. Dense: Data-free one-shot federated learning. *Advances in Neural Information Processing Systems*, 2022.
- [92] Lin Zhang, Li Shen, Liang Ding, Dacheng Tao, and Ling-Yu Duan. Fine-tuning global model via data-free knowledge distillation for non-iid federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022.
- [93] Shaoting Zhang and Dimitris Metaxas. On the challenges and perspectives of foundation models for medical image analysis. *Medical Image Analysis*, page 102996, 2023.
- [94] Zhuo Zhang, Yuanhang Yang, Yong Dai, Qifan Wang, Yue Yu, Lizhen Qu, and Zenglin Xu. Fedpetuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models. In *Annual Meeting of the Association of Computational Linguistics 2023*, pages 9963–9977. Association for Computational Linguistics (ACL), 2023.
- [95] Haodong Zhao, Wei Du, Fangqi Li, Peixuan Li, and Gongshen Liu. Fedprompt: Communication-efficient and privacy-preserving prompt tuning in federated learning. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

- [96] Weiming Zhuang, Xin Gan, Yonggang Wen, Shuai Zhang, and Shuai Yi. Collaborative unsupervised visual representation learning from decentralized data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4912–4921, 2021.
- [97] Weiming Zhuang, Yonggang Wen, and Shuai Zhang. Divergence-aware federated selfsupervised learning. *arXiv:2204.04385*, 2022.
- [98] Weiming Zhuang, Chen Chen, and Lingjuan Lyu. When foundation model meets federated learning: Motivations, challenges, and future directions. *arXiv:2306.15546*, 2023.