WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

Master's Thesis

Leveraging PySyft for Personalized Federated Learning in Advertising

Aleksandr Aronikov

Subject Area: Information Business

Studienkennzahl: 066/960

Supervisor: Prof. Dr. Sabrina Kirrane

Date of Submission: 24.April 2024

Institute for Information Systems and New Media, Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria

DEPARTMENT FÜR WIRTSCHAFTSINFORMATIK
UND OPERATIONS MANAGEMENT DEPARTMENT
OF INFORMATION SYSTEMS AND OPERATIONS
MANAGEMENT

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence.

APFL Adaptive Personalized Federated Learning.

AUC Area Under the Curve.

CTR Click-Through Rate.

DO Data Owner.

DP Differential Privacy.

DS Data Scientist.

DSP Demand-Side Platform.

FedAvg Federated Averaging.

FedPer Federated Learning with Personalization Layers.

FL Federated Learning.

FLS Federated Learning System.

FTL Federated Transfer Learning.

GDPR General Data Protection Regulation.

HE Homomorphic Encryption.

HFL Horizontal Federated Learning.

IID Identically and Independently Distributed.

KD Knowledge Distillation.

Max Maximum.

Min Minimum.

ML Machine Learning.

ModAg Model Aggregation.

NLL Negative Log-Likelihood.

NN Neural Network.

PETs Privacy-Enhancing Technologies.

PFL Personalized Federated Learning.

pp Percentage Points.

PQM Prediction Quality Metric.

RelImp Relative Improvement.

RPC Remote Procedure Call.

RQ Research Question.

SD Standard Deviation.

SGD Stochastic Gradient Descent.

SMPC Secure Multi-Party Computation.

SSP Supply-Side Platform.

TL Transfer Learning.

VFL Vertical Federated Learning.

Acknowledgments

I want to thank Prof. Dr. Sabrina Kirrane for her continuous support and supervision throughout the development of this thesis.

Also, I want to express my gratitude to Robert Darius and Hubert Burda Media for helping ground this thesis into a practical context.

Lastly, this thesis was only possible due to the OpenMined community.

Abstract

This thesis investigates Federated Learning (FL) and Personalized Federated Learning (PFL) architectures for Click-Through Rate (CTR) prediction in the domain of advertising, responding to the industry's current need for privacy-preserving technologies. A state-of-the-art literature review of PFL in the advertising industry is presented. Further, this thesis demonstrates the design and evaluation of different (P)FL architectures using the PySyft framework. The findings highlight that while (P)FL can outperform centralized models under conditions of data heterogeneity, the effectiveness varies significantly across datasets and architectures. The thesis compares these architectures, discussing their implications for model performance. Further research for (P)FL settings and potential practical deployment scenarios are suggested.

1 Introduction

The list of problems that can potentially be addressed by Artificial Intelligence (AI) is extensive and includes curing diseases, controlling crime, and protecting the climate [1]. Here, data forms a critical foundation. Indeed, the rapid success of Machine Learning (ML) can be partly attributed to the accessibility of large quantities of data (often referred to as big data). For example, model training and testing require data of considerable quantity, usually millions or billions of entries, to provide satisfactory results [2].

Frequently, data does not exist in a central location but is fragmented in a plethora of sources, such as user devices and small databases. Zhang et al. [3] call these small data storage areas 'data islands' (also 'data silos' or 'small data'). The defining characteristic of this type of data is that it is insufficient for the application of ML, as it displays at least one of the following features [2]:

- The quantity is too small.
- It lacks certain information (missing labels or values).
- It is unorganized and requires data cleaning.

As a result, considerable efforts are required to consolidate these data silos into usable big data. The standard process is to collect and store that data centrally, for instance, on a dedicated server. There, the data can be unified, cleaned, engineered, and further processed [3]. Interestingly, one can argue that the consolidation of big data is no longer the only focus. Instead, data security and privacy are now crucial challenges. Potential data leakage, for example, is a threat increasingly paid attention to by researchers and the public alike [4], [3], [5]. Regulations, such as the European Union's General Data Protection Regulation (GDPR), also play a significant role in that trend [6]. These developments pose substantial boundaries and limitations to the standard process of central data collection and processing [3].

1.1 Research Gap and Scenario

Privacy-Enhancing Technologies (PETs) are generally indicated as a potential solution for data protection and adherence to privacy regulations. These can also be applied in the context of ML [7]. Bonawitz et al. [8, p.5] note: "*Although privacy-preserving data analysis has been studied for more than 50 years, only in the past decade have solutions been widely deployed at scale*". This is partly due to the sheer amount of data, computing power, and devices needed to execute such deployments. One such solution is called FL. It is concerned with multiple parties collaboratively training ML models without exchanging their raw data [9].

Research Gap

While many examples of FL are demonstrated in the literature, only a subsection of the research concerned with using it in the context of advertising [2], [3], [8], [10]. By the same token, recent studies successfully illustrate practical applications of FL in that domain [11], [12]. A key challenge of the advertising industry is the heterogeneity of models and data (Section 3.4). This is partly caused by the increasing need for data

privacy and the resulting data leaks. This is also why advertisers are often unwilling to participate in data exchanges [12]. A potential solution for this is Personalized Federated Learning (PFL). It involves a personalized model that can be seen as a hybrid between a local and a global model. PFL is thus an attempt to combine the global model's generalization ability with the more tailored approach of the local model [13].

The main contribution of this thesis is a proof-of-concept application of FL and PFL in the context of advertising. Detecting such application scenarios and demonstrating their implementation can be a valuable contribution to this growing field of research [2]. Using the PySyft software framework for this purpose facilitates the implementation, as it simulates FL and does not require extensive amounts of hardware, data, and implementation effort to use [14], [2], [8]. The advertising domain is chosen in light of the practical need for such a solution. Both the application of FL and PFL are evaluated on a real-life dataset [11]. The dataset includes feature values and click feedback for display ads within the context of Click-Through Rate (CTR) prediction.

Research Scenario

A hypothetical example is given here to highlight the practical application. It is then later translated into the implementation setup. The example is split into the 'before' and 'after' situations. The first situation describes how conventional technologies are currently widely used to illustrate the status quo in the advertising industry. The second scenario represents a potential way of redesigning the architecture and setup in a federated way.

Before Scenario. Figure 1 illustrates the 'before' scenario, in which we assume that two media technology companies have collected user data. We can call each of them a Data Owner (DO). They want to use that data to create an ML model to predict ads' CTR. As their ML models benefit from a larger dataset, they exchange data (for example, using third-party cookies). Finally, they train their ML models on the combined data [12].

After Scenario. Figure 2 illustrates the 'after' scenario. Let us assume that the two media technology companies undergo a significant change due to the GDPR and the discontinuation of third-party cookies. Still, they want to improve their ML models to show users more relevant ads. They are, further, willing to continue exchanging data with each other but do so in a privacy-preserving way. FL is chosen for this purpose.

Therefore, they use PySyft, a readily accessible open-source library. The companies train a local model on their user data during the new process. Then, the DOs upload their model weights using PySyft. The model weights are combined into global model weights. Finally, an updated local model is trained using the global model weights for each company respectively. Different variants of FL and PFL are tested. More details about the architecture are introduced in Chapter 4.

The main motive of the new scenario is to achieve desirable privacy and utility simultaneously. On the one hand, the participants want to continue reaping the benefits of the improved ML models through data sharing. On the other hand, they want to navigate the new 'cookieless' digital advertising landscape, comply with new regulations such as GDPR, and meet new consumer privacy standards [15], [6], [5]. Overcoming this privacy-utility trade-off is suggested to be desirable and achievable [1]. This thesis expands on that in Section 2.2.1.

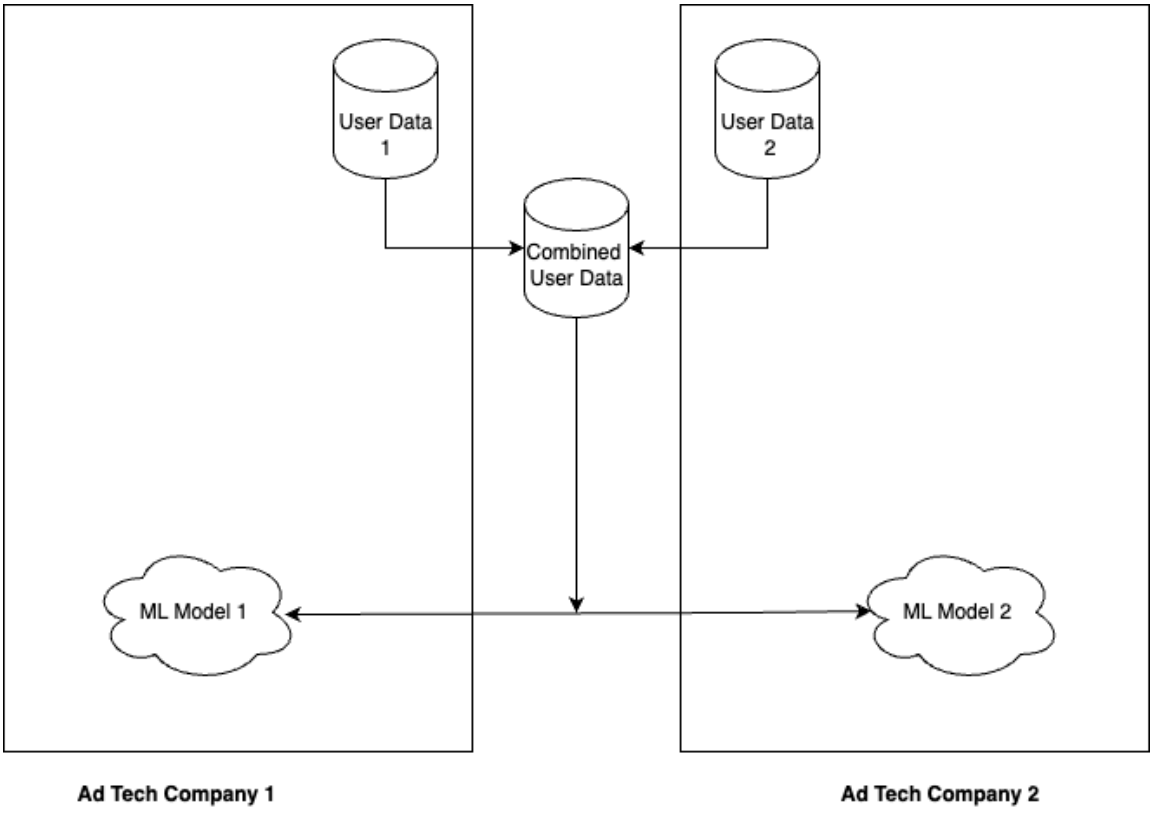


Figure 1: The before scenario

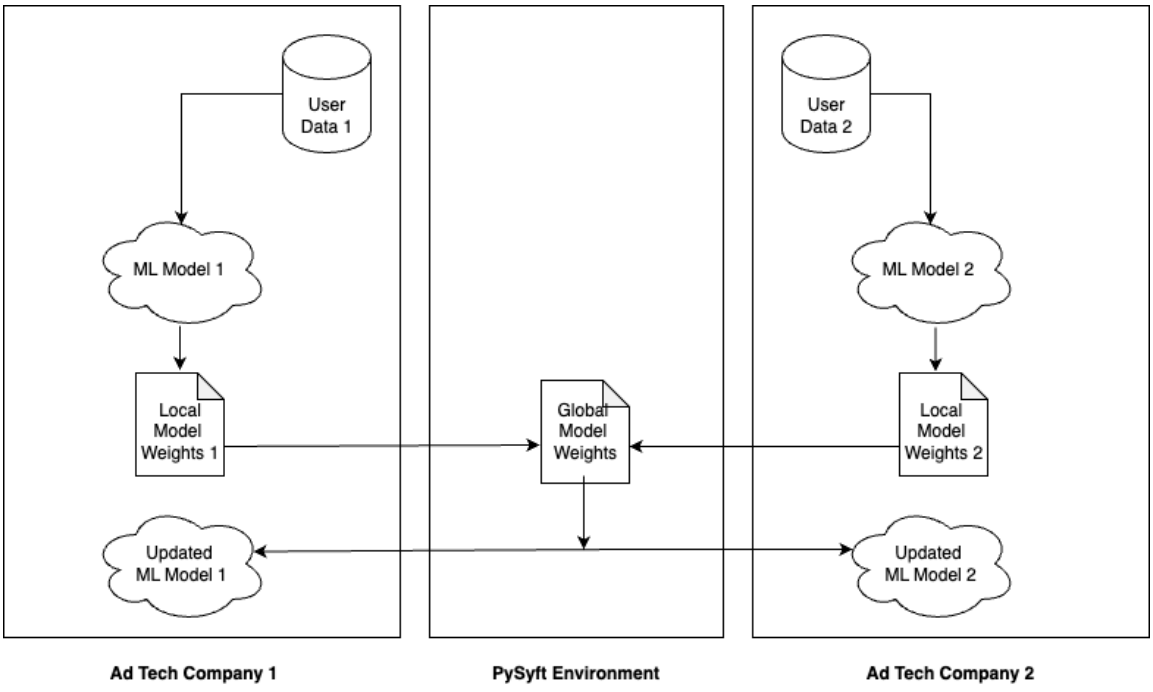


Figure 2: The after scenario

1.2 Research Question

Based on the previously mentioned research scenario, we derive the following overarching Research Question (RQ). It concerns evaluating the use of PySyft in the advertising context.

'How can federated learning through PySyft be used for privacy-preserving machine learning to predict click-through rates of ads?'

Four sub-RQs follow. They are revisited and answered in Chapter 6:

- **RQ1**: 'What is the state of the art regarding research in the field of applied PFL for CTR prediction in advertising?'.
- **RQ2**: 'How can two DOs exchange advertising CTR prediction model dictionaries federatively through PySyft?'.
- **RQ3**: 'How can architectures for PFL look like in that context?'.
- **RQ4:** 'How do the results of these FL and PFL implementations compare with each other when it comes to a set of metrics?'.

1.3 Methodology

The proposed research method falls into the field of design science. By creating a concrete, practical artifact for this novel use case, new knowledge can be generated [16]. Solving the problem practically advances the knowledge in a reproducible way. It further allows for potential discoveries in applying FL to business problems [16]. Hevner and Chatterjee [16] provide a list of guidelines for Design Science that practitioners can use as a checklist for its implementation:

1. ***'What is the research question?'*** [16, p.20].
   This thesis specified the RQ and its four sub-questions above.
2. ***'What is the artifact?'*** [16, p.20].
   The artifacts are this thesis, as well as several FL and PFL implementations via PySyft running on Jupyter Notebooks[1].
3. ***'What design processes (search heuristics) will be used when creating the artifact?'*** [16, p.20].
   The assumptions were that the existing PySyft documentation and provided examples on GitHub[2] were sufficient and translatable to the specific use case. Further, the same was assumed for the FL and PFL papers used for the implementation.

[1] github.com/PalexGoneGit/PFLAdsPySyft
[2] github.com/OpenMined/PySyft

4. ***'How does the knowledge base ground the artifact and design processes?'*** [16, p.20].
   One can distinguish between the practical and the academic knowledge base. For the first, the existing PySyft documentation and libraries[3] are used in the implementation part of this thesis. Further, Criteo's 'display advertising challenge' dataset and user submissions on Kaggle support this thesis[4]. The work is theoretically grounded in previous papers on privacy-preserving ML, particularly FL, PFL, and PySyft [17].

5. ***'What evaluations are performed during the internal design cycles?'*** [16, p.20].
   Firstly, technical evaluations of the code correctly running are used. Secondly, subject matter evaluations are utilized, such as asking oneself if the model is helpful for companies in hypothetical scenarios. Evaluation metrics include accuracy, Area Under the Curve (AUC), loss, and Relative Improvement (RelImp), which are explained in Section 4.2.3.

6. ***'How is the artifact introduced into the application environment, and how is it field tested? What metrics are used to demonstrate artifact utility and improvement over previous artifacts?'*** [16, p.20].
   The evaluation metrics introduced before are used. The different PFL implementations are compared to an FL implementation and a centrally trained version.

7. ***'What new knowledge is added to the knowledge base and in what form?'*** [16, p.20].
   A state-of-the-art table of practical PFL applications in the domain of advertising is shown in Section 3.4. Further, an implementation of ads CTR prediction for FL and PFL via PySyft is demonstrated[5]. Lastly, this thesis presents, analyzes, and discusses a comparison of three types of architectures (Central vs FL vs PFL) in Chapter 4. The results take the form of this thesis, a set of Jupyter Notebooks, and contributions on GitHub.

1.4 Structure

The thesis outline is as follows. Firstly, in Chapter 2, background information about the domain is provided. Specifically, online advertising and the problem of addressability are introduced. Then, a review of PETs follows. In particular, the pressing need for data security in ML is highlighted in that section. The concept of structured transparency, a potential solution, is presented. Lastly, popular PETs are described.

Chapter 3 is dedicated to FL and answering RQ1. Firstly, an introduction covers FLs definition and life cycle. Secondly, this thesis showcases the FL process. Differences to other ML systems are discussed. Thirdly, the following section encompasses a detailed categorization of an Federated Learning System (FLS), explaining its components and characteristics. It concludes with a proposed taxonomy of FLSs, which will be used throughout the paper. Lastly, this thesis sheds light on PFL. The last section of that chapter showcases the state of the art of application-focused PFL research in advertising.

[3] github.com/OpenMined/PySyft
[4] kaggle.com/competitions/criteo-display-ad-challenge
[5] github.com/PalexGoneGit/PFLAdsPySyft

Chapter 4 addresses RQ2 and RQ3, which are concerned with the implementation. First, it details PySyft, a framework enabling FL. Then, the dataset and the ML model are introduced. The central version, an FL implementation, and two different PFL implementations are described. Chapter 5 answers RQ4. It analyzes the implementations of the previous chapter and contrasts the performance metrics among the different variants. Then, a discussion interprets the results.

Chapter 6 acts as the conclusion and summarizes the results, highlighting the findings. Each of the four sub-research questions is answered, respectively. Then, the limitations of the work are discussed. Further research opportunities and practical implications are highlighted.

2 Background

This chapter provides background insights into two critical areas of this thesis. The first is the domain of online advertising. We argue why that industry may have an addressability problem: an issue (re)identifying online website visitors to serve ads. The second part introduces the concept of privacy in data science. Different aspects of privacy and the idea of structured transparency are presented. Lastly, this chapter introduces essential PETs to promote privacy in data science.

2.1 Addressability in Online Advertising

Generally, online advertising is concerned with finding the best ad to serve the user. Here, constraints like the user device and the applicable ad format apply. The advertiser displays ads to users to directly or indirectly influence their actions through clicks, conversions, or brand recognition. Therefore, the advertiser is interested in maximizing the chance of such action by displaying the best possible ad for that specific user [18].

Online Advertising Industry

Ad space online is typically sold via auctions, for which each advertiser selects a set of keywords or websites relevant to its products or services and submits a bit (typically via first-price or second-price auctions). Therefore, this bid represents the estimated utility for displaying the ad to the user and, ultimately, the potential action the user will take as a result of the ad [19].

Online display advertising is a subgroup of online advertising where advertisers embed visual ads on publishers' websites. That ad can take the form of pictures, videos, and text [20]. Due to the context and constraints mentioned before, achieving the advertiser's goal optimally is complex. Therefore, advertisers often use specialized technology. One type of these technologies is called Demand-Side Platform (DSP). DSPs, in a nutshell, help brands and agencies (the demand side) to determine how and at what price to buy ad space [21].

On the other side stand online web publishers. These entities control and curate online websites that sell ad space. While some publishers can use agreed-upon contacts to sell their ad inventory, they often use marketplaces. Here, the Supply-Side Platform (SSP) comes into play. That software enables publishers to sell their ad space by connecting them to DSPs and other similar entities [20]. We can refer to DSPs, SSPs, and similar entities (like ad exchange platforms) as ad-tech companies or third parties [22].

Interest-based advertising is a central component of the online ad landscape. Showing users content based on their interests enables better targeting options for advertisers [22]. It could provide a better ad experience for users, as the ads displayed are more relevant to them. Interest-based advertising predicts users' interest by analyzing their navigated content types and websites. Therefore, it indicates if and what product or service the user is 'in the market for'. Interest-based advertising stands in direct contrast to context-based advertising, which attempts to anticipate solely by the type of content the user is currently browsing [22].

Ad-tech companies enable interest-based advertising by building user profiles. By collecting user data, such as types of websites visited over time, specific user interests can be inferred [22]. A third party can then leverage that profile to enable a good match

between a user visiting a website and the ad displayed. For instance, a person with a browsing history of real estate websites could be assumed to be looking for property to buy. However, such information can never be predicted with absolute certainty. The individual could be, for example, looking to sell their property instead [21]. The third-party cookie is a critical technology that enables this, which will be more thoroughly explained in the section below. User clustering, also known as user segmentation, has been a popular method in the advertising space [18], [23].

Addressability Problem

Recent societal, political, and technological developments have increased the need for privacy-preserving technology [1]. In particular, using third-party cookies to track and analyze online user behavior has evolved from a de-facto industry standard to a practice criticized by many sides. Generally, cookies are small text files saved on the user's browser when they visit a website [24]. First-party cookies are stored directly by the domain that the user visits. The website publisher directly stores data collected using these cookies. Because the organization has a direct prior relationship with the user, it can use that data for communication purposes [25].

On the other hand, third-party cookies are stored by a different domain and are often used for cross-site tracking, retargeting, and serving ads [26]. This practice has been the subject of much discussion[6] [15]. This is because these types of cookies are placed by other parties and are mainly used for analytical and marketing purposes [24], [15]. Three factors are considered to cause the future depletion of the third-party cookie [26].

1. A (legal) environment increasingly focused on online privacy and consent.
2. Browser gate-keeping.
3. The use of ad blockers.

Regulators have raised the question of the legality of the current use of third-party cookies and several regulations pose potential restrictions on it [26]. The GDPR framework, governing the data controller's duties and the data subject's rights when personal information is processed, has collided with that practice. The GDPR applies when personal data (like online user behavior) is processed or analyzed [27], [6]. It also applies when AI is under development or is applied using personal data.

Many legal bases for such a collision can be found [6]. For example, the GDPR requires the user's consent if any personal data is collected, transparency about how it is used, and the ability to revoke the consent given [27]. Furthermore, the principle of fairness in the GDPR dictates that a model can not use sensitive private data (such as race or gender) if it would result in arbitrary, discriminatory treatment.

The GDPR is not the only regulation posing challenges to third-party cookies. For example, the public sector in Norway is subject to the Public Administration Act, where any person has the right to be informed about the circumstances leading to a decision being made. Complying with that act proves to be challenging for many models, as sometimes nobody knows how a result is produced, resulting in a 'black box' [27]. Another example is the California Consumer Privacy Act, which gives users the right to delete, access, or opt out of personal data they share with third parties [26].

[6] blog.google/products/chrome/privacy-sandbox-tracking-protection/

An additional reason for the increased need for privacy-enhancing or preserving technology is a push from the general public in favor of privacy preservation. Individuals are becoming increasingly aware of their rights in data processing [26]. Public scandals in which individual privacy rights were breached (e.g. Cambridge Analytica or Schrems I and II) have contributed to increased privacy sensibility [5], [28].

This has led to market pressure on companies to reevaluate how they handle personal data. Anonymization, the process of removing identifying information from data so it cannot be linked back to an individual, does not solve the issue entirely. One reason is that it is often possible to re-identify the data using other information, primarily given enough information, time, and resources [1], [29]. Large technology companies have evolved to reflect this. In particular, browsers have gone beyond the legal requirements (as of now) to allow users to control their data to gain a competitive advantage [26]. The increasing popularity of ad blockers further disrupts the ability of websites to track user behavior [26].

Impact on Advertising

Data plays a crucial role in marketing and customer experience, as companies need it to provide value to customers [30]. Thus, the phasing out of third-party cookies will significantly impact the online advertising industry. Advertisers currently use them to track users across different websites and target them with ads [15].

This has resulted in what is presently called the addressability problem. Its definition is the capability to uniquely identify a user or a device between several data sets of multiple parties, which can be used for advertising [31]. Advertisers have greatly benefited from collecting and tracking user data on websites (e.g. through third-party cookies) as well as leveraging that information to understand potential customer demographics and interests. Ultimately, that data is used to effectively place ads by targeting users based on their preferences [15]. Advertisers buy ad space on websites with prices determined by total traffic, click-through rate, and conversion rate, among others. These websites include news sites, blogs, and online stores [25].

Due to the depreciation of the third-party cookie, addressability is often not possible in the same form as before. In particular, Puffett and Mussard [26] propose that the advertising industry may face critical problems in several essential advertising functions:

- Frequency capping in the context of displaying ads.
- Audience targeting based on third-party data.
- Retargeting and most forms of dynamic creative targeting.
- Creation of identity linkages by data management platforms.
- Last or multi-touch attribution in the context of marketing attribution.

With the disappearance of third-party cookies, all previously mentioned stakeholders are thus strongly interested in finding alternative solutions for targeting users before [15]. Appendix A.1 details popular alternative online tracking technologies presently in use. A summary of privacy solutions currently developed by technology companies such as Apple and Alphabet is shown in Appendix A.2.

2.2 Privacy in Data Science

It is difficult to establish what exactly defines privacy in the context of data science. For example, Narayanan and Shmatikov [29] show how supposedly anonymized user data (in this case, the Netflix Price data set) can be deanonymized using background knowledge. Here, a correlation is found between a user's private Netflix ratings and their public IMDb ratings. As a result, the individual's anonymized entry in the Netflix data set could be identified. This can expose user information such as name and age or even uncover data as private as political inclination and sexual preferences [29].

2.2.1 Structured Transparency and its Limitations

The concept of structured transparency, introduced by Trask et al. [1], describes the ability to enable public data accessibility while avoiding undesired use. Especially in areas with highly sensitive data (e.g. medical records), questions arise on the complicated trade-off between ensuring high privacy standards and also providing researchers access to the valuable data they need for their work. The provision of decentralized ownership allows for AI models to be trained on data they cannot explicitly access, thus benefiting from the shared information without the downsides associated with direct data ownership [4]. A taxonomy of structured transparency is provided below. It is split into three components (privacy, verification, and flow governance).

Privacy [4], [1], [32]. The first element of structured transparency is privacy. It includes input privacy, defined as the ability to process hidden information without revealing it. Its second element, output privacy, describes the ability to receive information while simultaneously being unable to reverse engineer or infer further information about its input.

Verification [1], [32], [33]. Privacy must be balanced with verification, the second part of structured transparency. This describes the recipient's need to verify the data's validity to trust it. Firstly, there is input verification. It enables one to verify that the information received indeed stems from a trusted source. Also, output verification needs to be mentioned. It allows us to determine whether the results of any information processing within the information process are valid.

Flow Governance [1], [32], [33]. Lastly, flow governance can be seen as a binding element in structured transparency. It is given when each 'stakeholder' of the information used has guarantees that any use of that information will not go beyond the intended or agreed upon use.

Further, the authors expand on the limitations of structured transparency. They include the copy problem, the bundling problem, and the recursive oversight problem [1], [32].

The Copy Problem [1], [33]. The first issue is called the copy problem, which represents the case when once data is shared, the recipient is normally free to copy or misuse it in any other way without immediate consequences or limitations.

The Bundling Problem [1]. The bundling problem describes the fact that it is often difficult to share a piece of information without needing additional data. This is either because the way the data is structured does not allow for individual sharing or be-

cause the authenticity of the data can not be verified without additional context provided.

The Recursive Oversight Problem [1]. We often attempt to tackle the two problems mentioned above by creating organizations or institutions tasked with controlling and overseeing the actors in an information flow. That inherently comes with an additional issue: Who, in turn, holds these organizations accountable? This is called the recursive oversight problem.

2.2.2 Privacy-Preserving Technologies

The problems described above center around the issue of data accessibility: Often, data is present but restricted due to privacy barriers. As a result, many machine learning problems today arise not due to a lack of data per se but due to a lack of access to that data. As a result, privacy-preserving technology is currently relevant and actively researched [1], [3], [2], [4].

Secure Multi-Party Computation (SMPC). An example of PETs for input data privacy protection is SMPC, a cryptographic method. SMPC is concerned with enabling several parties to work together to compute a function while revealing only the output [34]. In SMPC, a secret value x is split into P_i shares. As a result, a party P_i only knows a portion of x, called x_i. All parties can collaboratively calculate the following function:

$$y_1, ..., y_n = f(x_1, ..., x_n)$$

In that way, each party P_i learns the output value y_i and knows its input x_i, without gaining any knowledge about the inputs of other parties [2, p.22]. There have been many different approaches for SMPC protocols. Often, they use a secret-sharing scheme called 'Shamir's secret sharing'. This technique enables parties to share information so no party can reconstruct the secret unless many parties pool together [34].

Homomorphic Encryption (HE). HE is another cryptographic method for privacy preservation, originally introduced by Rivest et al. [35]. It is concerned with an algorithm computing the sum or the product of two messages given the public key and the encryption of the messages, but not the messages themselves [36], [2]. An HE scheme H generally is composed of a set of four functions [2]:

$$H = \{KeyGen, Enc, Dec, Eval\}$$

Here, *KeyGen* refers to key generation. A pair of public and private keys is generated for asymmetric HE, while only a public key is created for symmetric HE. *Enc* stands for encryption, which includes using the public key (in the case of assymetric HE) or the private key (for symetric HE) to generate the ciphertext out of the plaintext [2]. During the decryption *Dec*, the secret key and cypthertext are used to produce the plaintext. Finally, the evaluation *Eval* function uses the plaintext and the public key (in the case of assymetric HE) to create the ciphertext [2].

Differential Privacy (DP). Another relevant example of PETs is DP. It is a way of preserving privacy through obfuscation [2]. As Dwork [37] shows, DP guarantees that adding or removing a single data entry does not (substantially) alter the outcome of any analysis performed on the data set [37]. More formally, a randomized mechanism M

adheres to (ϵ, δ)- differential privacy if provided any two data sets D and D' differing by only one record, and for all $S \subset$ Range (M),

$$Pr[M(d) \in S] \leq Pr[M(D') \in S] \times e^{\in} + \delta$$

where ϵ is the privacy budget and δ is the failure probability [2, p.29]. As a result, minimal risk is incurred when data points are added to a data set concerning the danger of the leakage of sensitive data. In practice, several algorithms ensure this [37]. As noise is added to data, DP inherently carries a utility-privacy trade-off. Generally, this happens by following the function's sensitivity or the exponential distribution among discrete values.

Cummings and Desai [6] argue in favor of using differential privacy for GDPR compliance. In their position paper, the authors detail how many recommender systems and ML models capture individual data entries and thus fail to comply with the GDPR. In contrast, DP achieves compliance while also allowing for the accurate functioning of the models, according to the authors [6].

Researchers at Apple acknowledge that while understanding user behavior can improve their experience, the attempt to do so can compromise user privacy [38]. They designed a system that enables learning at scale while protecting the privacy of users [38]. The use cases here range from identifying popular emojis to common health data types. The paper describes a system that uses local differential privacy, among other techniques, to enable learning. Notably, the algorithm is demonstrated to be scalable and efficient while requiring an opt-in and being transparent [38]. The authors conclude that their paper provides one of the first practical and successful examples of local differential privacy across several use cases. Therefore, it contributes to bridging the gap between theory and practice.

In their seminal paper, Shokri and Shmatikov [4] argue that previously proposed techniques are insufficient for fully achieving privacy. They demonstrate a system that allows several participants to collectively train a Neural Network (NN) model while simultaneously preserving the privacy of individual input data sets. The parties independently train the model on their own data set while only sharing small subsets of it with others. As a result, the model's utility can come to fruition while no sensitive data is exposed. The paper highlights the model's accuracy compared to other methods. The protocol used here is selective Stochastic Gradient Descent (SGD) [4].

McMahan et al. [39] present a new algorithm based on SGD called Federated Averaging (FedAvg). This approach leverages distributed data on mobile devices and then learns a shared model using aggregated updates, which are locally computed. The algorithm is specified in Appendix A.4. This is an example of on-device processing, defined as done 'locally', meaning on a user device, without communicating with external servers [40]. The authors indicate that this method, coined as FL, offers a variety of privacy benefits over techniques solely relying on DP, SMPC, or their combination [39].

3 Federated Learning

This chapter is dedicated to FL, which is among the most popular PETs for ML. It starts by introducing the FedAvg algorithm, providing a definition of FL, and illustrating the cycle of an FL model. Then, the characteristics of FLSs are contrasted with the concept of distributed learning. A taxonomy of FLSs is then presented. Subsequently, PFL is introduced, explaining why a need for personalization exists. Further, a taxonomy of PFL is presented. The last section answers RQ1 by leveraging both previous taxonomies, creating a state-of-the-art table of PFL in advertising.

3.1 Definition, Process, and Life Cycle of Federated Learning

The following properties characterize FL [2]:

- Several parties are interested in building an ML model together.
- Each party's data remains only with that party during the ML training process.
- The parties can exchange the model with each other without compromising the privacy of their data.
- The resulting model is comparable to an ideal model trained in a 'traditional', central way.

A potential loss in performance of the FL model is to be weighed against an attainable increase in privacy, referred to as the privacy trade-off [1]. Therefore, the ability of the individual users to benefit from the ML model without exposing their sensitive data is an advantage of FL [41].

However, FL may also improve model performance. Nowadays, data does not solely reside in one specific location. Instead, it is often distributed among several, often many (user)devices. This can be individual mobile keyboard data on smartphones, satellite data, or autonomous driving data stored in cars. FL can therefore contribute to connecting these 'data silos' [42], [2]. FL thus enables several parties to collaborate and jointly produce a potentially better model. Additionally, the inherently decentralized nature of the technology enables more democratic access to information and analytics without a central, controlling entity [2].

Li et al. [9] highlight the popularity of FL as a research topic, with circa 4,400 related papers in 2022. The authors suggest that an increase in the problems mentioned above, which they term 'data island phenomena', contributes to that rise in popularity. Further, they attribute it to increased privacy protection awareness, which this thesis highlighted before [5].

Zhang et al. [3] illustrate FL using a practical example. The 'traditional' scenario is that data must be sent from devices to a centralized server to understand and predict mobile phone users' behavior. A central ML model can be thus trained on the aggregated data. By contrast, the authors describe a second, federated scenario. There, individual users can collaborate in teaching the ML model with the personal data remaining on their respective devices. FL is, therefore, the building of a ML model using data located on different devices [2].

Hard et al. [43] show an influential application of FL for mobile keyboard prediction. The Google researchers used an on-device learning network for next-word prediction in

Android virtual keyboards. The results are achieved in a scalable way. Further, the FL algorithm is compared to a server-based training method and is described to have a better prediction recall. Thus, data privacy and better user experience can be practically reached, with the paper being one of the first published practical applications of FL to do so [43].

3.1.1 Definition

In their influential paper, Kairouz et al. [44, p. 4f] propose a broad definition of FL:

> "*Federated learning is a machine learning setting where multiple entities (clients) collaborate in solving a machine learning problem under the coordination of a central server or service provider. Each client's raw data is stored locally and not exchanged or transferred; focused updates intended for immediate aggregation are used to achieve the learning objective.*"

Building on the above definition, Li et al. [9, p.3] define an FLS as a system where:

> "*[M]ultiple parties train machine learning models collaboratively without exchanging their raw data. The system's output is a machine-learning model for each party (which can be the same or different). A practical FLS has the following constraints. Given an evaluation metric such as test accuracy, the model's performance learned by FL should be better than the model learned by local training with the same model architecture.*"

This can be more formally expressed and defined as follows (adapted from [2, p.4]).

> N data owners $\{F_i\}_{i=1}^{N}$ want to train a ML model. For that purpose, they use their respective data sets $\{D_i\}_{i=1}^{N}$. The standard method here would be to centrally train the model by collecting all data and saving it to one data server. The resulting ML model can be named M_{Sum}. Here, all data owners $\{F_i$ will expose their respective data sets $\{D_i$ to each other and the server. By contrast, using FL, the data owners will train a model M_{Fed} while not collecting all data $\{D_i\}_{i=1}^{N}$ centrally. Define the following as performance measures: V_{Sum} as the performance measure for M_{Sum}, and V_{Fed} as the performance measure for M_{Fed}. Finally, we can express a potential loss in performance for M_{Fed} using δ, a non-negative real number $|V_{Sum}$ - $V_{Fed}| < \delta$.

3.1.2 FL Life Cycle and Process

But how does an FL architecture come to be, and what does the FL process look like?

FL Life Cycle. One can abstractly define the FL life cycle as follows (adapted from Kairouz et al. [44, p.7]. Note that this high-level process parallels the 'traditional', centralized ML life cycle.

1. Problem identification.
2. Client instrumentation.
3. Simulation testing and prototyping.

4. Model training.
5. Model evaluation.
6. Model deployment.

The initial step is generally made by an engineer developing the model for a particular use case. The problem to be addressed by FL is identified [44]. As mentioned in this thesis, ML models and big data are used to tackle various problems across many domains [1]. For example, let us take the next word prediction on virtual keyboards on smartphones [43].

What follows are two optional steps, depending on the circumstances of the application and deployment of the model. The first is concerned with enabling clients to store the necessary data [44]. In the case of next-word prediction on smartphones, the client device is already capable of this. Yet, it is sometimes necessary to store additional (meta)data for this purpose and, in all cases, some consent for data processing is required if not already given [38], [43]. Then, in the second optional step, the data engineer could simulate the FL model using dummy data for prototyping [44].

The next step is deploying the FL training. Several variations of the model or differently optimized models are trained simultaneously. Then, the different models are evaluated [44]. The final step involves selecting and deploying the chosen model. This includes quality assurance, such as manual and automated testing, as well as A/B testing[7] of the models. The process concludes with a final model roll-out to the client devices [44].

FL Process. Seminal literature agrees that the FL process can be broken down into three different elements [2], [3], [44]:

1. Model initialization
   (a) Client selection
   (b) Global model broadcast
2. Model training
   (a) Client computation of the local modal
   (b) Local model upload
3. Model inference
   (a) Aggregation
   (b) Global model update

The first step is the model initialization, consisting of several sub-steps [3]. Firstly, one needs to select eligible clients based on eligibility requirements. For example, in the case of the next word prediction, a client device is only eligible if it is plugged in, on an unmetered WiFi connection, and not currently in use (for example, at night). This is done to avoid inhibiting the smartphone user's experience through additional computation processes [44], [43]. Then, the initial global model is sent from the server to each eligible device [2], [3]. The client downloads the model, the current model weights, and the training program, for instance, using TensorFlow[8] [44].

[7] hbr.org/2017/06/a-refresher-on-ab-testing
[8] tensorflow.org

Characteristic	Distributed Learning	FL
Data Type	'flat', homogeneous	Varying, heterogeneous
Clients	Nodes of a cluster	Different, independent participants
Data Distribution	Centrally stored	Remains local
Data Access	All clients can read all data	Clients only access own data
Orchestration	Central	Central/decentral, raw data not shared
Communication	Cluster communication	Central hub linking clients
Bottleneck	Computation	Computation or communication
Data Partitioning	Flexible	Fixed
Typical Scale	1-1000	$2 - 10^{10}$

Table 1: FL vs distributed learning (adapted from Kairouz et al. [44, p.6])

Secondly, the device trains the model using its own data, resulting in an individual, local version of the model on each device [2]. The client computes a model update using local data, for example, running SGD as in FedAvg [39], [43]. The device does not need to share its local data [3]. The process can either be coordinated by a central computer or function peer-to-peer, while this example focuses on the former [2]. The local model is uploaded from the client devices to the central server.

Then, the server aggregates the local models into an updated global model [2]. Sometimes, straggler devices are dropped here when a critical mass of model updates has been received to improve efficiency. Also, at this stage, other methods for privacy preservation, such as adding noise, can be applied [44]. This global model is then, again, shared with all devices [3]. This final step, model inference, happens when the model is applied to new data. Ideally, FL is carefully designed so that no partner can reverse-engineer or second-guess the data of any other party [2].

Considerations. The above processes only serve as an illustrative example of FL and do not always apply. Importantly, Kairouz et al. [44] emphasize certain exceptions. With asynchronous SGD, for example, local model updates are immediately applied to the global model without synchronizing with or waiting for the other devices during that protocol. A strict peer-to-peer architecture would also exhibit different characteristics and steps [2].

Further, the processes demonstrate another consideration in deploying FL, especially in practical settings. User experience is pivotal to the technique's success in a real-world environment. For that, firstly, the additional computational effort should not put a strain on the device, impacting its purpose [38], [43], [44]. Secondly, even though models can be trained and updated in real-time, the roll-outs to the users should only happen during the planned deployment phases. This is because creating well-performing ML models is complex, and untested live updates can decrease the usefulness of the product and hamper user experience [44].

3.1.3 Characteristics

Contrasted with a central ML architecture, three characteristics of FL databases can be emphasized [9].

Autonomy. A client engaging in FL is autonomous, meaning it is under autonomous control and can be individually managed by its owner.

Heterogeneity. The databases do not have to be homogeneous and can differ in structure, specifications, and types of data stores.

Varying Data Distribution. How data is spread can vary among the databases, e.g. the partitioning and the frequency of data entries.

An FLS falls into the category of federated systems. These two concepts share the common trait of 'federation', where multiple independent parties collaborate. The first two characteristics from above can, therefore, also be applied. For the third point, the execution distribution in both system types is important, as it strongly influences its performance [9]. For example, FL model updates are narrowly scoped, and aggregation is done as early as possible. This is to minimize data flows and improve computation performance [44]. Yet, data distribution does not vary to the same extent for conventional distributed systems as for FL [9]. Kairouz et al. [44] contrast the typical characteristics of a distributed data center to FL. Some key differences are highlighted in Table 1.

3.2 Taxonomy of Federated Learning

Dimension	**Characteristics**			
Data Partitioning	HFL	VFL	FTL	
Privacy Mechanism	Model aggregation	Cryptography	DP	
Applicable Model	Linear	Tree	NN	
Hetero. Solution	Asynch. comm.	Sampling	Fault Tolerance	Hetero. Model
Client Types	End-user devices	Company servers		
Client Setting	Cross-device	Cross-silo		
Comm. Architecture	Centralized	Decentralized		
Framework	FedAvg	SimFL		
Motivation	Regulation	Monetary	Non-monetary	

Table 2: Taxonomy of FLSs

Characteristic	**HFL**	**VFL**	**FTL**
Data Features	Shared	Varying	Varying
Data Samples	Varying	Shared	Varying

Table 3: Data partitioning in FL

The following section introduces and explains the elements of the different FL. In their survey on FL systems, Li et al. [9] examine the current landscape of the FL technology. The authors propose a taxonomy for its categorization [9]. Similarly, Zhang et al. [3] surveyed FL and its mechanisms while ordering them into different aspects. A potential taxonomy based on these papers is proposed; see Table 2.

3.2.1 Data Partitioning

FL techniques can be classified into three types of data partitioning approaches (see Table 3) [44], [2], [10], [3]. More detailed explanations are shown in Appendix A.3:

- Horizontal Federated Learning (HFL) (also example-partitioned or sample-partitioned): clients share overlapping data features but vary in data samples.
- Vertical Federated Learning (VFL) (also feature-partitioned): clients share data samples but vary in data features.
- Federated Transfer Learning (FTL) (also Hybrid FL): neither samples nor features are shared.

3.2.2 Privacy Mechanism

FL requires privacy preservation, as it is not privacy-preserving by itself [44]. To achieve this, it can employ several PETs. This thesis has elaborated on different PETs in Section 2.2.2. They all can come into play in FL. Additionally, another mechanism called Model Aggregation (ModAg) plays a significant role in the privacy preservation of FLSs [3], [2].

Model Aggregation. ModAg is a common privacy technique for FL. It is concerned with avoiding transmitting original data from the parties by summarizing the model parameters from all participants. These parameters are then used to train the global model [3]. McMahan et al. [42] propose such a technique in the context of FL, where a summarized local model trains a global model in each update.

Private aggregation of teacher ensembles is an example approach using a 'teacher' and 'student' model. Here, the teachers use noisy voting to train the student to predict an output. The student cannot access individual teacher's data or parameters directly. Therefore, the method provides strong privacy guarantees for training data [45]. Various other ModAg techniques exist [3].

Cryptographic Methods. This thesis has introduced the cryptographic techniques of HE and SMPC in the PETs Section [35], [36], [34]. These methods are widely used for privacy preservation in ML. HE enables users to process encrypted data without disclosing the original data. Zhang et al. [3, p.5] refer to it as the "*icing on the cake*" of FL, solving the problem of encrypting private information without hampering the model performance. It further protects the leakage of private party information to the central server. [3]. Similarly, SMPC enables participants to compute a function and share its output without revealing their private inputs.

Differential Privacy. As mentioned in the PETs Section, DP quantifies how much information about a user can be found when analyzing a dataset that includes that user's data. It uses obfuscation in the form of noise or randomness to reduce that revealed information [37]. DP has seen several practical applications in FL [3]. One example of such an application is the privacy-preserving news recommendation model Qi et al. [46] used. In that paper, experiments with realistic data show the ability for the deployment of a news-recommendation model, displaying comparable performance to centrally-trained alternatives, while preserving privacy through DP [46].

3.2.3 Model Types in FL

Characteristic	Cross-silo FL	Cross-device FL
Clients	Different organizations	Many mobile or IoT devices
Data Availability	Almost always	Only a fraction of clients at the same time
Typical Scale	2-100	$2 - 10^{10}$
Bottleneck	Computation or communication	Mostly communication
Addressability	Each client has an identity	Clients cannot be directly indexed
Client statefulness	Stateful, each client may participate in each round	Stateless, each client will likely participate only in one round
Client reliability	Reliable	Highly unreliable
Data Partitioning	Horizontal or vertical	Horizontal
Client hardware capacity	Higher	Lower

Table 4: Cross-silo vs cross-device FL (adapted from Kairouz et al. [44, p.6] and Zhang et al. [3])

Several models types are used in FL. The most common belong to one of three types [3], [10]:

- Linear models.
- Tree models.
- Neural networks.

While a detailed explanation of each model type is beyond the scope of this thesis, it will touch upon them shortly. Zhang et al. [3] and Li et al. [10] provide a more thorough explanation. Linear models include linear, logistic, ridge, and lasso regressions. An upside of using linear models is the relative ease of their implementation [3]. Tree models encompass decision trees and random forests. Arguably, they are efficient to train and rather uncomplicated to interpret [10]. Lastly, NNs are widely used and popular, as they can achieve state-of-the-art outcomes in next-word prediction and image classification. Many FL studies, such as Hard et al. [43], use NNs combined with FedAvg [10].

3.2.4 Heterogeneity Solution

As previously mentioned, heterogeneity is a key differentiator of FLSs. However, it can pose a barrier to their success. Several solutions have been implemented to address this issue [3].

Asynchronous Communication. In FL, real-time communication is a necessity. As with other distributed systems, asynchronous communication is a practical solution to this challenge. For example, one can imagine millions of mobile devices corresponding with the central server in the mobile-keyboard experiment. As consistent exchange of information with all devices at the same time would be impossible, the researchers leverage asynchronous communication as a key answer to that problem [3], [43].

Sampling. In addition, an advantage of FLSs is that not all participants need to partake in the process. Due to the aforementioned scale of many FL applications, this is key for the realistic implementation of FLSs [3]. While being a part of the system is beneficial for model improvement and can therefore be incentivized, it is not needed from all potential parties, especially in large-scale FLSs [2], [47].

Fault Tolerance. The heterogeneity of participants induces lagging or even crashing of individual devices during several process steps. An FLS overcomes this by dropping straggler devices for that part of the process when a satisfying amount of information has been received. This makes model iterations more efficient concerning communication costs [44], [47].

Heterogeneous Model. The data stored by the parties can be described as non-Identically and Independently Distributed (IID). It has been shown to reduce the performance of FL models by up to 55 percent [48]. Thus, handling non-IID data is a central challenge for all FLSs. We address potential solutions in the dedicated PFL section in Section 3.3

While most research addresses device and resource heterogeneity, task heterogeneity can also be challenging for FL systems. Researchers showcase a technique to enable individual end-devices to complete training when performance targets are met. This has been shown to improve model performance [49].

3.2.5 Client Types and Setting.

Clients (also entities, parties, and participants) are the main beneficiaries of FL; hence their collaboration [44]. Kairouz et al. [44, p.6] provide a comparison of client settings in FL, an adapted version of which is shown in Table 4. The table further includes the hardware capacity aspect as mentioned in Zhang et al. [3].

Cross-device. Initially, FL was mainly of interest for mobile and device applications, like the experiments run by Google and Apple on their client-edge devices [38], [50]. This type of FL is generally referred to 'cross-device' FL. Note that cross-device FL, popularly used on smartphones and laptops, can also be executed using other devices. For example, Nguyen et al. [51] deploy FL on off-the-shelf IoT devices, ranging from smart coffee machines and over-door sensors to IoT hubs. Zhang et al. [52] demonstrate using FL on unarmed aerial vehicles. These examples also fall under cross-device FL.

Cross-silo. More recently, systems with fewer but larger participants (like different organizations or companies) have also collaborated to train a shared model. One can name this 'cross-silo' FL application [44]. Yang et al. [2] published a white paper providing several useful theoretical examples for this application. They include companies collaborating to improve their marketing intelligence and medical institutions establishing a data alliance [53].

3.2.6 Coordinating Entity

The literature somewhat diverges when discussing the coordinating entity in FLS. While Kairuz et al. [44] propose that FL is to be distinguished from fully decentralized, peer-to-peer distributed learning, others argue that a peer-to-peer architecture for FL can be

designed. While it would potentially increase security, it would require more computation power [2], [3]. Here, the central server is fully replaced by peer-to-peer communication between the individual clients. Compared to the Hub-and-spoke communication framework mentioned in 1, this architecture attempts to minimize the number of edges for each node to not overload the system [44]. Kairuz et al. [44] propose that, even for a theoretically fully decentralized system, some central authority may still coordinate the learning task, the model, the algorithm used, and more aspects. Yet, the authors argue, such decisions could be solved using a consensus scheme. Yang [2], however, mentions variations of FL using the peer-to-peer distributed learning architecture, citing several such cases in the literature [54], [55]. Nevertheless, the literature agrees that both variations are closely connected and worth exploring further [44].

In practice, a fully decentralized FL setup has been demonstrated. Savazzi et al. [56] deploy FL on a large-scale, dense IoT network without any central coordination entity. The authors' methodology is further successfully validated in a practical experiment using real devices and data for passive body detection in an industrial setting [56].

The coordinating entity can broadly vary between cross-silo FL and cross-device FL. For the first, most organizations would have comparable and performant hardware participating in the process. With sufficient trust from all parties, one of the participating servers can act as the central coordinator. Alternatively, a dedicated server can perform that duty [3]. In many cross-device settings, the clients need a central server acting as the manager, as their hardware capacities are insufficient to perform that assignment [3].

3.2.7 Communication-Computation Framework

The communication-computation framework defines the protocol in which the parties and the coordinator collaborate. This includes the exchange of model parameters and model computation [3]. FedAvg is a commonly used ModAg framework [39]. It functions similarly to the FL process illustrated earlier (see Appendix A.4 for the algorithm).

1. The global model is sent to the clients.
2. The clients update their model with local data.
3. The local models are sent to the server.
4. The global model is updated.

Other frameworks are also proposed. For example, an alternative algorithm called SimFL has been proposed for decentralized FLSs [10]. Here, the participating clients first update the gradients of their local data. Then, these gradients are sent to another party. That party uses the gradients and its local data to update the model. Finally, the model is sent to all other parties. The protocol is repeated using different parties before the final model is created [10], [3].

3.2.8 Motivation of Federation

At this point, one can ask, 'Why should a party participate in an FLS?'. Indeed, encouraging long-term cooperation is a critical factor for the success of FLSs [2]. Here, one can generally distinguish between motivation by regulation and incentives (both monetary and non-monetary) [10]. The first is rather direct: an organization-wide policy or

a governing body could mandate participation in an FLS. While such an order would undoubtedly increase the amount of participants, it would raise questions of legality and morality.

The second type of motivation, incentive-based motivation, is more technically sophisticated. Yang et al. [2] provide a detailed overview of payment systems for data contributions. A DOs contribution could be used to build an ML model that generates revenue which is then shared with them. This includes approaches like profit-sharing games and reverse auctions. The authors propose a fairness-aware profit-sharing framework, which integrates incorporated fairness to promote continuous participation of data-owning parties [2].

Parties can also voluntarily participate in an FLS without direct monetary incentives. Instead, they could partake due to other benefits, like better model accuracy [10]. Examples can include participation for higher personal convenience, such as the next-word prediction of virtual keyboards [43]. One can also imagine that, for example, a hospital could be interested in partaking in an FLS if its disease detection model is improved [10].

3.2.9 Summary of Taxonomy

Dimension	Mutually Exclusive	Collectively Exhaustive
Data Partitioning	Yes	Yes
Privacy Mechanism	No	No
Applicable ML Model	No	No
Heterogeneity Solution	No	No
Client Types	Yes	No
Client Setting	Yes	Yes
Comm. Architecture	Yes	Yes
Comm. Comp. Framework	Yes	No
Motivation of Federation	No	Yes

Table 5: Mutually-exclusive collectively-exhaustive assessment of FLSs taxonomy

The previous sections and Table 2 is an attempt to summarize the different elements of an FLS based on the seminal literature [8], [2], [3], [10]. While this taxonomy is neither complete nor exhaustive, it enables the creation of the state-of-the-art literature review. In this thesis, the categorization of the literature was based on those criteria. In Table 5, the previous dimensions are assessed using the Mutually-Exclusive Collectively-Exhaustive principle by Minto [57].

3.3 Personalized Federated Learning

The previous section characterizes FL, FLSs, and their components. They generally rely on weighted averaging such as FedAvg [39]. While the performance of such protocols is good for homogeneous clients, it faces challenges when dealing with heterogeneity, which is often the case in practice [58]. The following part describes how heterogeneity may create a need for personalization. Further, a taxonomy of PFL is shown.

3.3.1 Need for Personalization

Recognizing that many FLSs operate on heterogeneous, non-IID data, we can observe that performance deterioration may occur [59]. This is because of two different factors:

Client Drift. The first is called 'client drift' and refers to slower convergence of FLSs due to the heterogeneity of data [60]. This considerably slows down the training process and negatively impacts model performance [58].

Local Data Generalization. Also, challenges may arise regarding local data generalization. While the global model is aggregated from local modal updates, that model is not personalized for each participant. This is because, after all, FL uses a global model for all clients [61]. Yet, in many use cases, a tailored model for each user could benefit the utility of the FLS [59]. Indeed, the discrepancy in performance between a global and a personalized model emphasizes the need for personalization in the context of FL [44].

Let us envision a practical example highlighting the need to personalize models. We can do this by taking the use case of next-word prediction for virtual keyboards [43]. After the finalization of the training process, a new global model is deployed on the participating devices. The same prediction pattern will be used across all user devices. However, each texting pattern has nuances and cultural patterns, making them unique. We can argue that having a tailored model for each user would improve prediction performance [59]. Thus, we can ask: "*How can we leverage the global model in FL to find a 'personalized model' stylized for each client's data?*" Dinh et al. [61, p.1].

3.3.2 Taxonomy of PFL

The literature proposes many solutions to that question. For example, Zhao et al. [48] improved model performance for non-IID data by training a global model with fractions from all device information. Similarly, Liang et al. [62] propose an algorithm that learns a local representation for each party while also training a global model. Also, Smith et al. [47] suggest a novel optimization method considering communication cost, stragglers, and fault tolerance, resulting in significant speedups of FLSs.

But how can we structure these approaches in groups? Several attempts to categorize such forms of PFL can be found in the literature [63], [13], [59]. We found that among these papers, Tan et al. [59] provide the best overview for this thesis. This is because, firstly, it has been recently published. Being up-to-date with the consistently changing academic status quo is key, especially for a research field that developed as recently as in the past few years [44]. Secondly, the paper provides the only comprehensive and holistic survey of PFL specifically, to the best of our knowledge [59]. That PFL taxonomy is presented in Figure 3. Note that both Deng et al. [63] and Mansour et al. [13] provide a similar but less exhaustive categorization and should be viewed as augmenting this taxonomy.

Tan et al. [59] generally distinguish between two overarching strategies for PFL. They are marked as 'personalization strategies' in Figure 3. Analogously to the two challenges of FLSs mentioned earlier, the two strategies are separated into solving the problems of heterogeneity and lack of personalization, respectively. The following section will explain each personalization strategy and provide a recent academic paper to illustrate it.

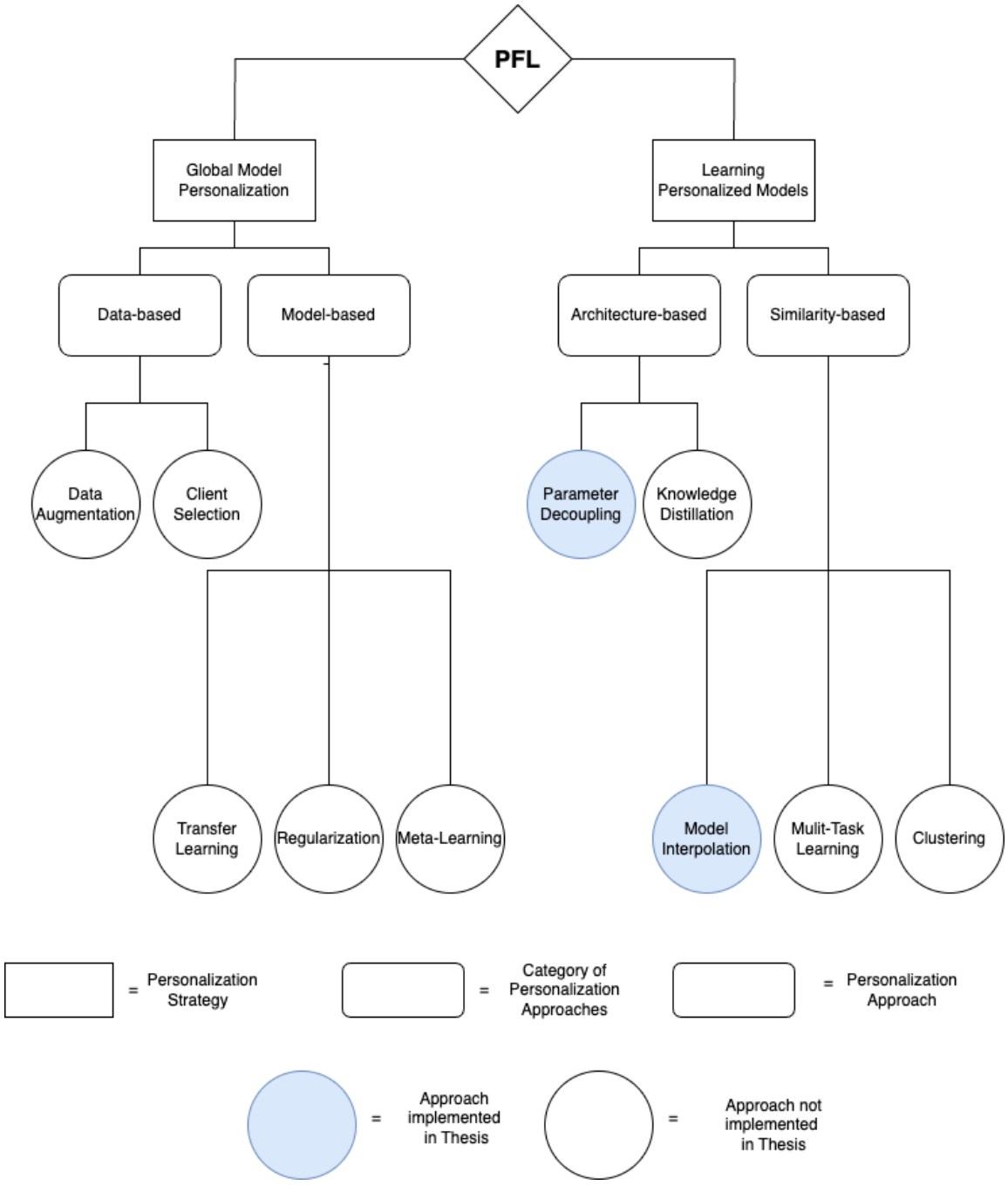


Figure 3: PFL taxonomy as in Tan et al. [59, p.2]

3.3.3 Global Model Personalization

The first strategy is called 'global model personalization' and is concerned with solving data heterogeneity. The averaging of results is negatively impacted due to client drift, and this is addressed by personalizing the trained global model the participants. Firstly, a global FL model is trained analogously to the FL process. Then, that model is personalized for the clients by training it on the client's data [59]. Global model personalization distinguishes between data-based and model-based personalization approaches.

Data-based. The first approach category, 'data-based', aims to reduce data heterogeneity in the participating party's datasets. Two types of variations exist in this approach category. Data augmentation is the first [59]. Here, data sharing makes the individual participant's data more homogeneous. This can happen, for example, through distributing a subset of data among all clients. Zhao et al. [48] demonstrated that sharing five percent of the global data among all clients can increase model accuracy by circa 30% [48]. The taxonomy refers to the other variation of data-based approaches as 'client-selection'. It modifies the strategy by which participants are picked for every federated training round. For example, a paper formalized a participant-selection approach for every learning round to optimize accuracy and minimize the number of rounds required [64].

Model-based. The last category of personalization approaches in the context of 'global model personalization' is model-based personalization. In contrast to the previously mentioned data-based approaches, it is not concerned with allocating data among the clients but with creating a well-performing global model [59]. Firstly, let us examine Transfer Learning (TL) as an approach. It aims to reduce the gap between the global and local models by transferring knowledge from a source domain to a target domain. A popular example of this is the 'FedMD' framework. Each model is first trained on public data, which creates a global model. Then, a local model is trained on the private data. In the final step, a personalized model is created by refining the local model with the global model's knowledge through TL [65].

Regularization is the second 'model-based' personalization approach. This technique is concerned with reducing local loss by limiting the impact of local updates, resulting in improved convergence stability and a better generalization of the global model [59]. For example, an algorithm called 'SCAFFOLD' uses variance reduction to correct for the client drift of FedAvg. The authors prove that the algorithm successfully works with heterogeneous clients and converges in fewer rounds [60].

The final approach in this category is meta-learning. That method is often described as 'learning to learn' and refers to how systems improve their ability to adapt to domains by using prior knowledge. Fallah et al. [66] leveraged meta-learning to reformulate the goal of FedAvg: instead of learning a global model that performs well on most tasks, the aim is now to create a good initial global model that works well on heterogeneous tasks after adaptation to local data [66]. Their approach is shown to outperform FedAvg when it comes to personalization.

3.3.4 Learning Personalized Models

'Learning personalized models' is the second strategy, standing in contrast to the first. Instead of adapting and personalizing the global model, it learns individual personalized models. The taxonomy groups these approaches into 'architecture-based' and 'similarity-based' [59].

Architecture-based. Architecture-based approaches attempt to personalize models by individually tailoring the model configuration. The first way to achieve that is parameter decoupling. It functions by separating the model parameters used in the local and the global model. This technique allows for a degree of personalization by enabling specific components of the model to adapt to the unique characteristics of local data while main-

taining a shared model structure for standard features learned across all clients [59]. An example here is Federated Learning with Personalization Layers (FedPer), as in Arvazhagan et al. [67]. The authors separately train the base layer on global data and the top layers on local data.

Alternatively, Knowledge Distillation (KD) can be used for architecture-based approaches. In the context of NNs, it is concerned with transferring knowledge from teacher to student models. Several studies demonstrate the utility of this practice in the context of VFL, where the global model teaches the local model while performance is improved [68], [11], [69].

Similarity-based. The last category of personalization approaches is called 'similarity-based'. Here, each participant learns a personalized model, while personalized models have similarities, as their participants are alike [59]. Model interpolation combines the global and local models into a personalized model for each participant. The goal is to find the optimal mixing parameter for each client [63]. For example, Deng et al. [63] introduce model interpolation using Adaptive Personalized Federated Learning (APFL) and demonstrate its effectiveness.

Smith et al. [47] argue that multi-task learning, the second approach of similarity-based personalization, is naturally suited to solve problems in FL. Multi-task learning aims to train a model that jointly performs several connected tasks. One can consider each client a separate task in FL. Using Federated multi-task learning can leverage domain-specific knowledge and thus improve performance through personalization.

Clustering is the final approach to be discussed. Scientists at Meta AI propose a novel approach called 'group personalized FL', which falls under that category. For inherently partitioned groups, each client can leverage the knowledge from its respective group and improve its model. This is done by fine-tuning the global model through another FL process over the client group [70].

3.4 State of the Art of PFL in Advertising

> *"In [...] targeted advertising delivery, the heterogeneous content of data samples and models at different nodes is an important issue. However, most FL frameworks overlook this problem, resulting in defects in practical applications"* [12] p.5.

The previously mentioned challenges of FL are equally relevant in advertising. Additionally, this is a significant challenge for CTR prediction models in FL. As a result, the training of such models is often slowed down, and accuracy is suboptimal [58].

PFL, as in the previous section, is used to tackle these FL challenges. With PFL being an emerging research topic, narrowing down on advertising reduces the amount of existing research substantially [44]. RQ1 concerns a state-of-the-art literature table for PFL in advertising. For that purpose, the following criteria are used:

1. The publishing date needs to be from 2016 or later, taking MacMahan et al. [39] as a pivotal turning point for FL research.
2. The paper needs to be application-focused, not merely providing theoretical proof.
3. Further, it must tackle a problem in advertising or be firmly placed in that domain.
4. That problem must be related to or be concerned with CTR prediction.

5. Lastly, the experiment must be performed on real-world data, not dummy data.

The state of the art is summarized in Table 6 and Table 7. To create these tables, both the taxonomy of FLSs (see Table 2) and the taxonomy of PFL (Figure 3) were consulted. Only the most relevant aspects were included to ensure a focused, state-of-the-art summary. As a result, each column serves the purpose of making the tables a comprehensive yet narrow overview.

Both tables examine the same ten distinct academic papers. Table 6 describes the more essential characteristics of the paper in question and the type of personalization used. Table 7 is concerned with explaining the results of the studies. Both tables will be discussed in the following section. The aim with this is not to thoroughly describe every paper, as a more concise approach will be taken. The literature will be analyzed and put into context on a higher meta-level. Similarities and differences will be evaluated. It is thus an attempt to summarize the state of the art as of the writing of this thesis within the five constraints provided above.

3.4.1 Set-Up

Author and Date	**Dataset Source**	**Use Case**	**Privacy Mechanism**	**Model**	**Personalization**
Ren et al. (2022) [68]	Criteo, Taobao	CTR	ModAg, HE	Regression	KD
Li et al. (2022) [69]	Criteo, Game-Ad (Tencent)	CTR, CVR	ModAg	NN	KD
Li et al. (2022) [11]	Criteo, Avazu	CTR	ModAg	NN	KD
Wan et al. (2023) [71]	Criteo	CTR	ModAg, DP	NN	KD
Su et al. (2023) [12]	Avazu, Taobao	CTR	ModAg	NN	KD, TL
He et al. (2022) [72]	Avazu	CTR	ModAg	NN	Meta-Learning
Liu et al. (2022) [58]	Tmall	CTR	ModAg	NN	Meta-Learning
Rong et al. (2022) [73]	Algo.qq (Tencent), Taobao	CTR	ModAg	NN	Meta-Learning
Wei et al. (2023) [74]	FedAds (Alibaba)	CVR	ModAg, DP	NN	Data Augmentation
Hejazinia et al. (2022) [75]	Taobao	CTR	ModAg, DP	NN	Clustering

Table 6: Set-up and personalization in state of the art of PFL in advertising

The set-up of the ten papers is described in this section and Table 6. Notably, we can find similarities regarding the origin of the dataset used. Firstly, they strongly overlap in their dataset, as displayed in Figure 4. Secondly, they are all produced by large advertising companies or retailers. Thirdly, they are all of Chinese origin, except Criteo. On one hand, this allows for some comparability among the papers. On the other hand, it naturally leads to a lack of diversity regarding the datasets used in the state of the art.

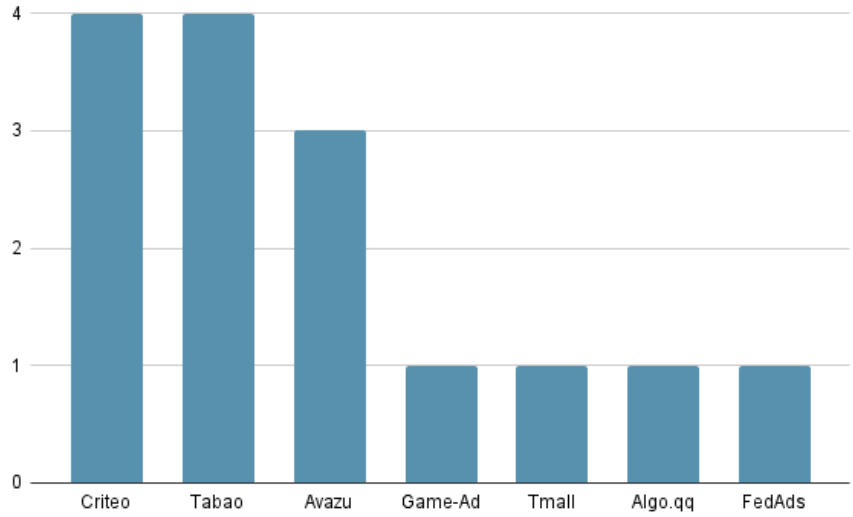


Figure 4: Datasets used in state-of-the-art papers

Not only, as Wei et al. [74] argue, is there a lack of standardized datasets, but authors often also adapt the few datasets publicly available. Often, these are not federated by themselves, so the federation is then simulated. This is the case for all but one dataset, the one used in [74]. This brings challenges to the comparability of results. Further, Wei et al. [74] propose the large-scale dataset they collected from Alibaba called 'FedAds'. This dataset is genuinely federated, as one part belongs to a publisher and the other to an advertiser. Thus, they propose this dataset and a set of metrics as a benchmark for further studies in the realm of PFL in advertising. Both [72] are referred to in that benchmarking paper [11].

Interestingly, nine of the ten papers have CTR prediction as their goal. The other is concerned with conversion rate (CVR). CVR is closely related to CTR but is more complex to estimate because it is rarer, among other factors [21]. Also, only a few use cases of FL in advertising have been found while creating the state-of-the-art tables, such as facilitating private data exchange between advertisers [76]. CTR prediction seems to be a valuable use case and somewhat easy to facilitate [21]. This combination might be one of the reasons for the popularity of CTR in the state of the art. Considering the above, it becomes clear why NNs are predominantly used, with only one paper using a regression model instead [68]: NNs are a popular and successful form of CTR prediction in advertising [77].

Looking at the privacy mechanism used as in Figure 5, it is unsurprising to see ModAg used by all, as it is an inherent property of FLSs [39]. Notably, only four papers go beyond solely relying on ModAg, with three using DP and one using homomorphic encryption. He et al. [72] explore privacy in the context of PFL, being one of the first papers to directly address the issue of label leakage in VFL. It provides a thorough privacy analysis, including risks and potential solutions. The work, however, does not directly deal with advertising but tests its novel approach on an advertising dataset [72].

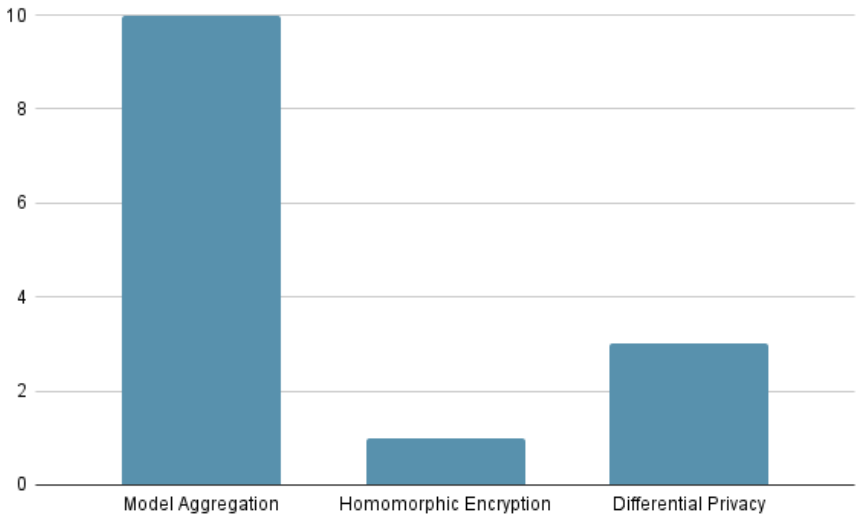


Figure 5: Privacy mechanisms used in state-of-the-art papers

3.4.2 Personalization Approach

Comparing the state of the art (see Table 6) to the taxonomy, we find that both personalization strategies, 'global model personalization' and 'learning personalized models', are included. Also, all personalization categories, namely data-based, model-based, architecture-based, and similarity-based, are represented at least once in the table. However, the distribution of personalization approaches is somewhat uneven (see Figure 6). While KD is used five times and meta-learning three times, transfer learning, data augmentation, and clustering have one representative each. The approaches are further examined in the following sections.

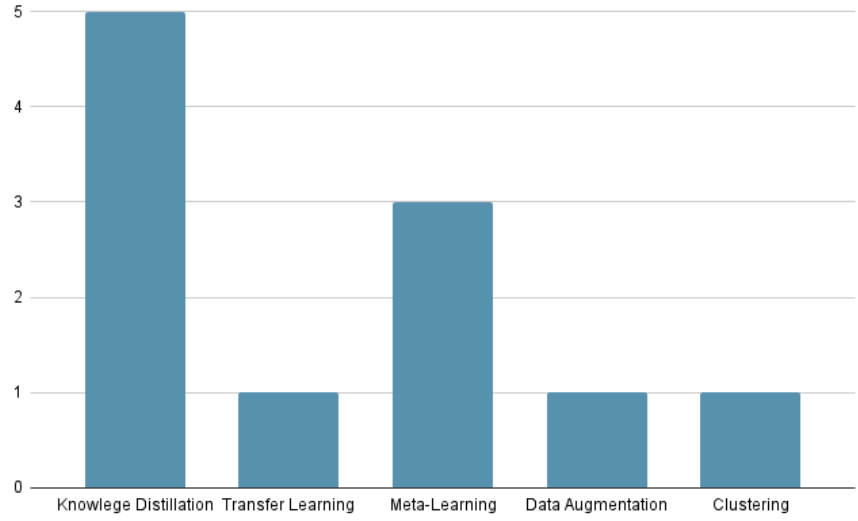


Figure 6: Personalization approach used in state-of-the-art papers

Knowledge Distillation. Ren et al. [68] can be considered a pioneer work when it comes to PFL in advertising through KD [11]. Recognizing the limited ability of VFL systems to infer for non-overlapping data samples, the authors propose KD as a solution. Firstly, the global model, the teacher model in this paper, is trained on combined datasets. Then, the local model, acting as the student model, is taught by the global model. However, the local model only has access to its local dataset and the label information provided by the teacher. In that way, a better inference of the local model without direct access to global data is possible [68]. While an advertising dataset is used in this paper to demonstrate the effectiveness of the novel approach, it is not specifically targeted toward advertising as a use case. Further, this is the only paper to train a regression model and deploy HE for additional privacy [68].

This circumstance changes in the second paper to be presented [69]. By contrast, it addresses challenges typical to advertising, like a lack of data labels and easy deployability on advertising systems (as systems often differ in computation power or network power, for example). The first issue is managed by utilizing both labeled and non-labeled data available through self-supervised learning. The second challenge is addressed by minimizing the system requirements of the approach. This is done by facilitating the knowledge of federated models to be distilled into an independent local model, limiting the need for cross-party operations [69].

A similar focus on real-world advertising deployability is provided in Li et al. [11]. However, the paper considers the entire sample and feature space. A practical distillation framework for feature correlation is proposed for that purpose. The authors name their approach 'vertical semi-FL', a hybrid between full FLS and a non-federated model. Both [69] and [68] are included in that definition. That term can be interpreted as another way of expressing PFL, although that name itself is not directly mentioned [11].

Focusing less on the domain of advertising, Wan et al. [71] emphasize privacy preservation in their work. Using a double-distillation approach, knowledge can be transferred despite the restraint of limited data overlap. Importantly, it explicitly targets the problem of potential privacy leakage by limiting the amount and capacity of server communication and introducing an offline training scheme. This improves privacy and reduces communication costs. Further, DP was introduced to strengthen privacy.

Transfer Learning. In their approach, Su et al. [12] introduce two properties unique to this state-of-the-art overview. It is the only paper presented here that uses two different personalization techniques and leverages TL. Their framework, named Fed-TLKD, is specifically designed for advertising. The framework is divided into two modules. The first uses maximum average difference TL to select valuable information from sample data and improve the global model's performance. In particular, TL is applied to enhance the adaptability of the global model to various ad-targeting datasets, like those containing unique text data. Firstly, an optimal aggregate model is computed, considering each client's unique data distribution. This ensures that the global model is customized to the specific needs of each participant. The second module utilizes KD. Here, local models are trained by a global (teacher) model. Thus, both approaches are used to achieve personalization.

Meta-Learning. Similarly to Li et al. [69], He et al. [72] utilized self-supervised learning to use non-labeled data and improve model performance. Yet, the key difference is that He et al. [72] leverage meta-learning to combine cross-party and local self-supervised

learning methods to achieve that. It adapts the learning process based on the data it encounters across parties. As a result, the model better understands how to train itself on different datasets.

The approach proposed by Liu et al. [58] also falls into the meta-learning category for model personalization. In contrast, the method does not use self-supervised learning. Instead, it optimizes ModAg and adjusts learning rates dynamically across participants. Meta-learning is used here to adapt the teaching of the global model consistently. Therefore, client heterogeneity takes is taken into account.

The approach outlined in Rong et al. [73] involves users, advertisers, and ad servers. Initially, a base model is trained through FL, with user devices optimizing the model locally based on their local data and advertisers aggregating the weights of ads. Following this FL, the model leverages the embeddings developed during this phase to initiate a meta-learning process. The model further uses a gradient-based meta-learning approach for fine-tuning. The framework can dynamically adjust to new or less frequent data by initializing the meta-learning with the previously learned federated embeddings. It emphasizes the improvement of the model's accuracy and relevance in real-world advertising scenarios.

Data Augmentation. The only paper leveraging a data augmentation in the state-of-the-art tables is introduced in [74] and is called Diffu-AT. It addresses the challenge of incorporating unaligned samples. Typically, these samples would not be usable due to missing features, which is solved here by generating these missing features through a diffusion model. This approach augments the data, creating synthetic features that fully utilize unaligned samples. Incorporating these synthetic features makes more comprehensive training of the federated model across diverse data sets possible.

Clustering. The last paper to be introduced and the only one leveraging Clustering for personalization is Hejazinia et al. [75]. It uses the 'federated ensemble learning' approach to tackle the challenges associated with recommendation systems. The method entails grouping client devices by distinctive characteristics and then training multiple model versions on these clusters. As a result, each cluster's model specializes in the traits of the group. After training, the models are aggregated on a central server. Here, the models are used for inference. The final prediction is created by passing the output of each cluster model into the ensemble aggregation layer.

3.4.3 Results

Table 7 presents the results of the state-of-the-art literature examined in this thesis. All papers used at least one model to compare and benchmark their results. Often, a local or global model is used. This helps compare FL with non-FL approaches. We can emphasize that FL increases privacy due to ModAg and outperforms them in prediction accuracy, as seen in the AUC or accuracy metrics. Other FL models are also used in the papers for comparison. The comparison is important to highlight the need for personalization in advertising and is done by using ads datasets with popular domain constraints, like:

- Non-overlapping data samples (e.g. [68], [71]).
- A lack of data labels (e.g. [69]).
- Device and system heterogeneity (e.g. [69], [71]).

Author and Date	Comparison Model(s)	Performance Metric	Privacy Metric	Result
Ren et al. (2022) [68]	Local	AUC	-	Better
Li et al. (2022) [69]	Local, FL	AUC	-	Better
Li et al. (2022) [11]	Local, FL, [68], [69]	AUC, RelImp	-	Better
Wan et al. (2023) [71]	Local, Global, FL, PFL	Accuracy	-	Better
Su et al. (2023) [12]	Global, PFL	AUC, Accuracy, Loss	-	Better
He et al. (2022) [72]	Local, Global, FL, PFL	AUC, F1 Score	CAP	Better
Liu et al. (2022) [58]	Global, FL	AUC, Logloss	-	Better
Rong et al. (2022) [73]	FL	AUC, Logloss	-	Better
Wei et al. (2023) [74]	Local, FL, PFL, [69], [72]	AUC, NLL	$\Delta LeakAUC$	Better
Hejazinia et al. (2022) [75]	FL	AUC, Accuracy	-	Better

Table 7: Results of state of the art of PFL in advertising

- Increased need for privacy (e.g. [71]).

Lastly, other PFL methods are also sometimes used to compare the papers. These contributions enable a more coherent and comprehensive overview of the existing methods. Interestingly, two papers in the state-of-the-art table compare their models to ones used in other works appearing in it [11], [74].

Metrics. Apart from accuracy and AUC, F1 scores are used to assess the performance of the models. Loss is another popular metric, also reflected in Negative Log-Likelihood (NLL) and logloss [74]. Perhaps the most unique performance measure in the papers is called RelImp, used in [11]. It is a popular metric in the industry and is used to compare a model to a baseline model [78]. More formally, they define it as [78, p.10]:

$$\text{RelImp} = \frac{\text{AUC(model)} - 0.5}{\text{AUC(baseline)} - 0.5} \times 100\%$$

In comparison, only two papers measure privacy directly through metrics. He at al. [72] quantify the privacy-utility trade-off using Calibrated Averaged Performance (CAP). They define it as [72, p.10]:

"*For a given protection mechanism $\mathcal{M}_\lambda$ with a protection strength parameter λ and an attacking mechanism A, the Calibrated Averaged Performance (CAP) for a given privacy-utility trade-off curve is defined as follows*,

$$CAP(\mathcal{M}_{\lambda\in\{\lambda_1,\ldots,\lambda_v\}}, A) = \frac{1}{v}\sum_{\lambda=\lambda_1}^{\lambda_v} U(\hat{G}_\lambda) * E(\tilde{D}_\lambda, D), \tag{1}$$

where $\hat{G}_\lambda = \mathcal{M}_\lambda(G)$ is the VFL model protected by $\mathcal{M}_\lambda$, $\tilde{D}_\lambda = A(\hat{G}_\lambda, D)$ is the data recovered by the attacking mechanism A from $\hat{G}_\lambda$ given the private data D as input, $U(\cdot)$ measures the main task utility (e.g., accuracy) of a given model, and $E(\cdot)$ measures the distance between recovered data $\tilde{D}_\lambda$ and original data D."

The last privacy metric mentioned here is named ΔLeakAUC. It helps assess the ability to defend from a label inference attack [74]. The metric does so in three steps [74, p.9]:

1. Create a baseline model without any defense approach. Calculate AUC given labels inferred by an attacker as $\text{LeakAUC}_{\text{base}}$.

2. Learn a new model with a defense approach and compute the AUC for the inferred model analogously to 1 as $\text{LeakAUC}_{\text{exp}}$.

3. Calculate ΔLeakAUC as follows:

$$\Delta\text{LeakAUC} = \frac{(\text{LeakAUC}_{\text{exp}} - \text{LeakAUC}_{\text{base}})}{\text{LeakAUC}_{\text{base}}}$$

.

4 Implementation

This chapter examines how PySyft and ML are implemented to answer RQ2 and RQ3. It begins by explaining PySyft and its practical deployment. Both the dataset and data modifications are subsequently described. Further, the implemented ML model is discussed. Then, the metrics used to assess the model's performance are introduced. Also, the comparative metric of RelImp is shown. The chapter concludes by detailing the four different types of architectures implemented.

4.1 PySyft

Yang et al. [2] and Bonawitz et al. [8] provide overviews of implementation frameworks for the simulation of FL. These are often developed as open-source initiatives [8]. They facilitate the creation proof-of-concept applications, accelerating the development of FLSs. Below is a short selection [2], [8]:

- TensorFlow Federated[9].
- Federated AI Technology Enabler[10].
- PySyft [17].

4.1.1 Introduction

This thesis implements PySyft, which was proposed by Ryffel et al. [17]. PySyft was selected for several reasons:

- It has a considerably large and active open-source community and is continuously developed further[11].
- It is somewhat easy to use and implement for people familiar with Python and data science (e.g. it can be used via Jupyter Notebooks) [14].
- It enables the simulation of FL, which is useful for proof-of-concept applications [2].

In their work, they train a ML model without providing direct access to that data. They denote a distributed approach where models are trained on decentralized data residing on remote devices, such as mobile phones [17]. Instead of bringing the data to the machine training the model, the code for training the model is sent to the devices. The remote devices receive the current model and subsequently train that model on their data. Afterward, the changes made to the model are summarized and sent back, with the training data of the device remaining local throughout the entire process. Also, the implementation averages the updates of several remote devices and does not store individual updates, thereby improving both model quality and privacy. According to the authors, the system's most significant drawback is the long training time [17].

Researchers behind Ryffel et al. [17] and Trask et al. [1] created OpenMined[12], an open-source software community with the goal of achieving structured transparency through

[9] tensorflow.org
[10] fate.fedai.org
[11] github.com/OpenMined/PySyft
[12] openmined.org

PETs. Its vision is to provide access to data without the negative consequences that might come from misuse, such as intellectual property theft and discrimination.

In particular, OpenMined is leading the development of PySyft. Ziller et al. [14] describe PySyft as an open-source library that facilitates the creation of secure, private ML models. It integrates with deep learning frameworks like PyTorch. The primary objective of PySyft is to enable the use of privacy-preserving techniques, making them accessible through commonly used interfaces and tools. Additionally, it allows for incorporating additional PETs, such as DP and SMPC [14].

Functionality. PySyft's functionality is centered around tensors, which extend those from existing ML libraries (e.g., PyTorch or TensorFlow), adding further functionalities(e.g. the ability to encode). This happens via chains where each connected tensor adds a particular feature, making PySyft modular [14]. So-called workers, composed of a server and a client, own the tensors and can use them for computations. Workers communicate with each other via messages to perform remote executions, which can be cross-platform (e.g. Python to JavaScript). These remote computations are executed through pointers [14].

The architecture of PySyft is illustrated in Figure 7. The central idea is that the data and code are separated. The data is stored on the side of the DO, which has control over it. The Data Scientist (DS) is a peer node that can request certain computations on the data stored with the DO [79]. Any operation on the code by the DS on the DO's data is wrapped into a Remote Procedure Call (RPC) and sent to the DO. A remote computation of any object or library mapped to the abstract syntax tree is possible. In addition, agents are used to verify the identity and credentials of both actors [79].

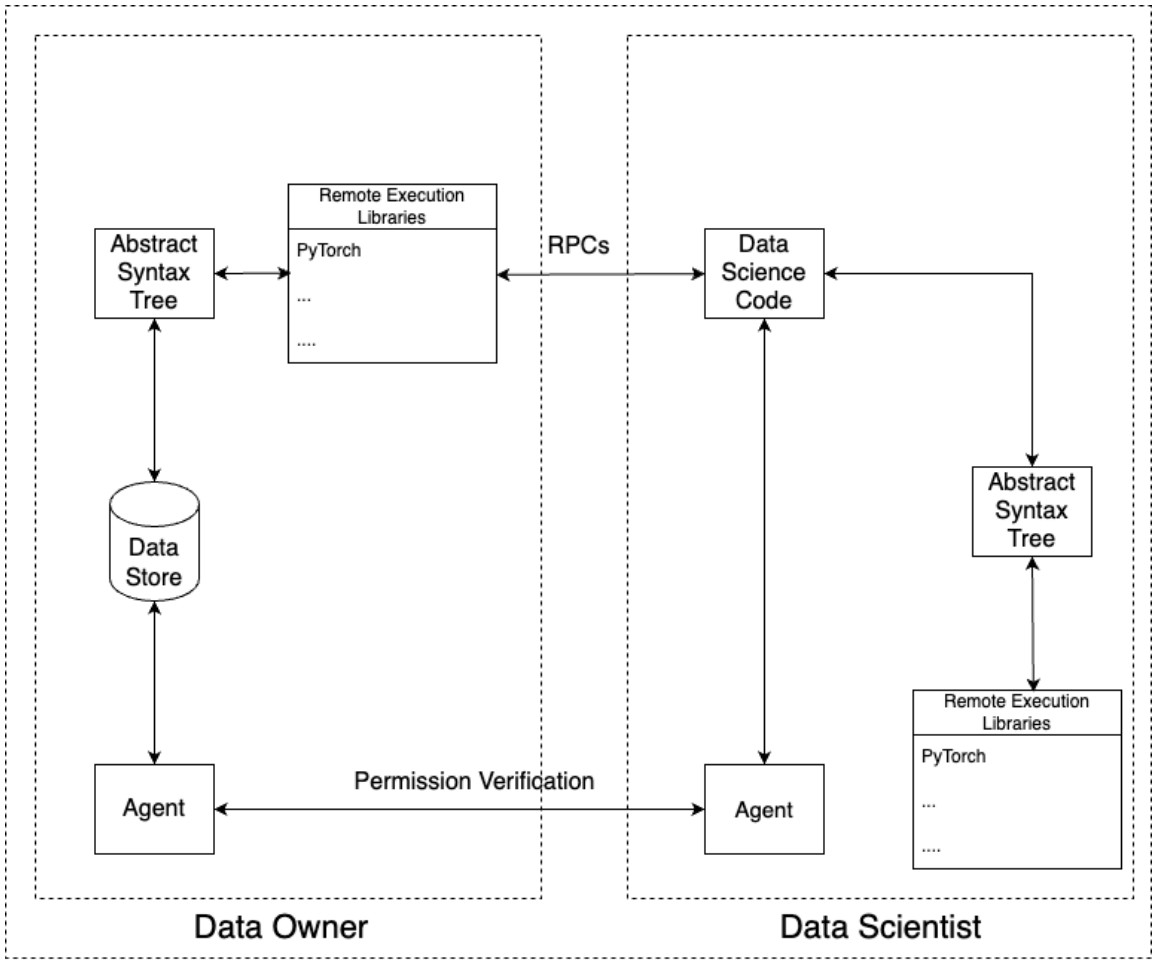
The DO can, in turn, review both the code and the request result. They can further approve or deny it manually or via an automatic policy. If approved, the result is returned to the DS and can be encrypted [79]. In that way, a computational operation was performed on data without revealing it. The architecture aims to provide a transparent layer for the two actors, enabling privacy preservation without much change to the original code [79].

4.1.2 Deployment in Thesis

This subsection describes how PySyft was installed and implemented for this thesis.

Installation. For the installation, the macOS tutorial for PySyft was used[13]. In short, it allows deploying a PySyft domain on the local machine. The process includes installing and activating Python packages like Conda, pip, and jupyterlab. HaGrid is a command line tool that speeds up the deployment of PyGrid, which provides peer-to-peer networks of DOs and DSs using PySyft[14]. Syft Version 0.8.4 was used for all implementations. For replication purposes, it is to be noted that many other versions of Syft may not work. The implementation was done via Jupyter Notebooks and can be accessed here[15].

[13] openmined.github.io/PySyft/install_tutorials/osx_11_5_1.html
[14] see Footnote 11
[15] github.com/PalexGoneGit/PFLAdsPySyft



PySyft

Figure 7: PySyft architecture adapted from Hall et al. [79, p.3]

Passing of Model Weights in the Implementation. First, the model weights are extracted, and noise with a standard deviation of 0.01 is added. Then, both DOs login to their PySyft domain, convert their weights into Syft ActionObjects, and set them in their domain (Figure 8). The DS then defines a function to access the weights and requests it to be executed (Figure 9). After approval by the DOs, the DS receives and averages both model weights, creating the global weights (Figure 10).

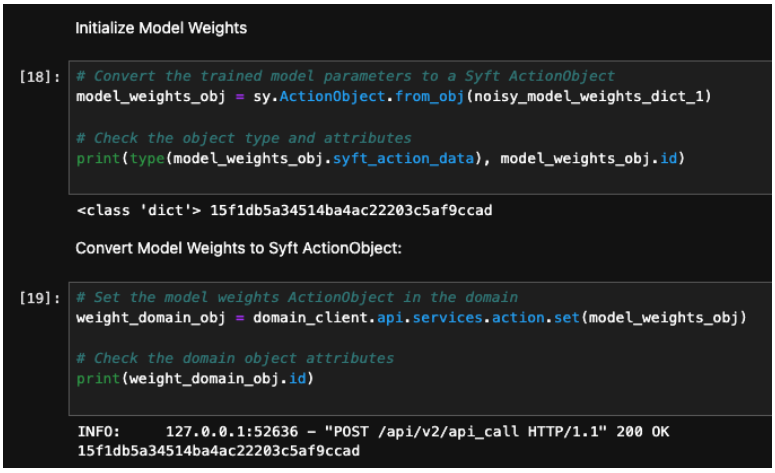


Figure 8: Initialization and conversion of model weights in PySyft

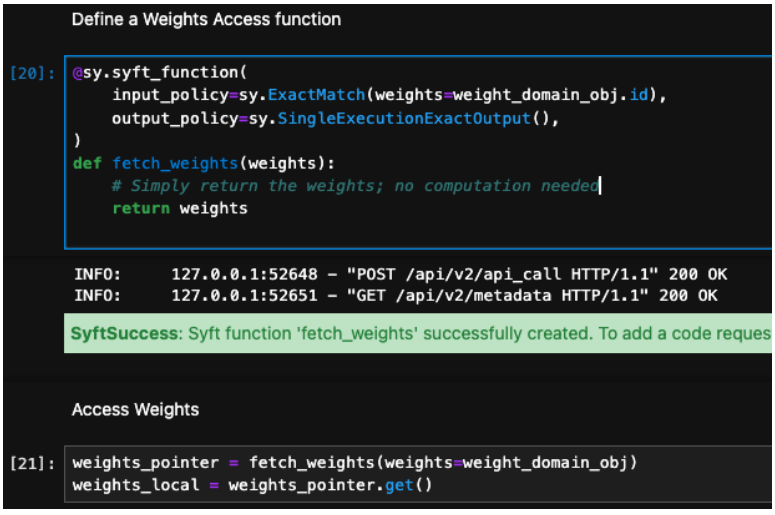


Figure 9: Definition of weights access function and access of weights in PySyft

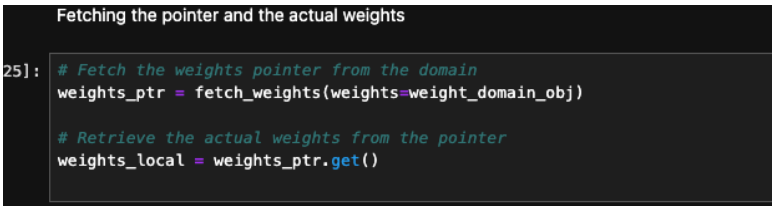


Figure 10: Fetching of pointer and weights in PySyft

4.2 Dataset and Model

As described in Section 1.1, the implementation highlights a practical use case. Two media companies are switching from a centrally trained ML model for CTR prediction to a federated setup. The CTR prediction problem has been validated as central to the advertising industry in Chapter 2 and Section 3.3.

4.2.1 Original Dataset

To simulate the CTR prediction problem, the 'display advertising challenge' dataset by Criteo was selected[16], available for download here[17]. It was chosen as it is:

1. Fitting to the research scenario,
2. Of sufficient size for ML,
3. Generally available and free to use publicly, and
4. Among the most used datasets in the state of the art (see Figure 4).

The source provides a 'training' and 'test' dataset. Only the first contains a label, so the second is useless for this thesis and discarded. The remaining dataset contains feature values and click feedback for display ads and has been collected by Criteo over seven days. It includes 1,264,729 rows, each representing an instance of an ad being served. There are 40 columns, with the first being the label, indicating whether an ad has been clicked (1) or not (0). The positive and negative examples have been sub-sampled at different rates to reduce data size. The other 39 columns are divided into 13 features in the form of integer values and 26 categorical features. The latter has been hashed for privacy purposes. The semantics of all features are undisclosed, while some features may have values missing. With the raw CSV file loaded and 'not applicable' values replaced with zeros, an excerpt of the dataset can be seen in Figure 11. By dividing the total negative labels by the total number of rows, we can calculate the accuracy of guessing the label to be 0 for every instance, which we can refer to as a baseline (74.63%)

4.2.2 Dataset Modifications

While the original dataset is popular in the state of the art and suitable for ML, it is relatively homogeneous. This thesis aims to exhibit PFL, so reflecting the need for personalization is key. As argued in Section 3.3, PFL is particularly useful for heterogeneous, non-IID data. Therefore, six dataset variants are used, displayed in blue in Figure 12.

Firstly, the variations are split into the entire dataset ('full dataset') and a variant using one-tenth of all rows. The full dataset is sorted chronologically and thedividedit in half. For the 10% dataset, the rows are sampled randomly. All further variations exist analogously for both datasets.

Two data modifications, 'factor 2' and 'random noise', are deployed and explained below. When the dataset is not modified, it is called 'regular'. These three variations exist twice for the 'full dataset' and '10% dataset'. This results in six distinct datasets (shown in blue in Figure 12).

[16] kaggle.com/competitions/criteo-display-ad-challenge
[17] tianchi.aliyun.com/dataset/144733

```
[2]: # Load the dataset and fill the nan with 0
     columns = ['label', *(f'I{i}' for i in range(1, 14)), *(f'C{i}' for i in range(1, 27))]
     df = pd.read_csv('/Users/alex/Thesis/New Dataset/train.txt', sep='\t', names=columns).fillna(0)
     df
```

[2]:

	label	I1	I2	I3	I4	I5	I6	I7	I8	I9	...	C17	C18	C19	C20
0	0	1.0	1	5.0	0.0	1382.0	4.0	15.0	2.0	181.0	...	e5ba7672	f54016b9	21ddcdc9	b1252a9d
1	0	2.0	0	44.0	1.0	102.0	8.0	2.0	2.0	4.0	...	07c540c4	b04e4670	21ddcdc9	5840adea
2	0	2.0	0	1.0	14.0	767.0	89.0	4.0	2.0	245.0	...	8efede7f	3412118d	0	0
3	0	0.0	893	0.0	0.0	4392.0	0.0	0.0	0.0	0.0	...	1e88c74f	74ef3502	0	0
4	0	3.0	-1	0.0	0.0	2.0	0.0	3.0	0.0	0.0	...	1e88c74f	26b3c7a7	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1364724	0	0.0	2	0.0	2.0	41171.0	90.0	0.0	2.0	47.0	...	07c540c4	ab194a92	0	0
1364725	0	0.0	3	7.0	10.0	3912.0	400.0	2.0	17.0	104.0	...	e5ba7672	698d1c68	0	0
1364726	0	0.0	1	48.0	0.0	17375.0	0.0	0.0	0.0	0.0	...	e5ba7672	7b06fafe	2f4b9dd2	a458ea53
1364727	0	0.0	0	49.0	2.0	16610.0	0.0	0.0	6.0	0.0	...	1e88c74f	c21c3e4c	efa3470f	a458ea53
1364728	1	1.0	106	1.0	3.0	152.0	33.0	0.0	0.0	0.0	...	0	0	0	0

1364729 rows × 40 columns

Figure 11: Excerpt of dataset

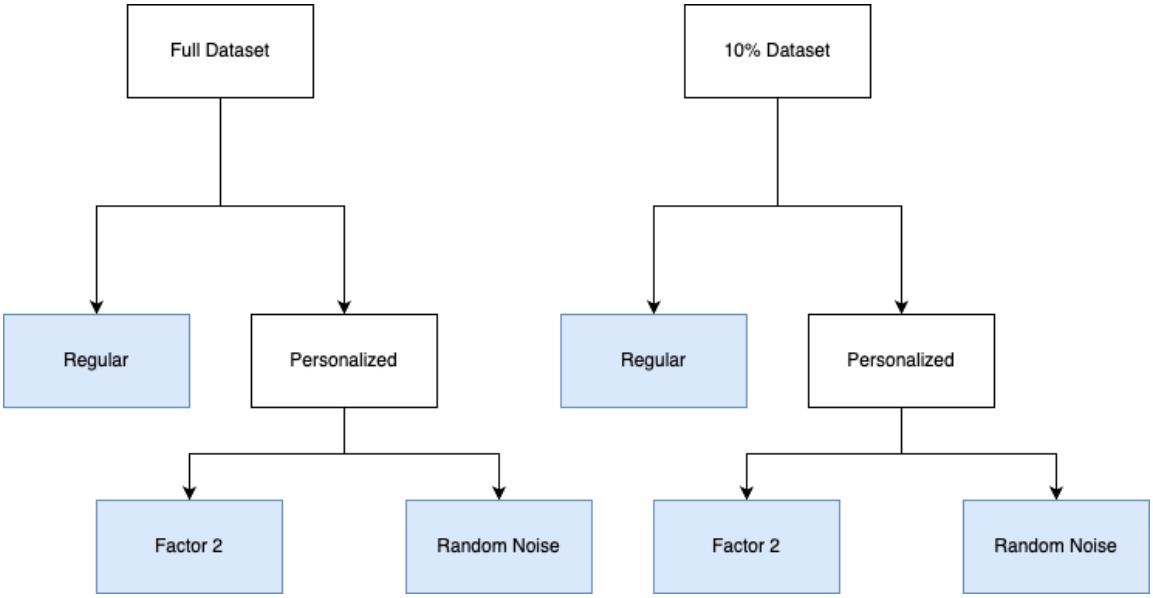


Figure 12: Six variations of dataset

Factor 2. The 'factor 2' datasets are created as follows. Firstly, the dataset is split at half of the rows. The first part has its integer values amplified by multiplying them by two. At the same time, the second part has its integer values dampened via division by two. The split datasets' mean integer values are contrasted in Figure 13.

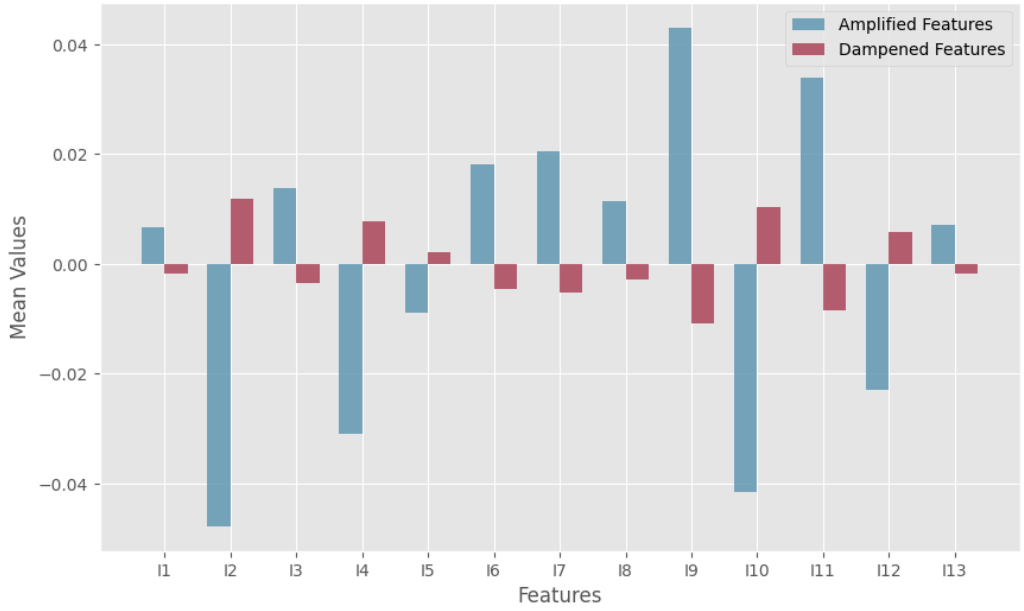


Figure 13: Mean integer values of factor 2 dataset

Random Noise. The other method of artificially instilling heterogeneity into the datasets is the addition of random noise. As in the 'factor 2' method, the dataset is split in half. Then, random noise is created with a mean of 0 and a standard deviation of 0.5. The shape of the noise array matches the shape of the slice. Finally, the random noise is added to the first slice. The second slice is kept as is. This introduces randomness and variability to the data. Figure 14 illustrates the resulting mean integer values.

Data Slices. For all implementations besides 'central,' data halves (data slices) are created for separate DOs. This is analogous to the 'before' and 'after' scenarios (Figures 1 and 2). For the 'factor 2' method, DO1 receives the amplified, and DO2 gets the dampened data slice. In the case of 'random noise', DO1 receives the altered data slice, whereas DO2 obtains the unmodified half. There is only one data owner in the 'central' architecture, so the data slices are merged again.

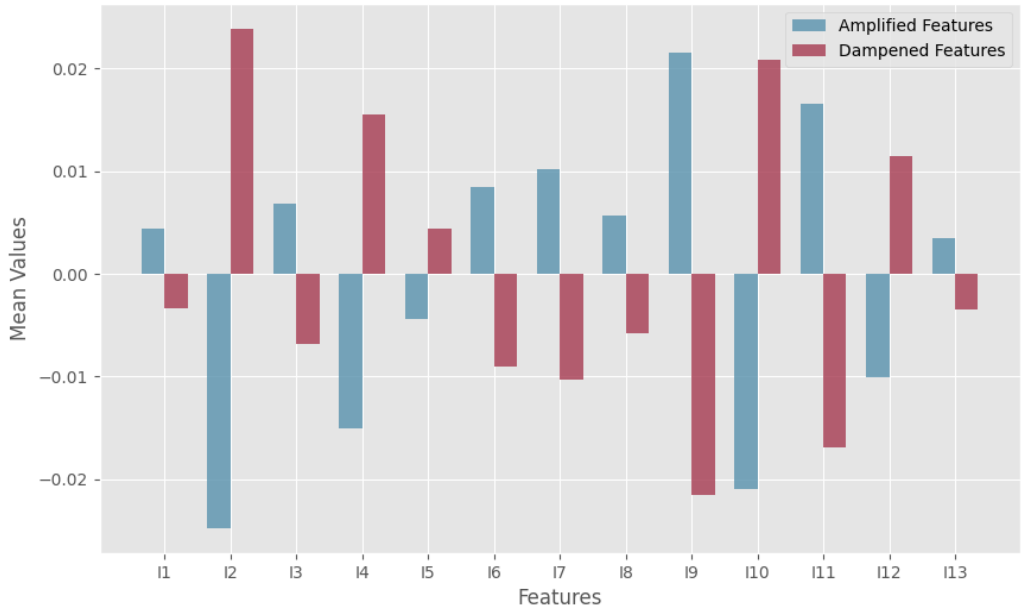


Figure 14: Mean integer values of random noise dataset

4.2.3 Model and Metrics

For all dataset variations and DOs, the ML model is identical and based on the 'Deep Learning Recommendation Model', proposed by researchers at Meta [80]. The model is designed explicitly for recommendation tasks, CTR prediction in particular. It is also well suited for the dataset, as it handles numerical and categorical features in an efficient and scalable manner [80]. A TensorFlow implementation of the 'Deep Learning Recommendation Model', submitted publically for Criteo's 'display ad challenge', was used as a code template[18].

As the dataset includes both numerical and categorical features, the latter must be encoded into numerical for the model to function. Their cardinality is partly very high (up to over 400,000). Known as the 'curse of dimensionality', using some encoding techniques here can cause slow computation [81].

This is solved the following way. First, each categorical feature is mapped to dense vectors. The numerical features are also transformed, resulting in a dense representation [80]. An 80/20 training/test split is used. Additionally, 20% of the test data is used for validation to improve model tuning. Second, the 'Deep Learning Recommendation Model' is defined. The bottom layer prepares both feature types to function together [80]. It is initialized with two hidden layers, while the top layer uses three. The top (or output) multilayer perceptron post-processes the results and sends them to a sigmoid function to receive a probability. For the compilation, keras.optimizer.Adam is used in its legacy version due to hardware constraints. Binary cross-entropy loss is used here. The learning rate is defined at 0.0001, five epochs are used, and the batch size equals 128.

[18]https://kaggle.com/code/egordm/deep-learning-recommendation-model-dlrm

Prediction Quality Metrics. The performance metrics used for analysis are accuracy, AUC, and loss. They were selected for their utility in reflecting the quality of the prediction. Also, they were chosen to build upon the state-of-the-art papers analyzed in Figure 7, as they are popular there. They are used to illustrate the model's prediction quality directly, hence each is referred to as a Prediction Quality Metric (PQM) or collectively as PQMs.

RelImp. Further, RelImp, as explained in Section 3.4.3, is calculated where applicable. We can leverage RelImp to measure the relative improvement of the federated model compared to the local model of each implementation. It was chosen for its unique ability to do so, its appearance in the-state-of-the-art table (see Figure 7), and its popularity in the industry [11].

We can use RelImp to see if the FL or PFL architecture causes the model predictions to improve. If RelImp is above 100%, the architecture enhances the model predictions. That statement can be reversed if RelImp is below 100%. All metrics are presented in Tables 11 - 16 in Appendix A.7. They are summarized, analyzed, and discussed in Chapter 5.

4.3 Architectures

The architectures were selected for the following reasons. Firstly, they are fit to answer our RQs. As RQ2 is concerned with the implementation of FL, using FedAvg is key. It is among the most seminal and popular versions of FL [39], [44]. Further, RQ3 and RQ4 are related to the implementation and comparison of PFL, so two PFL architectures were selected.

The reasons for selecting the PFL architectures are now discussed. Firstly, they represent two different personalization approaches (parameter decoupling vs model interpolation), as in the PFL Taxonomy in Figure 3. Implementations of these approaches are yet to be found in the state-of-the-art literature (see Table 6).

Secondly, they express two different categories of personalization approaches: architecture-based and similarity-based. These categories of personalization are indeed already represented in the state-of-the-art table. The approaches are thus yet to be implemented in the state-of-the-art literature, but similar approaches have already been implemented. Therefore, this thesis attempts to introduce both methodical standardization and heterogeneity to enable a comparison.

Finally, we turn to practical reasons. Both PFL approaches are demonstrated to work with NNs, which is the type of model used [63], [13]. Also, these architectures (in a simplified form) were found to be realistically implementable, considering the technical limitations of this thesis. These limitations are further discussed in Section 6.2.

4.3.1 Centralized

The centralized architecture is the most straightforward and is representative of the before scenario, as seen in Section 1.1 and Figure 1. The implementation of the central model is depicted in Figure 15. The centralized approach encompasses a model being trained on training data, and then testing that model on testing data to evaluate its prediction quality.

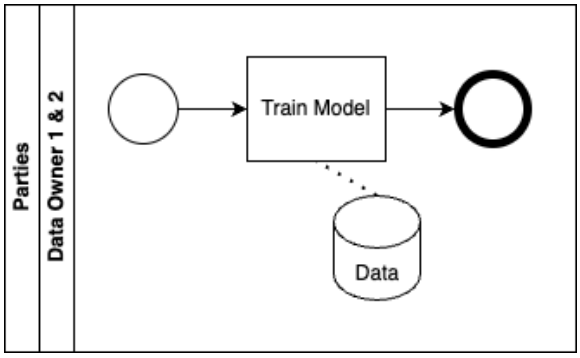


Figure 15: Implementation of 'central'

4.3.2 Federated Averaging

FedAvg, the first federated architecture, is displayed in Figure 16. Here, model weights are passed as described in the Section 4.1.2. Both DOs then use the new global weights to train the model on their local data. Finally, the performance metrics are calculated. This performs ModAg using a simplified version of the FedAvg algorithm (see Appendix A.4).

4.3.3 Adaptive Personalized Federated Learning

Mansour et al. [13] propose that the constraints of FL give rise to three different approaches for algorithms for model interpolation for personalization:

- User clustering, by training a model for subsets of users.
- Data interpolation, by training a model on a combination of local and global data.
- Model interpolation, by training a local and global model and using their combination.

Deng [63] argues that the first two of these approaches tend to have practical drawbacks, as privacy is reduced, e.g. when meta-features are collected from participating parties [63]. Building and expanding upon the third schema, the authors propose APFL. This method belongs to the category of similarity-based approaches and is concerned with model interpolation (see Figure 3).

In APFL each participant trains its local model first. The personalization happens by mixing the global and the local model weights to create a personalized model for each client. This combines the global model's robustness with the local model's personalization. That algorithm is described in Appendix A.5.

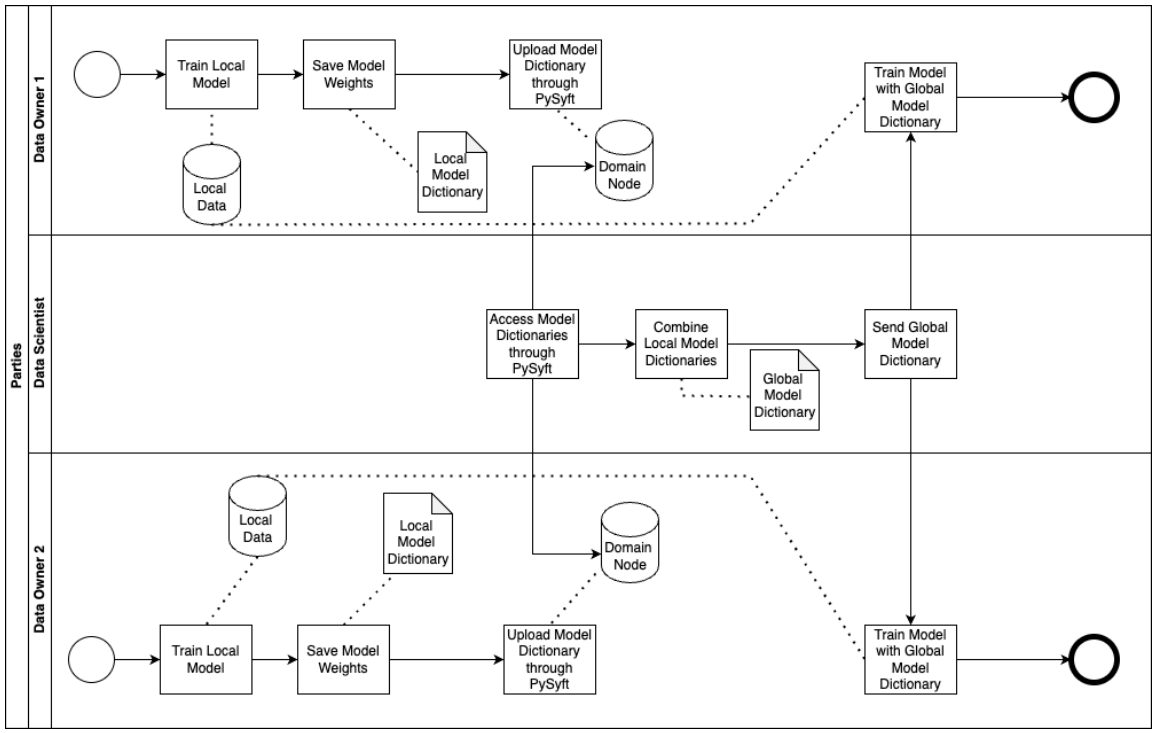


Figure 16: Implementation of FedAvg in PySyft

The APFL architecture is implemented in this thesis (see Figure 17). Initially, it follows the same approach as the FedAvg implementation. This changes after the global model dictionary is sent to the DOs. Instead of directly applying the global weights to a new model, they are combined with the previous local weights into a new, personalized model dictionary. In this case, α, the mixing parameter, is set to 0.5. Finally, the new, personalized model dictionary is used to deploy the model and the metrics are calculated.

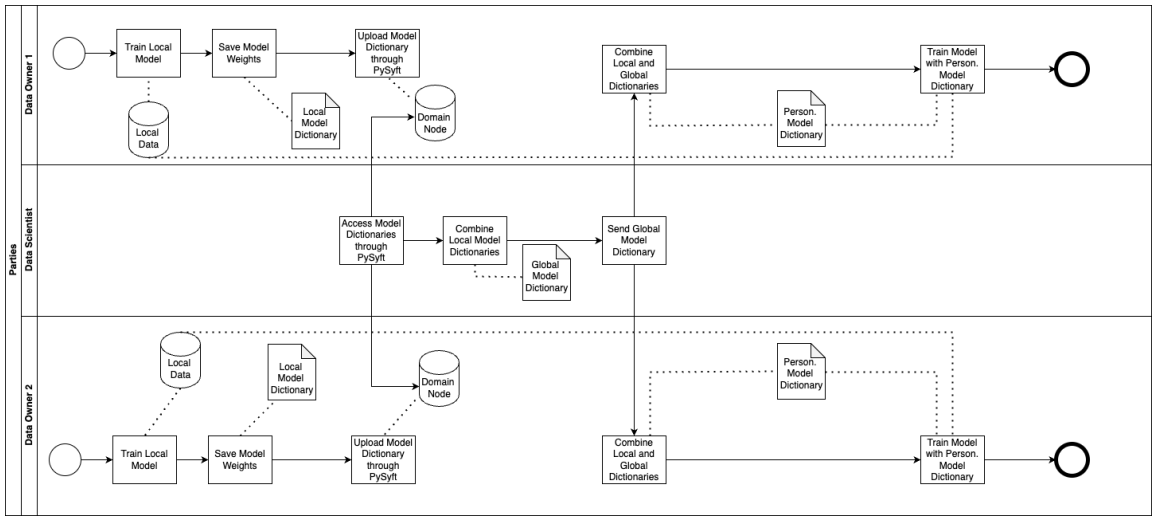


Figure 17: Implementation of APFL in PySyft

4.3.4 FedPer

The second PFL approach implemented in this thesis relies on local fine-tuning. Arvazhagan et al. [67] introduce a PFL algorithm with personalization layers for NNs. This method belongs to the category of architecture-based approaches, more specifically, parameter decoupling (see Figure 3).

That method, called FedPer, trains the model base layers using FedAvg. The personalization layers are trained only on local data. Please see Appendix A.6 for the specific algorithms. For an overview of the FedPer implementation in the thesis, please see Figure 18.

First, the DOs' base layer and top layer models are individually trained on their local data. Only the bottom layer model dictionaries are passed through PySyft in the same way as before, and the DS combines them into the global model dictionary. Each DO retrains the base layer model with the global model dictionary. In the final step, each DO's bottom and top layers are merged into a personalized model, whose performance metrics are then calculated.

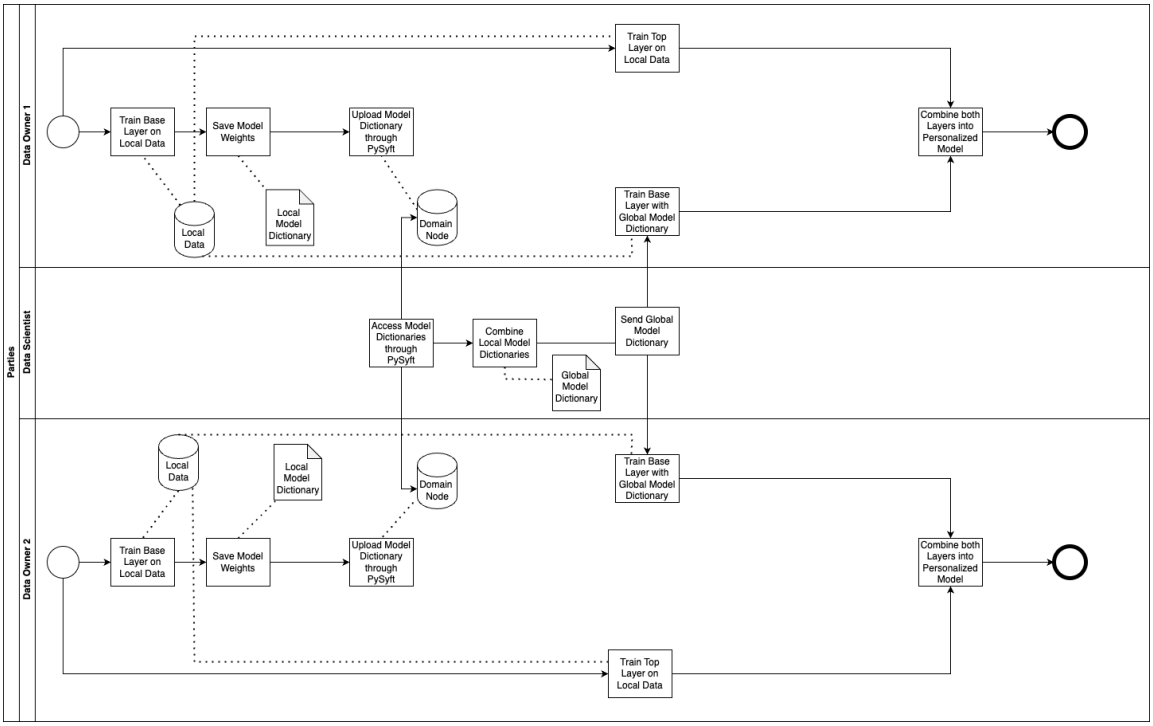


Figure 18: Implementation of FedPer in PySyft

5 Results, Analysis, and Discussion

The following chapter describes the implementation results. Then, a thorough analysis of the outcomes structured by architecture follows. Lastly, the discussion section uses that same structure to interpret the results.

5.1 Results

There are a total of six pairings of dataset variants and architectures. They are shown starting Table 11 until Table 16 in Appendix A.7. Each table displays a unique dataset variant with all architecture combinations used. Since all architectures besides 'central' have two variants, one for each data slice as in the DO1 and DO2 split, both results are shown for each architecture variant. The respective data slice result is indicated with a 1 or 2 at the end of the architecture name column (e.g. FedAvg1 or FedAvg2), leading to seven columns for each table. The results for each metric - architecture - dataset combination are thus shown in the respective cell of the appropriate table.

Performance Metrics. Performance metrics include the PQMs and RelImp. For all architectures besides 'central', all performance metrics were calculated using the final federated model (not the first, local model). Due to its nature, the 'central' architecture only has one model output and no federated model. Therefore, the RelImp - Central combination is always empty, as no relative improvement calculation is possible for that architecture. For the remaining architectures, RelImp is calculated as specified in the formula in Section 3.4.2. For ease of reuse and accessibility, the performance metrics are aggregated in table form in Appendix 27 and as a CSV file on GitHub[19].

Performance Metrics Comparison vs Central. To promote ease of readability and comparability, the metrics are shown in one comprehensive table, namely Table 8. Each architecture-DO-Dataset-modification pairing receives a column for that purpose. Instead of showing the results directly, they are contrasted with the results of the respective central implementation, which we use as a baseline. Note that RelImp for all 'central' implementations is denoted as 100%. This lets us see the results directly while comparing them to the baseline.

Summary Statistics. Additionally, summary statistics are exhibited. For this, the performance metrics for each architecture were aggregated across all datasets (including the results of DO1 and DO2 for each implementation). Their median, Maximum (Max), Minimum (Min), and Standard Deviation (SD) values are displayed in Table 17 until Table 20 in Appendix A.7. Further, the summary statistics are graphically shown in Figure 30 until Figure 32, also in Appendix A.7.

Summary Comparison Statistics. The summary statistics are used to compare the architectures among each other to help answer RQ4. PQMs for the FL and PFL architectures are compared to the 'central' version. The RelImp metric is contrasted with the results of the FedAvg architecture. These comparisons are illustrated in Figure 19 until Figure 28. Their tables are in the Appendix A.7, from Table 22 until Table 26.

[19] github.com/PalexGoneGit/PFLAdsPySyft

Architecture	DO	Dataset	DataMod	Accuracy	AUC	Loss	RelImp
FedAvg	1	Full	Regular	-0.0092	-0.0101	0.0174	-0.73%
FedAvg	2	Full	Regular	-0.0026	-0.0040	0.0139	-0.43%
APFL	1	Full	Regular	-0.0056	-0.0073	0.0150	1.25%
APFL	2	Full	Regular	-0.0089	-0.0106	0.0187	-0.37%
FedPer	1	Full	Regular	-0.0316	-0.0528	0.2240	-8.11%
FedPer	2	Full	Regular	-0.0216	-0.0432	0.2088	-7.20%
FedAvg	1	Full	Faktor 2	-0.0048	-0.0054	0.0188	1.29%
FedAvg	2	Full	Faktor 2	-0.0126	-0.0146	0.0220	-2.85%
APFL	1	Full	Faktor 2	-0.0038	-0.0052	0.0119	0.53%
APFL	2	Full	Faktor 2	-0.0047	-0.0068	0.0162	0.98%
FedPer	1	Full	Faktor 2	-0.0231	-0.0393	0.1480	-4.48%
FedPer	2	Full	Faktor 2	-0.0268	-0.0417	0.1405	-7.83%
FedAvg	1	Full	Random Noise	0.0113	0.0121	-0.0325	-0.62%
FedAvg	2	Full	Random Noise	0.0182	0.0210	-0.0446	0.49%
APFL	1	Full	Random Noise	0.0117	0.0122	-0.0327	-0.23%
APFL	2	Full	Random Noise	0.0181	0.0183	-0.0409	-0.80%
FedPer	1	Full	Random Noise	0.0176	0.0208	-0.0448	23.85%
FedPer	2	Full	Random Noise	0.0269	0.0307	-0.0588	15.46%
FedAvg	1	10%	Regular	0.0147	0.0146	-0.0357	-3.10%
FedAvg	2	10%	Regular	-0.0022	0.0014	-0.0072	-2.34%
APFL	1	10%	Regular	0.0049	0.0084	-0.0274	2.44%
APFL	2	10%	Regular	-0.0013	0.0010	-0.0125	-0.94%
FedPer	1	10%	Regular	-0.0170	-0.0261	0.1483	25.15%
FedPer	2	10%	Regular	-0.0098	-0.0195	0.1806	36.16%
FedAvg	1	10%	Faktor 2	-0.0143	-0.0093	0.0082	0.87%
FedAvg	2	10%	Faktor 2	0.0063	0.0083	-0.0156	-1.27%
APFL	1	10%	Faktor 2	-0.0058	-0.0109	0.0103	0.58%
APFL	2	10%	Faktor 2	0.0027	0.0025	-0.0068	0.53%
FedPer	1	10%	Faktor 2	-0.0329	-0.0399	0.1676	51.58%
FedPer	2	10%	Faktor 2	-0.0197	-0.0267	0.1645	36.63%
FedAvg	1	10%	Random Noise	-0.0008	-0.0016	-0.0132	0.16%
FedAvg	2	10%	Random Noise	0.0160	0.0146	-0.0322	-0.03%
APFL	1	10%	Random Noise	0.0026	-0.0034	-0.0035	-0.22%
APFL	2	10%	Random Noise	0.0149	0.0121	-0.0340	-0.44%
FedPer	1	10%	Random Noise	-0.0330	-0.0435	0.1634	25.39%
FedPer	2	10%	Random Noise	-0.0136	-0.0266	0.1569	34.48%

Table 8: Full metrics comparison vs 'central'

5.2 Analysis

The analysis describes the results and is organized into four subsections, one for each respective architecture. Each time, the PQMs are discussed first, then RelImp (if applicable). Figures are provided and referenced for illustration. Finally, the discussion presents an interpretation of the findings. Subsections for each architecture again separate that section.

5.2.1 Central

The central model forms a standard of comparison for the other architectures. Generally, we can see that all three metrics follow the same pattern for each version, i.e., no outlier metric stands out against the others.

PQMs. While the 'full—regular' and 'full—factor 2' implementations display the best PQMs (see Tables 11 and 12), the two worst performing pairings are '10%—random noise' and '10%—regular' (see Tables 16 and 14). Their maximum differences are small: 76% vs. 74.30% accuracy, 83.47% vs. 81.70% AUC, and 0.5255 vs. 0.5565 loss. The downsized dataset versions display worse PQMs compared to the 'full' variants. One exception exists, in which the '10%—'factor 2' version outperforms the 'full—regular' counterpart (see Table 15 and 12).

5.2.2 FedAvg

Let us now turn the FedAvg implementation and its results. It is the first federated architecture to be examined.

PQMs. Comparing the FedAvg implementations to their central counterparts yields varied results. All 'full—regular' and 'full—factor 2' FedAvg models are worse than the central implementations (see Table 8). Yet, all 'full—random noise' FedAvg models outperform their central equivalents. Interestingly, all FedAvg variations of the '10%' dataset outperform 'central' when it comes to one DO, but not the other (see Table 13).

Looking at the summary statistics, we can see that all median PQMs are slightly worse than the ones of the central architecture, as illustrated in Figure 19 and Figure 20. Yet, the differences for FedAvg are only within the range of two decimals (see Table 22).

However, the best FedAvg implementation beats the maximum 'central' performance by 1.76 and 1.86% for Accuracy and AUC, respectively see Figure 21 and Table 23. The highest loss of FedAvg is 1.31% lower than the 'central' maximum (Figure 22). The minimum loss for FedAvg is 8.73% lower than the 'central' equivalent, as in Figure 24 and Table 24.

RelImp. For the RelImp metrics, we see that using FedAvg never substantially improves the performance of the personalized model. We often see almost no change in the performance of the local and personalized model, with RelImp varying between 99 and 101%. Sometimes, a decline in model performance can be seen (see Table 8). This is the case for FedAvg2 in the 'full—factor 2' and the '10%—factor 2' implementations, with 97.15% and 98.73% respectively (see Tables 12 and 15). The worst performance deterioration is displayed in both '10%—regular' implementations, with a RelImp metric as low as 96.90%, the worst of all FedAvg versions (see Table 14).

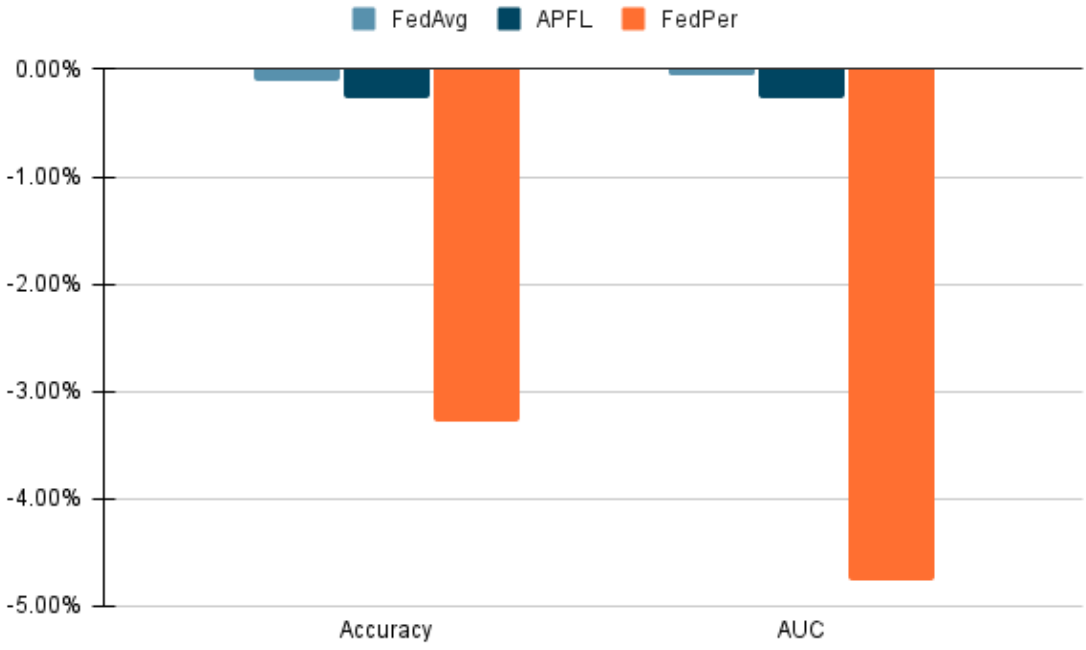


Figure 19: Median accuracy and AUC (vs 'central')

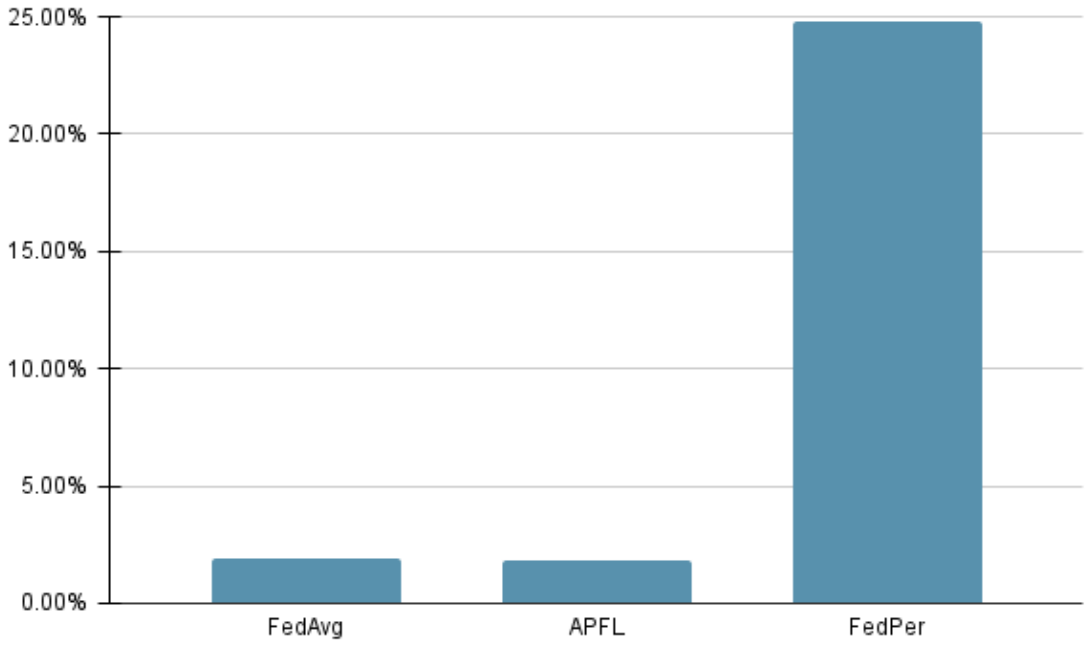


Figure 20: Median loss (vs 'central')

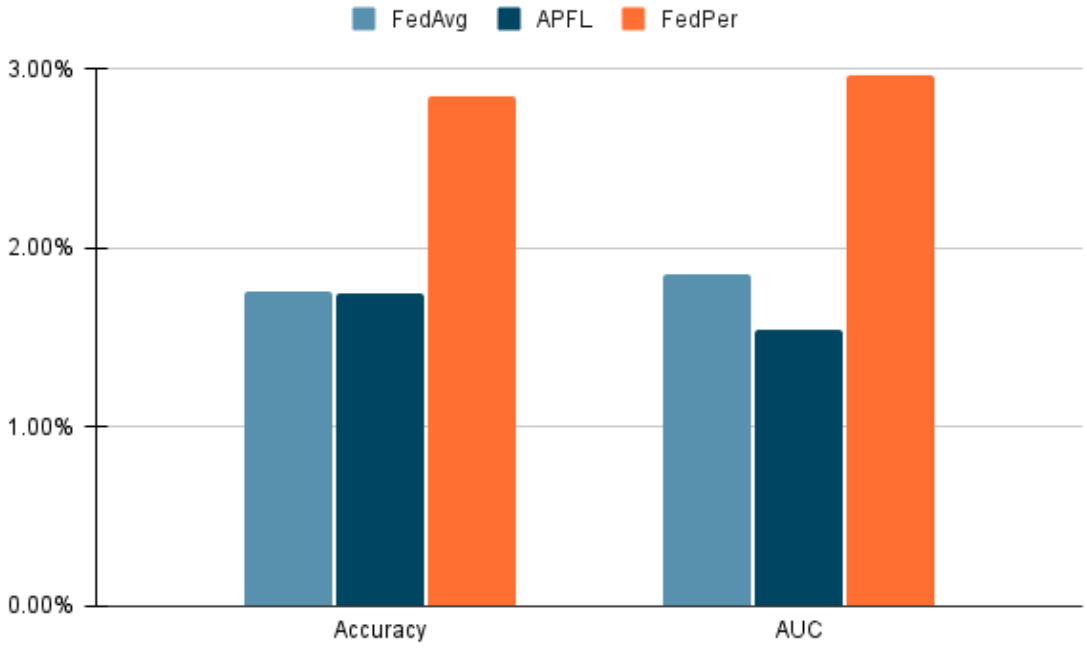


Figure 21: Max accuracy and AUC (vs 'central')

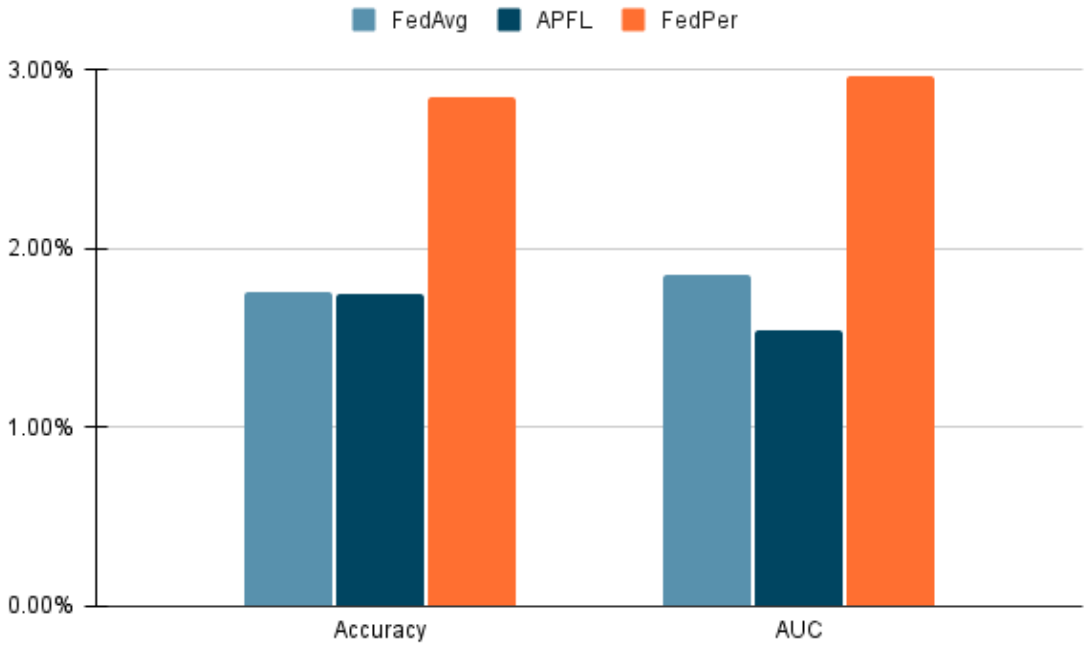


Figure 22: Max loss (vs 'central')

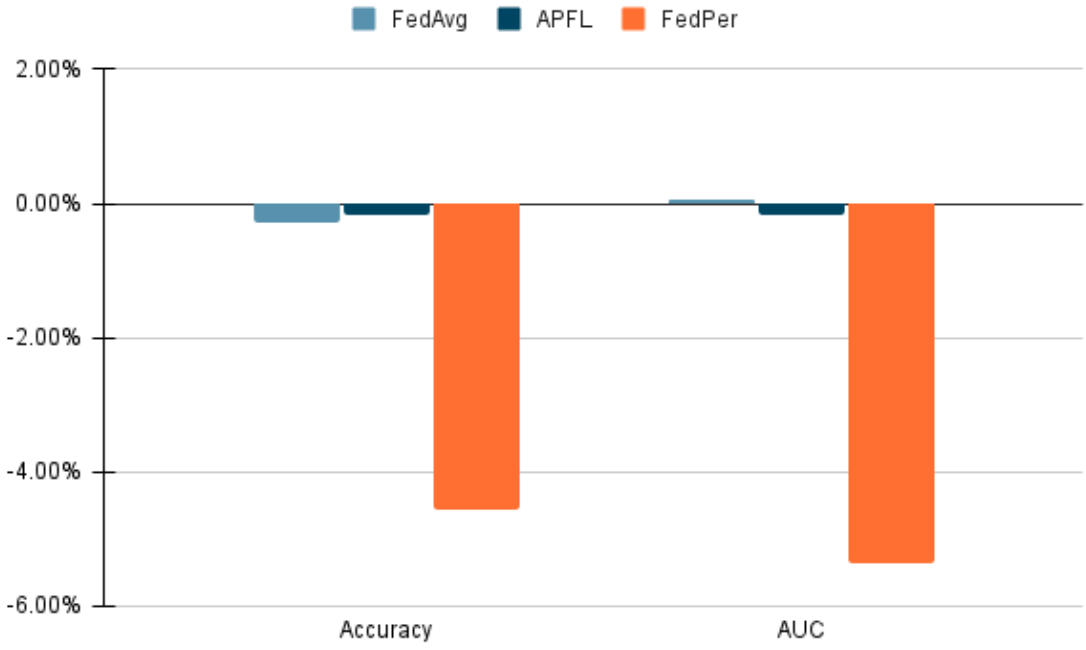


Figure 23: Min accuracy and AUC (vs 'central')

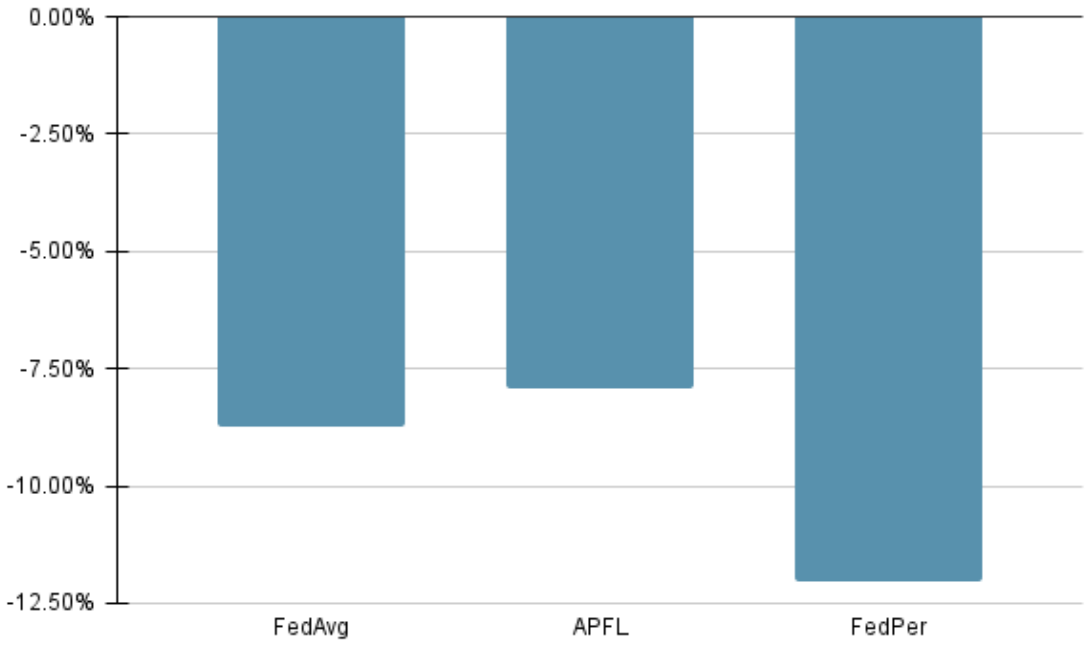


Figure 24: Min loss (vs 'central')

5.2.3 APFL

Let us now examine APFL, the first PFL architecture.

PQMs. Overall, APFL's PQMs are similar to their FedAvg counterparts. This is best shown in Figure 19 and Figure 20 (see Table 17). We can see that the median APFL accuracy and AUC only deviate from the FedAvg metrics by 0.2 Percentage Points (pp) or less. Loss is almost identical here with a difference of 0.0003. The same can be said for the Max and Min of these architecture's metrics, as in Figures 21, 22, 23, and 24 (see Table 19 and Table 20).

APFL displays the lowest variance of all federated architectures across all metrics; see Figures 25 and 26 (see Table 25). Only the central architecture's SD is lower (see Table 21). The increase of APFL's SD is 9.06%, 13.03%, and 19.03%, for accuracy, AUC, and Loss. The increase of the FedAvg architecture in that context is larger, with 20.95%, 19.15%, and 23.75%, respectively.

RelImp. In RelImp, APFL slightly outperforms FedAvg, which is not the case for the PQMs (see Table 26). The architecture's Median RelImp is 0.68% higher compared to FedAvg. Further, the Max is 1.12%, and the Min is 2.18% larger. RelImp SD here is the lowest of all federated architectures, illustrated in Figure 28, 47% smaller compared to FedAvg (see Table 26).

APFL is better than FedAvg when looking at RelImp in the case of certain datasets and DOs (see Table 8). Whereas FedAvg sees a drop in RelImp for both DO2s of the 'factor 2' versions, this is not the case for APFL. RelImp is better here, with an improvement of 3.83 pp and 1.8 pp, respectively (see Tables 12 and 15). APFL in the '10%-Regular' combination performs similarly, with an improvement of 5.54 and 1.4 (as in Table 14).

Finally, when looking at the Min RelImp, we can see that APFL performs best here, as in Figure 27. In the worst case, APFL only deteriorates model performance by 0.94 pp. FedAvg, the second-best performer here, is lower by an additional 2.16 pp (see Table 20).

5.2.4 FedPer

Lastly, we turn to FedPer, the second PFL to be analyzed.

PQMs. FedPer displays the largest differences in metrics compared to all other architectures; see Figures 19 until 24. Table 17 shows that FedPer has the lowest median accuracy and AUC, and the highest median loss. This becomes apparent when comparing FedPer's and APFL's median PQMs to the central architecture's. Here, FedPer's difference to APFL is -3.01 pp (accuracy), -4.48 pp (AUC), and +23 pp (loss) (see Table 22).

The gap is even larger when looking at the Min comparison vs. Central as in Table 24. Here, the difference to APFL is -4.37 pp (accuracy), -5.19 pp (AUC). Nevertheless, the Min loss is 4.14 pp lower. Also, FedPer outperforms APFL by 1.1 pp and 1.41 pp in accuracy and AUC when comparing their Max metrics to 'central'. The loss, however, is 26.2 pp higher here (see Table 23).

In all datasets besides one, FedPer is outperformed by APFL in PQMs (see Table 8). The exception is that FedPer performs better than all other architectures in the full—random noise combination, where it shows its best performance (see Table 13). Interestingly, that is not the case for the 10% equivalent, where it performs worst of all architectures (see Table 16).

The SD reflects these differences. It is the highest for FedPer across all metrics. Compared to the central architecture, its SD is 59.47%, 67.49%, and 82.33% larger for accuracy, AUC, and loss. By contrast, APFL's metrics here are only 9.06%, 13.03%, and 19.03% higher, respectively (see Table 21).

RelImp. The median RelImp for the FedPer architecture is 18.14 pp higher compared to APFL (see Table 17). The difference is even larger for Max RelImp (49.14 pp), as in Table 19. On the other hand, Min FedPer's RelImp is smaller (-7.17 pp) (see Table 20). The SD for the RelImp metric is 93% larger than FedAvg (see Table 26).

Often, FedPer displays the best RelImp, sometimes by a large margin. This is the case for all datasets besides 'full—regular' and 'full—factor 2', where it is worst (see Tables 11) and 12). The highest RelImp displayed by all architectures is FedPer1, present in '10%—factor 2' (see Table 15).

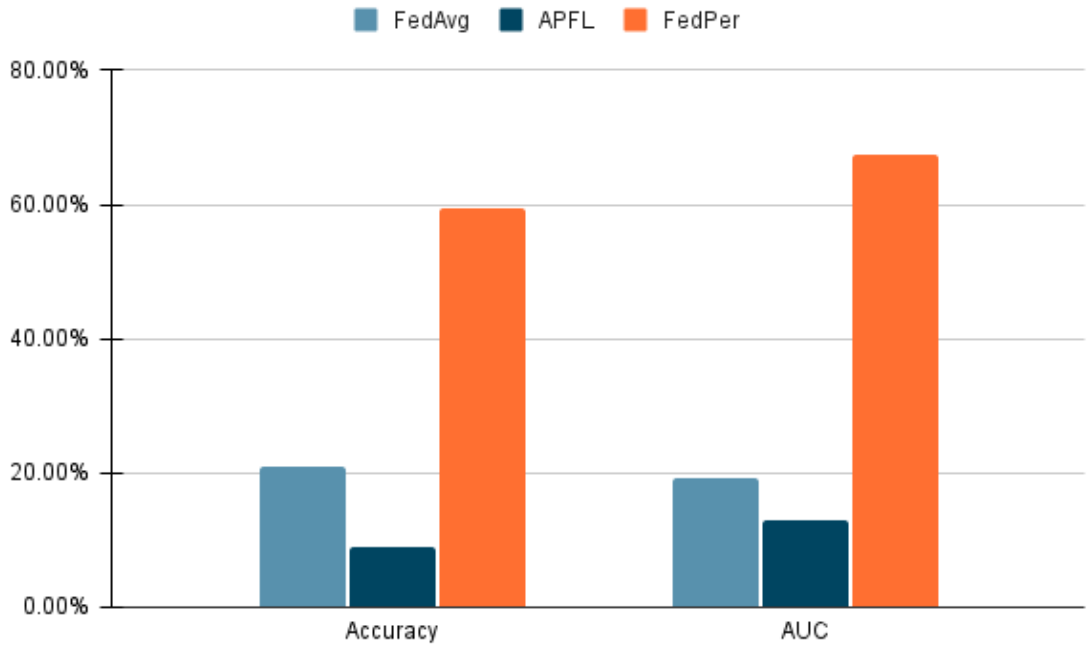


Figure 25: SD accuracy and AUC (vs 'central')

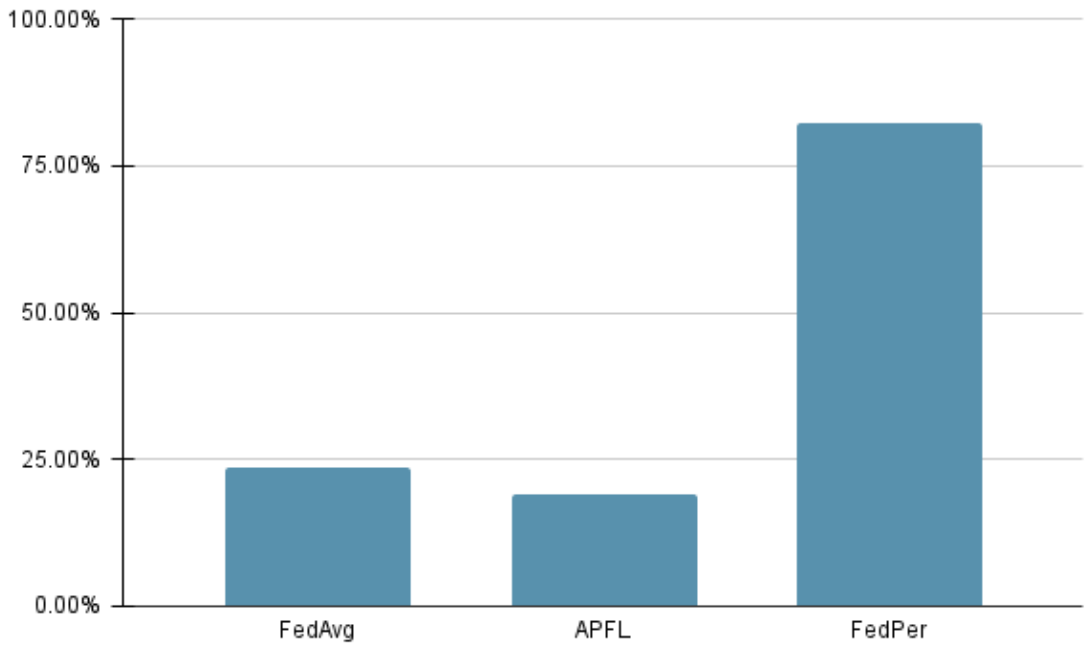


Figure 26: SD loss (vs 'central')

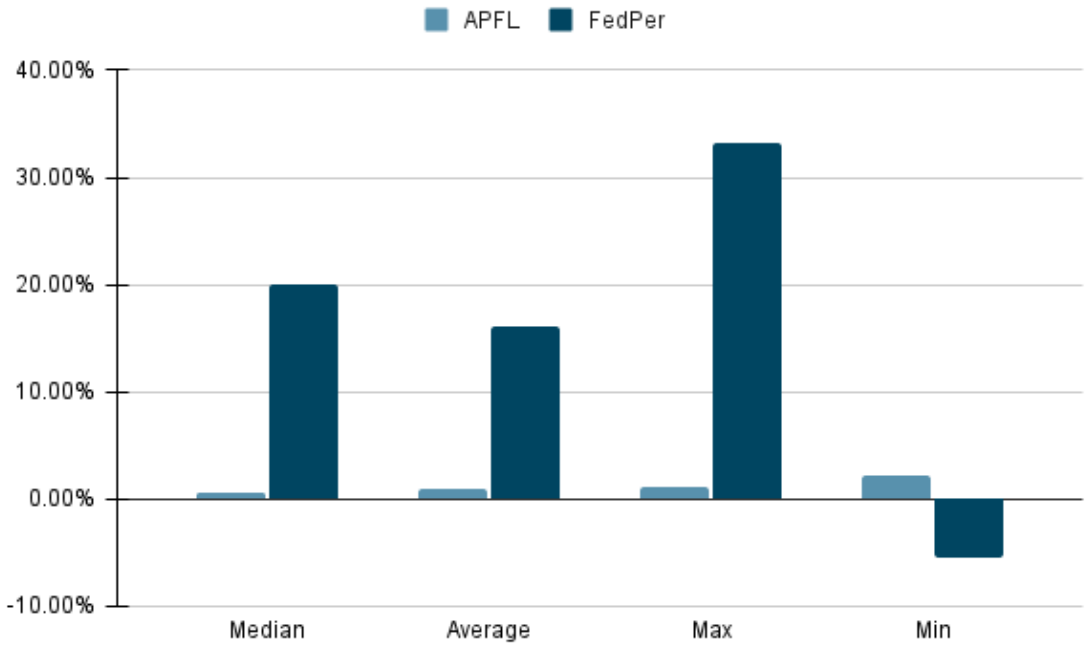


Figure 27: RelImp (vs FedAvg)

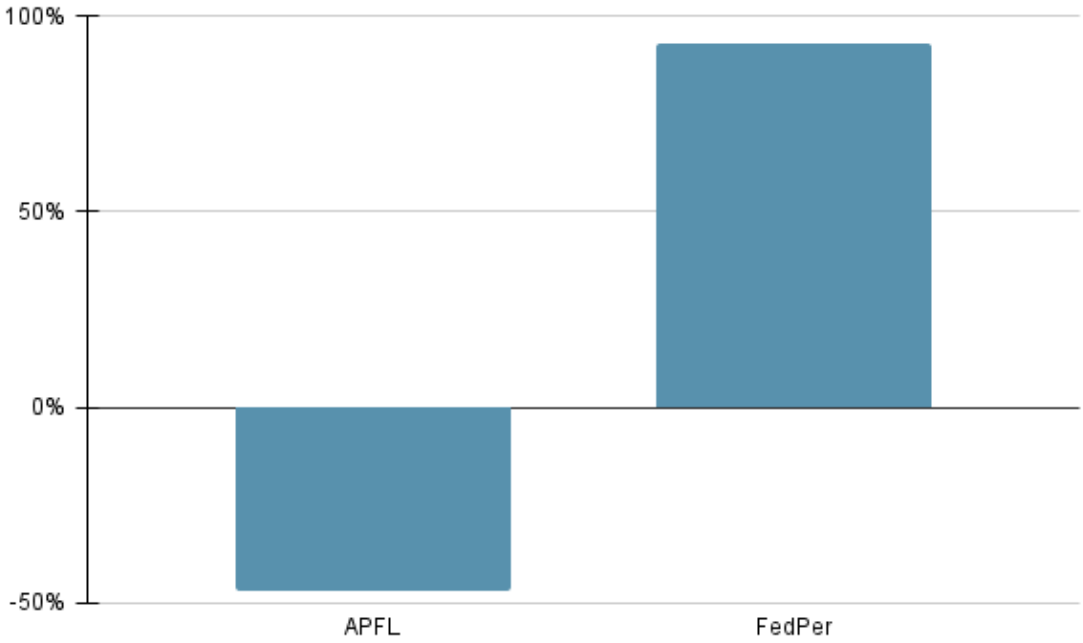


Figure 28: RelImp SD (vs FedAvg)

5.3 Discussion

It is important to observe not only where the differences in the results lie but also how large that difference is. A popular method for that is Cohen's d [82]. It can be defined as [83, p.3]:

$$d = \frac{\overline{X}_1 - \overline{X}_2}{s_{\text{pooled}}}$$

Cohen's d is a popular measure of the effect size between two means [83]. To deploy it, the results are first grouped by architecture. Then, we calculated the pooled standard deviation for each federated architecture s_{pooled}. As the control group, the central model performance was used. The results are displayed in Table 9. This allows us to compare effect sizes across all datasets, providing more substance to the discussion.

The larger Cohen's d, the greater the effect size. Important, its interpretation is not objective and should not be taken rigidly [83]. Nevertheless, for this thesis, the suggested interpretation of effects will be used [82]:

- small (d $\approx$ 0.2).
- medium (d $\approx$ 0.5).
- large (d $\approx$ 0.8).

5.3.1 Central Model

We will shortly address the performance of the central model. We first compare it to the random guessing baseline and then contrast the model results across the datasets.

Architecture	Accuracy	AUC	Loss	RelImp
FedAvg	0.1710	0.2243	-0.4112	-0.6015
APFL	0.2351	0.0904	-0.3668	0.3418
FedPer	-0.8825	-1.1436	1.6892	1.0738

Table 9: Cohen's d median improvement vs 'central'

Performance vs Baseline. The central architecture's median accuracy of 75.67% (see Table 22) offers a small improvement compared to the random guessing baseline of 74.63% shown in Section 4.2.1. Further, the median loss is somewhat high but acceptable, with 0.5255. Indeed, this is an existing critique of the implementation of the model[20]. However, its median AUC is well over 80% for all variants in this thesis, which is generally considered acceptable.

Performance Across Datasets. In general, the central architecture has the best median metrics, but only slightly (see Table 22). The architecture performs best with the unmodified dataset (see Table 11). With the altered datasets, the central model performance worsens. It could be due to the introduction of heterogeneity [59], [44].

The fact that the central performance on 'factor 2' datasets is better than on 'random noise' data (see Tables 12 and 15 vs 13 and 16) for this architecture could be caused by the presence of more irregularity in the second dataset variant. The further reduction of the metrics for the 10% dataset versions possibly showcases that the smaller dataset provides fewer learning opportunities for the model. Generally, the second slice of the full-random noise performs best for almost all models and architectures. This could be because it provides the whole sample, is chronically ordered, and is non-modified, helping the model to optimize its learning.

5.3.2 FedAvg

Here, we discuss the results of the FedAvg architecture. First, they are compared to the results of the central architecture. Patterns in its performance across different datasets are then discussed.

Performance vs Other Architectures. For the median, FedAvg generally closely follows the central model's performance but is slightly worse (see Table 22). Indeed, according to the literature, FedAvg is generally not aiming to improve model performance, all else equal. Instead, this architecture's primary goal is to preserve privacy through ModAg and, potentially, additional methods such as DP [39] [44]. It is to be noted that a small amount of noise is added to the model weights in all three federated architectures, slightly decreasing their model accuracy.

While the median is slightly worse, the average is slightly better (see Table 18). This is why, looking at Cohen's d in Table 9, we can find a small positive effect on Accuracy and AUC, and a small to medium negative effect on Loss. The effect on AUC here is the largest among all architectures. At the same time, the RelImp of FedAvg has a medium negative effect, the largest negative impact of all architectures for that metric.

[20]https://kaggle.com/code/egordm/deep-learning-recommendation-model-dlrm

Performance Across Datasets. FedAvg's PQMs vary between being slightly better and worse than 'central', depending on the dataset (see Table 8). A performance improvement here, however, was not to be expected [39] [44]. A possible explanation could be that, due to the low number of DOs in the implementation, a moderate amount of model personalization is happening in this implementation FedAvg. More details are provided in Section 6.2.

Also, differences between DO can be observed with FedAvg. We can highlight that both factor 2 modified datasets have decreased RelImp for DO2. At the same time, this is not the case for DO1 (see Tables 12 and 15). This could indicate that the FedAvg architecture had more performance issues for the second data slice, which was partly divided by 2, versus the first, which was partly multiplied by 2.

Both DO1 in 10%—regular and DO2 in 10%—factor 2 slightly outperform their FedAvg and central counterparts (see Table 8). That can not be related to the FedAvg technique itself, as both RelImp values are <100%, indicating that their first versions performed better. The same can be implied for both FedAvg versions of full—random noise, as their RelImp metrics are close to 100%. This further points to personalization due to the low amount of participants, as explained in Section 6.2.

5.3.3 APFL

This part discusses the APFL results. First, they are contrasted with the other architectures. Patterns in its performance across different datasets are then expanded upon.

Performance vs Other Architectures. The results of APFL and FedAvg display many similarities (see Tables 22, 23, and 24). A potential reason for this is that this thesis simplifies the architecture's implementation (for more details, see Section 6.2). As a result, APFL shows more parallels with FedAvg than originally designed in Deng et al. [63]. If implemented differently, the results would likely be more heterogeneous. The same limitation as in FedAvg (due to the low amount of participants) can be applied here; see Section 6.2.

However, differences between the results of the FedAvg and APFL architectures can be found. As mentioned in Section 5.2.3, all APFL metrics show the lowest SD of all three federated architectures (see Tables 25 and 26). This makes APFL the federated architecture with the slightest variance within the context of this thesis.

Looking at impact size, APFL shows the most significant positive impact on accuracy, a positive impact on AUC, and a negative impact on loss. Yet, all three effects are considered small (see Table 9). Still, APFL has a small positive impact on RelImp, while FedAvg has a medium negative impact on that metric.

Performance Across Datasets. The most notable performance metric for APFL is RelImp. The architecture considerably outperforms FedAvg's RelImp in certain datasets and data slices (see Table 8). Further, it displays the highest Min RelImp, never dropping below 99% (see Table 24).

Small to moderate improvements in PQMs can be seen with APFL, too (see Table 8). An outperformance of FedAvg by APFL is thus given in specific datasets and modifications. We can hypothesize that this is due to APFL's intended ability for personalization, which FedAvg by design has not. This would justify using APFL over FedAvg in specific applications.

5.3.4 FedPer

Lastly, let us now turn to the FedPer discussion. First, we discuss the results in comparison with the other architectures. Then, patterns in the architecture's performance across different datasets are highlighted.

Performance vs Other Architectures. FedPer's results stand in contrast to the other architectures. Compared to APFL, they do not follow FedAvg's performance as closely and display a much higher variance (see Table 21). A potential reason for this could be that this thesis' FedPer implementation is the most distinct, while APFL and FedAvg display more similarities, as further explained in Section 6.2.

Further, APFL's personalization approach (similarity-based model interpolation) can be hypothesized to be inherently closer to FedAvg. As used by FedPer, architecture-based parameter decoupling could be more systemically and structurally distinct [59]. It should also be less impacted by the low amount of DOs, as argued in Section 6.2.

FedPer is perhaps best characterized by its SD, which is the highest across all metrics (see Table 21). This variance is demonstrated in FedPer's results. Its Min metrics are the worst, while the Max results are best (see Tables 20 and 19). Interestingly, its median PQMs are the worst of all architectures. Its median RelImp, on the other hand, is best (see Table 17).

This is confirmed when examining Cohen's d, as in Table 9. All impact sizes here are large and by far the biggest. Further, the impact on accuracy and AUC is negative while being positive for loss. This signifies a negative impact on prediction quality. However, the impact on RelImp is largely positive, indicating that the relative improvement of the final, personalized model is the largest. We need to consider that RelImp here could be somewhat artificially higher than for the architectures, as further explained in Section 6.2. We also argue that this effect alone can not explain the improved RelImp.

Performance Across Datasets. The 'full—random noise' combination shows FedPer's best performance, beating all other architectures. FedPer's metrics here are outliers in two ways. Firstly, FedPer2 outperforms even FedPer1. This is reversed for all other architectures for that dataset, as DO1 is more heterogeneous. We can attribute this partially to the architecture, as the RelImp strongly increased for both DOs, and more so for DO1. Secondly, for FedPer, all 'full' variations, besides full—random noise, display a RelImp $< 100\%$.

FedPer's PQMs outperform APFL only for the 'full—random noise' pairing (see Table 13). In all other cases, FedPer performs worst when it comes to PQMs (see Table 8). Potentially, the FedPer architecture worsens the prediction quality. However, that answer would exclude the outlier of the 'full—random noise' results.

Another pattern is that, for FedPer, the RelImp is always above 100% for the reduced datasets (see Table 8). Further, the RelImp is always positive for FedPer if the data set has random noise modification. A hypothesis could be that FedPer tends to perform better for more heterogeneous datasets requiring more personalization. This could partially explain the better performance for the reduced, random noise datasets and DOs.

6 Conclusion

This chapter concludes the thesis by reflecting on its findings and implications. It starts by giving a summary of the previous chapters. Then, each RQ is answered. This thesis's theoretical and practical limitations are illustrated. In the next part, we discuss its implications for practitioners. Finally, additional areas for future research in this sphere are presented.

6.1 Summary and Findings

Privacy-preserving ML methods can be essential for DSs and DOs intending to navigate the current legislative, technological, and societal landscape. In this thesis, we leverage Design Science [16] to create a practical proof-of-concept application. In particular, we introduce FL and personalized PFL architectures for the advertising domain. A hypothetical research scenario, where previously centralized data-sharing is to be replaced with a federated architecture, is a reoccurring example in this thesis, highlighting the practical implication.

Firstly, the research problem is put into context in the background chapter. Here, we explain the basics of online advertising technology. Importantly, we argue that the ads industry has an 'addressability' problem. With the depletion of third-party cookies, among other factors, the industry may face difficulties (re)identifying users online. All stakeholders are thus incentivized to find alternative tracking solutions.

What if decentralized data ownership would enable DSs to train their models on data they can not directly access? This is the critical question in the concept of structured transparency, introduced in the second part of the background chapter. A selection of popular privacy-enhancing technologies in data science, including SMPC, HE, and DP, is presented there.

The fundamental concept of this thesis is FL. In its dedicated chapter, we define it as an ML learning setting where multiple clients collaborate to solve a problem. In contrast to distributed learning, however, the clients do not have direct access to the other parties' data. We further display a taxonomy of FLSs and explain their characteristics based on seminal literature [44].

Two drawbacks for FL in the context of heterogeneous, non-IID data are discussed: the slower convergence of FLSs (called client drift) and local data generalization. PFL is attempting to target these issues. A PFL taxonomy is provided and explained, grouping personalization approaches into categories [59]. For this thesis, the implemented PFL architectures can be classified into similarity-based model interpolation (APFL) and architecture-based parameter decoupling (FedPer); as in the taxonomy.

RQ1. The heterogeneity of user data is often overlooked in FL applications [12]. Thus, RQ1 is concerned with the following: 'What is the state of the art regarding research in the field of applied PFL for CTR prediction in advertising?'

To answer RQ1, ten distinct papers from the past two years are presented in state of the art table. The first part of the state-of-the-art table describes the setup and personalization used. We see that there may be a lack of standardization in the datasets and metrics. A potential solution is the 'FedAds' benchmark proposed by Wei et al. [74]. Almost all papers are dedicated to CTR prediction and use NNs, a popular use-case in

the domain. Interestingly, six papers rely solely on ModAg, which is inherent to FedPer. Three other studies use DP, while one uses HE. Comparing the state of the art to the taxonomy, we find that both personalization strategies and all four personalization categories are represented at least once in the table. However, KD appears five times, and meta-Learning three times. In contrast, TL, data augmentation, and clustering each have one representative. Accuracy, AUC, and loss are the most popular performance metrics, while privacy metrics are rare. The metrics are leveraged in the literature to contrast the model performance with local, global, or other (P)FL models. All papers note an outperformance of their model to the comparison models.

RQ2 and RQ3. The second RQ deals with implementing FL in the context of ads CTR prediction models via PySyft: 'How could two data owners exchange advertising CTR prediction model dictionaries federatively through PySyft?'. The third RQ further explores PFL implementations: 'How can architectures for PFL look like in that context?'

PySyft's architecture and its deployment are explained in the PySyft section. This thesis uses a public ad dataset[21] and partly modifies the data for additional heterogeneity. A public ML model is leveraged[22], while accuracy, AUC, loss, and RelImp are used to assess model quality. Four architectures are implemented for each of the six different dataset variations. One is a central version used as a baseline. The three federated architectures used are FedAvg, APFL, and FedPer. The federated implementations use PySyft to store and pass the model dictionary. All implementations are done in Jupyter Notebooks and can be found here[23].

RQ4. Lastly, the results are thoroughly analyzed, discussed, and interpreted in the results, analysis, and discussion chapter. This is done for the fourth RQ: 'How do the results of these FL and PFL implementations compare with each other regarding a set of metrics?'

We answer this RQ by using prediction quality metrics and the relative improvement of the respective models. The comparison results are shortly summarized below. It is essential to note the tentativeness of the interpretations due to the thesis' limitations; see the limitations section. The central architecture performs best with the unmodified dataset. On the other hand, the federated architectures often outperform their central counterparts when training on more heterogeneous and limited datasets. This is the case, for example, for the full—random noise variation, where all three architectures beat 'central' in terms of PQMs. In general, both FedAvg and APFL have positive impact sizes on the PQMs. A potential interpretation of this finding is that the personalization effect of PFL positively impacts prediction quality when dealing with data heterogeneity. This is the intended purpose of PFL [59]. Yet, FedPer's median impact on the PQMs is large and negative. We note that FedAvg sometimes also improves performance, which we assume to be due to a limitation of this thesis; see the limitations section. A prediction quality improvement is, all else equal, not expected according to the theory [39]. FedAvg has a medium negative impact on the relative improvement of the model. Both APFL and FedPer outperform it in that regard, with a small and medium improvement,

[21] kaggle.com/competitions/criteo-display-ad-challenge
[22] kaggle.com/code/egordm/deep-learning-recommendation-model-dlrm
[23] github.com/PalexGoneGit/PFLAdsPySyft

respectively. This, again, is intended by the PFL architecture, as it looks to improve local model performance, especially on heterogeneous data [59]. While RelImp improvements are essential, relying solely on them is insufficient. Absolute improvements in prediction quality are vital in justifying the choice of architecture. While we sometimes see minor improvements in prediction quality when choosing APFL over FedAvg, this changes depending on the dataset. Contrasting APFL with FedPer yields mixed results. APFL shows more consistent, but only slightly better PQMs, of which the largest impact is on accuracy. It is, additionally, characterized by a relatively low variance. Further improvement of performance metrics might be needed for most cases to justify its use over FedPer properly. We could label it to be a 'small-risk, small-reward' personalization approach for the context of this thesis. FedPer is outperformed by both APFL and FedPer in all but one dataset. However, that dataset shows the best performance of all models for all PQMs and RelImp. This inconsistency and variance characterize FedPer. We could call it a 'high-risk, high-reward' personalization approach for our context. We can summarize that a general recommendation of one architecture over the other is impossible. If privacy is necessary, the literature agrees that FL has many advantages over central learning. Yet, it is associated with higher communication and computation costs [44]. When choosing FL over PFL or vice versa, the literature suggests that the latter is better for heterogeneous, personalized data [59]. The price for that is a particular increase in algorithmic complexity. We can tentatively confirm that with the findings of this thesis. With this thesis ' results, always choosing one PFL architecture over the other is impossible. Depending on the situation, APFL might outperform FedPer or vice versa. Always selecting APFL over FedPer might cost potentially substantial improvements in performance. Yet, FedPer seems to be only better in a few cases. A tentative hypothesre could be that one of these cases is the presence of highly heterogeneous data with a strong need for local personalization. These hypotheses need to be, of course, further explored. The feature comparison matrix in Table 10 illustrates the potential advantages and disadvantages of the different architectures for the context of this thesis.

Architecture	**Comm. & Comp. Costs**	**Privacy**	**Algorithmic Simplicity**	**Local Data Personalization**	**Variance Reduction**
Central	+	-	+	-	+
FedAvg	-	+	0	-	0
APFL	-	+	-	0	+
FedPer	-	+	-	+	-

Table 10: Feature comparison matrix

6.2 Limitations

In this section, the theoretical and practical limitations of this thesis are presented.

Lack of Privacy Guarantees. While FL may seem private initially, some issues have been identified. Researchers argue that a malicious party could exploit shared information through the intermediate layer [84]. Also, a paper has shown that private images can be reconstructed from FLSs [85]. Furthermore, Zhu et al. [86] demonstrate that private training data can be obtained through leakage by the central server.

FL is not privacy-preserving by itself. It needs to be combined with other PETs to offer more significant privacy guarantees [14]. Combinations of other PETs with FL are thus promising and possible, for example, in PySyft [14].

SGD Simplification. Due to practical constraints, the SGD protocol, part of all federated architectures presented (see the algorithm in Appendix A.4 for the notation), was simplified for all implementations. The simplifications include [39] the following:

- Lack of initialization of the global model and its distribution to the clients (this is only implied in the implementation).
- Lack of sending of the global model dictionary to the participants (shown in Figures 16,17, and 18 but only implied in the code).
- Only two DOs and no algorithm for their selection.
- Lack of iterative loops of training and testing between the global and local model. This would include model evaluation, weighing of local dictionaries, and convergence criteria.

APFL Simplification. For the APFL implementation, the parameter for the mixture weights α determines the extent to which the local model influences the client's personalized model, as in Deng et al. [63, p.8]. It has been set to 0.5 in this thesis, while in a full implementation, as described in the algorithm in Appendix A.5, the weights would be dynamically adapted. Further, the synchronization gap τ is also missing in the implementation present in this thesis. It specifies the number of local iterations before synchronizing with the global model. This makes this Thesis' APFL implementation more similar to FedAvg than in Deng et al. [63].

FedPer Simplification. In our FedPer implementation, the client does not send the number of local data points n_j to the server. In turn, the server does not calculate nor use weights y_j for the aggregation of the global base model (Deng et al. [63, p.8]). Further, due to the architecture of this implementation, the RelImp needs to be interpreted with more care. For FedPer, the DOs only train the bottom layer of the model first. The PQMs for that model are naturally worse than the PQMs of a model with an included top layer. The RelImp for FedPer is thus somewhat artificially higher than for FedAvg and APFL. An alternative way would have been to calculate the RelImp considering a different first model with both layers present. However, that would have been methodically inconsistent with the other RelImp calculations, which have been using the same model by which the model dictionary was directly trained. That effect alone, however, can not fully explain FedPer's performance. Firstly, sometimes PQMs are also improved in the personalized model of FedPer. This means that some 'true' RelImp must be present. Secondly, FedPer sometimes displays a negative RelImp, like in full—regular and full—factor 2. This disproves that the second model of FedPer is strictly better than the first.

Central Model Implementation. The central model was tested and validated on one shared dataset. Theoretically, one could split the datasets in half and test the model twice, simulating a DO1 and DO2. We decided against that approach to be methodically consistent with the other implementations, where each model is strictly

tested on the dataset on which it was trained. Further, it is unclear if that split would have been more or less representative of the real world, as the idea for the 'before scenario' is that all user data is combined for model training and evaluation purposes.

Few DOs. We find unintentional and unexpected personalization effects in the FedAvg implementation. The reason could be that only two DOs are simulated in this thesis. Hence, a single data owner's contribution to the final model weights is relatively large. That makes the global model dictionary relatively similar to the local model dictionary of both DOs. This, combined with the ability of NNs to adapt to the local data in the final training stage, could increase the ability for personalization for FedAvg. It further could impact APFL and, to a smaller degree, FedPer (as its algorithm is more distinctly different, see the APFL simplification). This effect should be smaller as the number of DOs grows because the global model dictionary becomes more and more diluted.

Dataset Limitations. The dataset was not originally federated and has been manually altered for that purpose. It is, further, artificially heterogeneous, employing data manipulation. Also, while heterogeneity was added, the data still has the IID property. That somewhat reduces the practical implications of the results [74]. Finally, due to the simplifications of the architectures, only local data generalization is addressed, while client drift is not [58].

6.3 Practical Implications and Further Research

In this final section, we derive implications for practitioners from this thesis. Lastly, areas for further research are highlighted.

6.3.1 Practical Implications

This thesis provides several potential practical implications.

Use in Advertising. In light of the addressability problem, organizations in the advertising sphere may find FL and PFL to be viable solutions. This is because these technologies can provide additional privacy and potential performance improvements. Such tangible benefits could be deemed satisfying for the motivation of federation [2].

Comparing (P)FL to currently developed privacy solutions, such as Google's 'Federated Learning of Cohorts', is important. Appendix A.1 and A.2 illustrate such solutions. (P)FL could be preferable over these solutions, as it is arguably more decentralized and less invasive [44], [2], [15]. Yet, more theoretical and practical developments in the realm of (P)FL for advertising are required.

However, FL does not guarantee increased privacy. Instead, it needs to be combined with other PETs, such as DP or HM [14]. PySyft, as well as other frameworks, are potential enablers here. It is, therefore, promising to explore further practical applications of such frameworks.

Architecture Selection. This thesis suggests that FedPer, APFL, and FedPer could be preferable over central model training. Yet, these findings are tentative and require further exploration. In practice, a hypothetical model performance improvement could be achieved indirectly through privacy preservation. This could happen due to increased

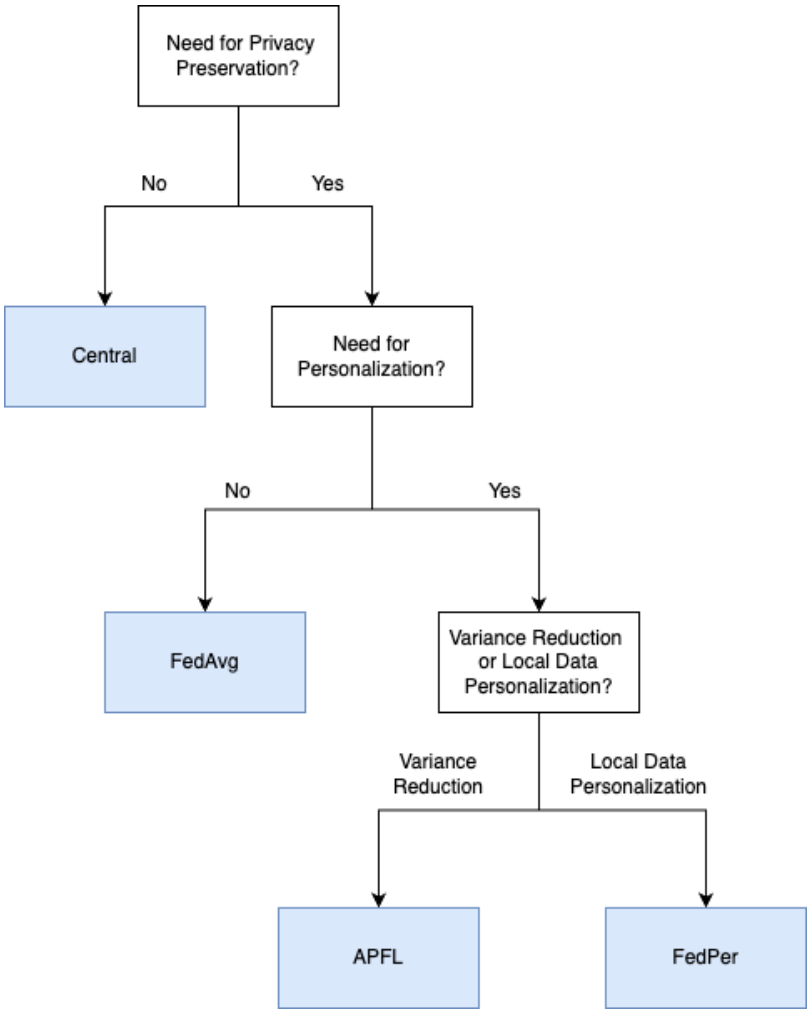


Figure 29: Decision tree for architectures

privacy incentivizing more parties to participate in the federation. The increased diversity and size of the data could, in turn, improve model performance. Yet, this is not reflected in this thesis.

When comparing FL to PFL, a larger RelImp alone can not be the deciding factor for practitioners. Instead, an increase in performance metrics is necessary. A better model performance could, however, offset the additional efforts associated with PFL. While this has been tentatively demonstrated in this thesis' implementations, it requires further exploration.

The results of this thesis let us conclude that APFL is not strictly better than FedPer, or vice versa. Instead, practitioners need to select a suitable model according to several factors. One such factor could be variance reduction (where APFL seems better), the other local data personalization (where FedPer is deemed superior), see Table 10. Figure 29 illustrates a hypothetical, simplified decision tree. Yet, these findings are tentative and need confirmation.

6.3.2 Further Research

The FL and PFL domains are relatively new and increasingly popular (see the Federated Learning chapter). They provide ample opportunities for further research [44]. The limitations of this thesis could be exploited for that purpose. For example, less simplified algorithms of the provided architectures could be tested. Other datasets could be used, which could inherently exhibit more heterogeneous, non-IID data. Personalization could be tested and confirmed, particularly for client drift. Also, using the advertising dataset and benchmarking toolkit 'FedBench' [74] could prove fruitful for standardization. Additionally, other advertising domains (non-CTR-related), as well as other industries altogether (e.g. other recommendation models) could be explored.

The research presented in this thesis could be expanded. For example, other parameter decoupling or model interpolation approaches could be tested. Further, new research could include other architecture-based and similarity-based personalization approaches. Also, methods from the global model personalization strategy, left unexplored in this thesis, could be examined.

Finally, this research can be augmented by conducting experiments on a larger scale. New findings could be investigated by adding more DOs and larger datasets. Especially not simulating the federated setting but directly implementing the models on client devices, like for the Google next-word-prediction model [43], would provide valuable insights for academics and practitioners alike.

A Appendix

A.1 Online Tracking Technologies

Currently, several promising technologies are being used to tackle the issue of the discontinuation of third-party cookies. They include:

- Fingerprinting [15],
- Third-party identity providers [87], [88],
- Data clean rooms [31], [89], and
- ID stitching [24], [15], [90].

Fingerprinting. Fingerprinting describes analyzing the user's browser characteristics to identify their device without using cookies. These features can include browser customization (e.g. plugins enabled), browser-level user configurations (e.g. timezone), browser family, operating system as well as hardware and network settings (e.g. screen resolution) [15]. Combining these characteristics can thus be used to identify and re-identify a user [24] uniquely. Its applications vary from fraud prevention to cross-website tracking for advertising purposes. Therefore, it can be considered invasive and can violate user privacy. Paradoxically, ad-blockers can be used for fingerprinting [15]. Yadav et al. [91] provides an overview of the key terminology of fingerprinting.

Identity Providers. An identity provider is a trusted party authenticating a user while logging in through single sign-on. When successful, the user can access the service provider's online cloud services without requiring a 'traditional' log-in through a username and password [87]. The identity provider is responsible for user security and privacy and, in turn, can track the user through websites and monetize user data [88].

Data Clean Rooms. A data clean room is a collaboration environment allowing several parties to share data while following data access and privacy limitations [31]. It can support several use cases for advertising, like joining multiple databases with matching keys [31]. As a result, insights for customer targeting, audience segmentation, and the measuring of ad impact can be gained [89]. Traditionally, advertisers and publishers share their first-party data with a third party to match and activate their audience. Data clean rooms function as a private environment and provide services that enable audience matching for advertising without exposing first-party data to other parties. For example, a customer overlap between different parties can be identified to estimate an ad campaign's effectiveness. Also, first-party data can be enriched using external data points while data control is maintained [31].

ID Stitching. ID stitching, sometimes called user stitching, refers to identifying and matching online data to the same user. This happens by identifying and grouping sessions corresponding to the same user on different channels, browsers, devices, and platforms. The process solely relies on user interactions with the online content and metadata. In contrast to fingerprinting (which uses device configurations), this method uses behavioral patterns that are device-independent [92].

A.2 Technology Companies' Privacy Solutions

Apple. Apple, for example, has declared itself to be 'building in privacy from the ground up'[24]. Its popular browser 'Safari', finds and eliminates third-party cookies in Safari by default using Intelligent Tracking Prevention[25]. The company has also introduced new privacy-preserving technologies for mobile, such as App Tracking Transparency[26] [93]. The App Tracking Transparency includes a compulsory opt-in system on the latest iOS devices called Identifier for Advertisers. It requires in-app prompts, asking for explicit permission for publishers to track user data. Such a change from the previous data tracking has lowered opt-in rates and severely impacted the advertising industry [93], [94]. In addition, the company also introduced the Private Click Measurement. It mostly prevents sites from tracking user interactions while allowing advertisers to register ad clicks and conversions. Privacy is more protected than third-party cookies, as user data reported to the advertiser is limited, anonymized, and submitted with a time delay [95]. To facilitate advertising under these new circumstances, the organization offers its SKAdNetwork. This solution provides ad insights without revealing personal user data (when the user has not opted-in to Identifier for Advertisers). However, the amount of analysis and attribution possible here is restricted due to the limited data[27]. Apple is adopting and actively researching the privacy-preserving technique of DP section [38].

Alphabet. Google has also committed to phasing out third-party cookies in its popular browser, Google Chrome. The company has also said it is working on new privacy-preserving technologies allowing advertisers to target users without using third-party cookies. Those initiatives are collected under the umbrella term Privacy Sandbox[28]. According to Google, the Privacy Sandbox is a set of industry-wide technologies currently under development aiming to preserve privacy on the web and Android. It provides safer alternative solutions to third-party cookies and covert tracking like Fingerprinting. This, in turn, enables organizations to continue their online business while respecting user privacy[29] [22].

The technology used here includes, among others, DP and on-device processing. These will be further explained later on. The goals include fraud and spam prevention, showing relevant content and ads, measuring online ads, and strengthening cross-site privacy boundaries. The company highlights three key technological initiatives that are particularly relevant in the context of the Privacy Sandbox[30]:

- Topics API.
- Protected Audience API.
- Reporting API.

[24] apple.com/newsroom/2023/01/apple-builds-on-privacy-commitment-by-unveiling-new-efforts-on-data-privacy-day/

[25] apple.com/safari/docs/SafariWhite$_{P}aper$Nov2019.pdf

[26] apple.com/newsroom/2021/06/apple-advances-its-privacy-leadership-with-ios-15-ipados-15-macos-monterey-and-watchos-8/

[27] developer.apple.com/documentation/storekit/skadnetwork/

[28] blog.google/products/chrome/privacy-sandbox-tracking-protection/

[29] privacysandbox.com/learning-hub/

[30] See Footnote 27

A key aspect here is Topics API, previously known as 'federated learning of cohorts'. It is not to be confused with FL, as it functions differently and is not utilizing that technology (hence the name change)[31]. This technology is designed to preserve privacy while displaying relevant content and ads to users. It groups each website the user visits into high-level topics and sub-topics. For example, the current list includes 'games', which can be broken down into 'games/boardgames', 'games/roleplaying games', and so on. The browser collects the most-visited topics of a user. Then, it groups the user in a clustered group and shares that information with the sites. A profile is thus built per group and not per individual. This enables advertisers to display more relevant ads without revealing the specific websites one has visited[32], [22]. Another key privacy technology under the umbrella of the Privacy Sandbox is called Protected Audience API. While today, third-party cookies expose collected information to others, this technology protects private data while ads can still be displayed. Here, as formerly, a cookie stores information regarding user interest. However, in the case of Protected Audience API, that cookie is stored on the user's device only, and the decision to display appropriate ads when visiting a site is also made on that particular device. In this method, personal data never leaves the user's device and protects their privacy[33]. The third topic is Attribution Reporting API, which allows advertisers to measure their campaigns' impact without tracking an individual's online activity. When an interaction with an ad, such as a click or a purchase, occurs, that information is stored on the user's device. The reports sent to the advertiser are time-delayed, contain only aggregated data, and include noise to protect user privacy further [22].

Others. Similarly, Mozilla Firefox, with its 'enhanced tracking protection' feature and Edge's 'Microsoft tracking prevention', also provide and continuously develop additional privacy protection options [26]. Ad blockers are a popular way to remove online ads on web pages. While in the past, they were mostly used in the form of browser extensions, ad-blocking features have been increasingly introduced into browsers themselves. Nowadays, most ad blockers block ad delivery and prevent tracking and profiling [26].

[31] github.com/patcg-individual-drafts/topics/blob/main/taxonomyv2.md
[32] privacysandbox.com/learning-hub/
[33] blog.google/products/chrome/privacy-sandbox-tracking-protection

A.3 FL Data Partitioning Approaches

Horizontal FL. Yang et al. [2] provide an extensively comprehensive overview of all three techniques, their algorithms, and their architectures. In the case of HFL, two datasets with many overlapping features exist (or even all features overlap). However, there is little commonality between the data entries [3]. To visualize this, one can imagine a simple data set in a table containing user data. This could be, for instance, the user database of two different banks. This would be an example of cross-silo FL [44]. The table is horizontally partitioned, meaning it splits the user entries into two parts belonging to each company. However, both parts share the vertical dimension, the features of the table [2]. Interestingly, a cross-device FL application is possible here, too. For example, the next-word prediction of a mobile keyboard falls into this category [43], [44]. Therefore, HFL contributes to increasing the sample size [3]. More formally (as in Yang et al. [2, p.49]:

$$\chi_i = \chi_j, \gamma_i = \gamma_j, I_i \neq I_j, \forall D_i, D_j, i \neq j$$

Here, the data feature space pair between two participants is deemed to be the same, i.e. $\chi_i = \chi_j$. The same applies to the label space pair, i.e. $\gamma_i = \gamma_j$. However, the user identifiers I_i and I_j of the different parties are different. D_i, D_j represent the respective data sets [2]. The FedAvg algorithm is popular in HFL systems [39].

Vertical FL. Contrasted to HFL, VFL is characterized by varying data features and shared data samples. This is the case when user features have little in common while the users strongly overlap. We can, again, imagine a simple dataset visualized in a table. This time, however, the table is split vertically: the user entries are the same, but the vertical features are split along two tables [3]. This could, for example, happen when companies from two industries share their customer base. One, a shoe company, would have data about the shoe size, gender, and favorite shoe color of the customers. The other, a company producing wearable smartwatches, could have the same user registered. However, the data the second organization has collected is the user's activity level, daily steps, and average heart rate. This basic example illustrates that these companies do not only have a similar user base and different feature sets in their databases but also that a potential exchange of that information could be mutually beneficial due to synergy effects. Indeed, VFL is generally only realistic in a cross-silo, B2B context [44], [2]. We can formally denote that (Yang et al. [2, p.70]):

$$\chi_i \neq \chi_j, \gamma_i \neq \gamma_j, I_i = I_j, \forall D_i, D_j, i \neq j$$

Here, the data feature space pair between two participants is deemed to be different, i.e. $\chi_i \neq \chi_j$. The same applies to the label space pair, i.e. $\gamma_i \neq \gamma_j$. Nevertheless, the user identifiers I_i and I_j of the different parties are the same. D_i, D_j represent the respective data sets [2].

Federated Transfer Learning. With HFL and VFL, the datasets either have common data features or common data samples, respectively. Realistically, however, many practical examples display neither commonality. Often, not many data features or samples are shared between the parties, while some little overlap can exist in both dimensions. Their heterogeneity may manifest in different aspects [2]:

- Samples and features.
- Data distribution.
- Data size.
- Amount and quality of labeling.

These issues exceed the problems considered in the HFL and VFL use cases. Li et al. [10] provides an example of such a case: Take a group of hospitals collaborating to improve their cancer diagnosis. Though overlaps can happen, they likely have mostly distinct patients and different medical examination results. Zhang et al. [3] provides a different use case. It is concerned with two companies: a Chinese e-commerce company and a social application in the US. Nguyen et al. [51] deploy a practical experiment on IoT devices using FL. The devices range from coffee machines to cameras, so their data samples and features can be described as rather heterogeneous. One can see the lack of overlap here. Can FL still be applied in these circumstances?

Pan and Yang [96] survey a transfer learning technique in their influential paper. It aims to address the data problems described in the general context of ML. The solution enables the creation of performant and reliable ML models in such circumstances. This is done by inferring and transferring knowledge between two domains. This happens by finding and exploiting an invariant between a resource-rich and a resource-poor domain. That invariant is then used to transfer information from the first to the latter [96].

For this reason, Liu et al. [97] first proposed FTL as a solution. It is, therefore, the application of transfer learning to the context of FL. It differs from the original by introducing three additional constraints [97], [2]:

- The system is built on data owned by different participants
- Data belonging to each party cannot be exposed to other parties
- User privacy, as well as data and model security, needs to be protected

The situation when FTL is applied can be formally defined as follows (Yang et al. [2, p.85]):

$$\chi_i \neq \chi_j, \gamma_i \neq \gamma_j, I_i \neq I_j, \forall D_i, D_j, i \neq j$$

Here, the data feature space pair between two participants is deemed to be different, i.e. $\chi_i \neq \chi_j$. The same applies to the label space pair, i.e. $\gamma_i \neq \gamma_j$. Also, the user identifiers I_i and I_j of the different parties are different. D_i, D_j represent the respective data sets [2].

A.4 FedAvg Algorithm

Algorithm 1 Server execution of FedAvg as in MacMahan et al. [39, p.5]. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

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

Algorithm 2 Client execution of FedAvg as in MacMahan et al. [39, p.5]. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

function ClientUpdate(k, w)
    $B \leftarrow$ (split P_k into batches of size B)
    for each local epoch i from 1 to E **do**
        for batch $b \in B$ **do**
            $w \leftarrow w - \eta \nabla L(w; b)$
        end for
    end for
    return w to server
end function

A.5 APFL Algorithm

Algorithm 3 Local descent APFL (Deng et al. [63, p.8])

Input: Mixture weights $\alpha_1, \ldots, \alpha_n$, Synchronization gap τ, A set of randomly selected K clients U_0, Local models $v_i^{(0)}$ for $i \in [n]$ and initial local version of global model $w_i^{(0)}$ for $i \in [n]$.
for $t = 0, \ldots, T$ **do**
  parallel for $i \in U_t$ **do**
  if $t \mod \tau \neq 0$ **then**
    $w_i^{(t)} = w_i^{(t-1)} - \eta_t \nabla f_i \left(w_i^{(t-1)}; \xi_i^t \right)$
    $v_i^{(t)} = v_i^{(t-1)} - \eta_t \nabla v f_i \left(\bar{v}_i^{(t-1)}; \xi_i^t \right)$
    $\bar{v}_i^{(t)} = \alpha_i v_i^{(t)} + (1 - \alpha_i) w_i^{(t)}$
    $U_t \leftarrow U_{t-1}$
  else
    each selected client sends $w_i^{(t)}$ to the server
    $w^{(t)} = \frac{1}{|U_t|} \sum_{j \in U_t} w_j^{(t)}$
    server uniformly samples a subset U_t of K clients.
    server broadcasts $w^{(t)}$ to all chosen clients
  end if
  for $i \notin U_t$ **do**
    $v_i^{(t)} = v_i^{(t-1)}$
  end for
end for
for $i = 1, \ldots, n$ **do**
  Output: Personalized model: $\hat{v}_i = \frac{1}{ST} \sum_{t=1}^{T} p_t \left(\alpha_i v_i^{(t)} + (1 - \alpha_i) \frac{1}{K} \sum_{j \in U_t} w_j^{(t)} \right)$;
  Global model: $\hat{w} = \frac{1}{KST} \sum_{t=1}^{T} p_t \sum_{j \in U_t} w_j^{(t)}$.
end for

A.6 FedPer Algorithm

Algorithm 4 FedPer-Client(j) (Arvazhagan et al. [67, p.4])

Require: $f(\cdot;\cdot,\cdot)$, e, b, $\{(x_{j,i}, y_{j,i})|i \in \{1,\ldots,n_j\}\}$
Require: $\eta(k)_j$ for $k \in Z^+$
Initialize $W(0)_{Pj}$ at random
Send n_j to server
for $k = 1, 2, \ldots$ **do**
    Receive $W(k-1)_B$ from server
    $(W(k)_B, j, W(k)_{Pj}) \leftarrow \text{SGD}_j(W(k-1)_B, W(k-1)_{Pj}, \eta(k)_j)$
    Send $W(k)_B, j$ to server
end for

Algorithm 5 FedPer-Server (Arvazhagan et al. [67, p.4])

Initialize $W(0)_B$ at random
Receive n_j from each client $j \in \{1,\ldots,N\}$ and compute $\gamma_j = \frac{n_j}{\sum_{j=1}^{N} n_j}$
Send $W(0)_B$ to each client
for $k = 1, 2, \ldots$ **do**
    for Each client $j \in \{1,\ldots,N\}$ **do**
        Receive $W(k)_B, j$ from client j
    end for
    Aggregate $W(k)_B \leftarrow \sum_{j=1}^{N} \gamma_j W(k)_B, j$
    Send $W(k)_B$ to each client
end for

A.7 Results, Additional Tables, and Figures

Metric	Central	FedAvg1	FedAvg2	APFL1	APFL2	FedPer1	FedPer2
Accuracy	0.7600	0.7508	0.7574	0.7544	0.7511	0.7284	0.7384
AUC	0.8347	0.8246	0.8307	0.8274	0.8241	0.7819	0.7915
Loss	0.5235	0.5409	0.5374	0.5385	0.5422	0.7475	0.7323
RelImp	-	99.27%	99.57%	101.25%	99.63%	91.89%	92.80%

Table 11: Full—regular performance metrics

Metric	Central	FedAvg1	FedAvg2	APFL1	APFL2	FedPer1	FedPer2
Accuracy	0.7596	0.7548	0.7470	0.7558	0.7549	0.7365	0.7328
AUC	0.8341	0.8287	0.8195	0.8289	0.8273	0.7948	0.7924
Loss	0.5229	0.5417	0.5449	0.5348	0.5391	0.6709	0.6634
RelImp	-	101.29%	97.15%	100.53%	100.98%	95.52%	92.17%

Table 12: Full—factor 2 performance metrics

Metric	Central	FedAvg1	FedAvg2	APFL1	APFL2	FedPer1	FedPer2
Accuracy	0.7554	0.7667	0.7736	0.7671	0.7735	0.7730	0.7823
AUC	0.8295	0.8416	0.8505	0.8417	0.8478	0.8503	0.8602
Loss	0.5255	0.4930	0.4809	0.4928	0.4846	0.4807	0.4667
RelImp	-	99.38%	100.49%	99.77%	99.20%	123.85%	115.46%

Table 13: Full—random noise performance metrics

Metric	Central	FedAvg1	FedAvg2	APFL1	APFL2	FedPer1	FedPer2
Accuracy	0.7423	0.7570	0.7401	0.7472	0.7410	0.7253	0.7325
AUC	0.8149	0.8295	0.8163	0.8233	0.8159	0.7888	0.7954
Loss	0.5568	0.5211	0.5496	0.5294	0.5443	0.7051	0.7374
RelImp	-	96.90%	97.66%	102.44%	99.06%	125.15%	136.16%

Table 14: 10 %—regular performance metrics

Metric	Central	FedAvg1	FedAvg2	APFL1	APFL2	FedPer1	FedPer2
Accuracy	0.7580	0.7437	0.7643	0.7522	0.7607	0.7251	0.7383
AUC	0.8297	0.8204	0.8380	0.8188	0.8322	0.7898	0.8030
Loss	0.5257	0.5339	0.5101	0.5360	0.5189	0.6933	0.6902
RelImp	-	100.87%	98.73%	100.58%	100.53%	151.58%	136.63%

Table 15: 10 %—factor 2 performance metrics

Metric	Central	FedAvg1	FedAvg2	APFL1	APFL2	FedPer1	FedPer2
Accuracy	0.7430	0.7422	0.7590	0.7456	0.7579	0.7100	0.7294
AUC	0.8170	0.8154	0.8316	0.8136	0.8291	0.7735	0.7904
Loss	0.5565	0.5433	0.5243	0.5530	0.5225	0.7199	0.7134
RelImp	-	100.16%	99.97%	99.78%	99.56%	125.39%	134.48%

Table 16: 10 %—random noise performance metrics

Architecture	Accuracy	AUC	Loss	RelImp
Central	75.67%	82.96%	0.5256	-
FedAvg	75.59%	82.91%	0.5357	99.48%
APFL	75.47%	82.74%	0.5354	100.16%
FedPer	73.27%	79.20%	0.6992	124.50%

Table 17: Median summary statistics

Architecture	Accuracy	AUC	Loss	RelImp
Central	75.31%	82.67%	0.5352	-
FedAvg	75.47%	82.89%	0.5268	99.29%
APFL	75.51%	82.75%	0.5280	100.28%
FedPer	73.77%	80.10%	0.6684	118.42%

Table 18: Average summary statistics

Architecture	Accuracy	AUC	Loss	RelImp
Central	76.00%	83.47%	0.5568	-
FedAvg	77.36%	85.05%	0.5496	101.29%
APFL	77.35%	84.78%	0.5530	102.44%
FedPer	78.23%	86.02%	0.7475	151.58%

Table 19: Max summary statistics

Architecture	Accuracy	AUC	Loss	RelImp
Central	74.23%	81.49%	0.5229	-
FedAvg	74.01%	81.54%	0.4809	96.90%
APFL	74.10%	81.36%	0.4846	99.06%
FedPer	71.00%	77.35%	0.4667	91.89%

Table 20: Min summary statistics

Architecture	Accuracy	AUC	Loss	RelImp
Central	0.82%	0.86%	0.02	-
FedAvg	1.04%	1.06%	0.02	1.43%
APFL	0.90%	0.99%	0.02	0.97%
FedPer	2.03%	2.64%	0.09	20.69%

Table 21: SD summary statistics

Architecture	Accuracy	AUC	Loss
FedAvg	-0.11%	-0.06%	1.88%
APFL	-0.27%	-0.27%	1.83%
FedPer	-3.28%	-4.75%	24.83%

Table 22: Median vs 'central'

Architecture	Accuracy	AUC	Loss
FedAvg	1.76%	1.86%	-1.31%
APFL	1.75%	1.55%	-0.69%
FedPer	2.85%	2.96%	25.51%

Table 23: Max vs 'central'

Architecture	Accuracy	AUC	Loss
FedAvg	-0.30%	0.06%	-8.73%
APFL	-0.18%	-0.16%	-7.90%
FedPer	-4.55%	-5.35%	-12.04%

Table 24: Min vs 'central'

Architecture	Accuracy	AUC	Loss
FedAvg	20.95%	19.15%	23.75%
APFL	9.06%	13.03%	19.03%
FedPer	59.47%	67.49%	82.33%

Table 25: SD vs 'central'

Architecture	Median	Average	Max	Min	SD
APFL	0.68%	0.99%	1.12%	2.18%	-47%
FedPer	20.10%	16.16%	33.18%	-5.45%	93%

Table 26: FedAvg vs RelImp

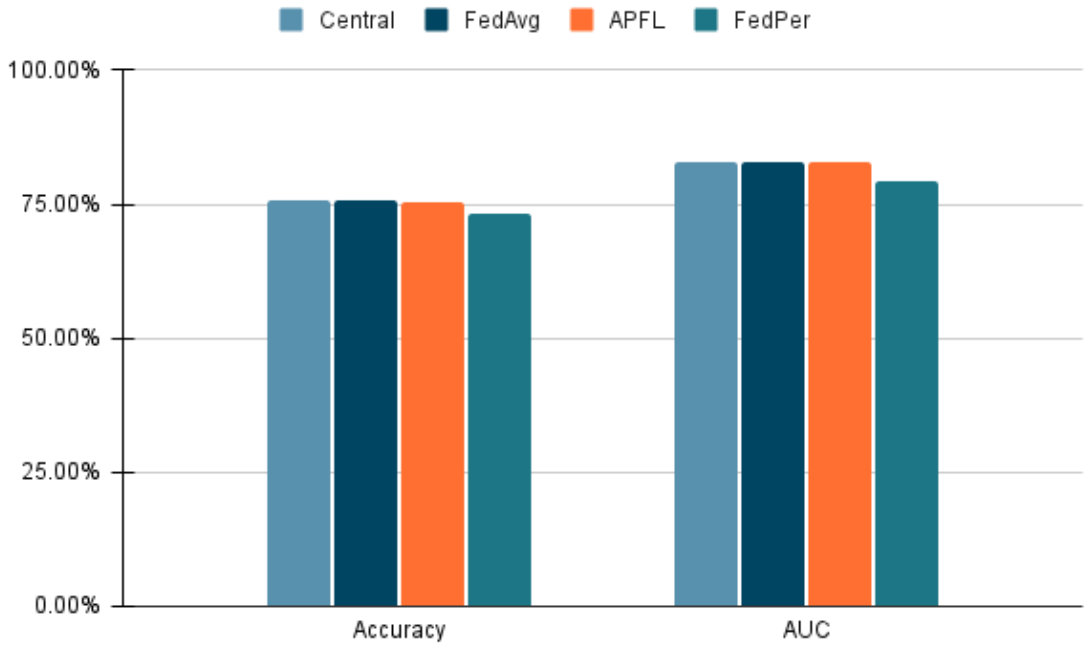


Figure 30: Median accuracy and AUC

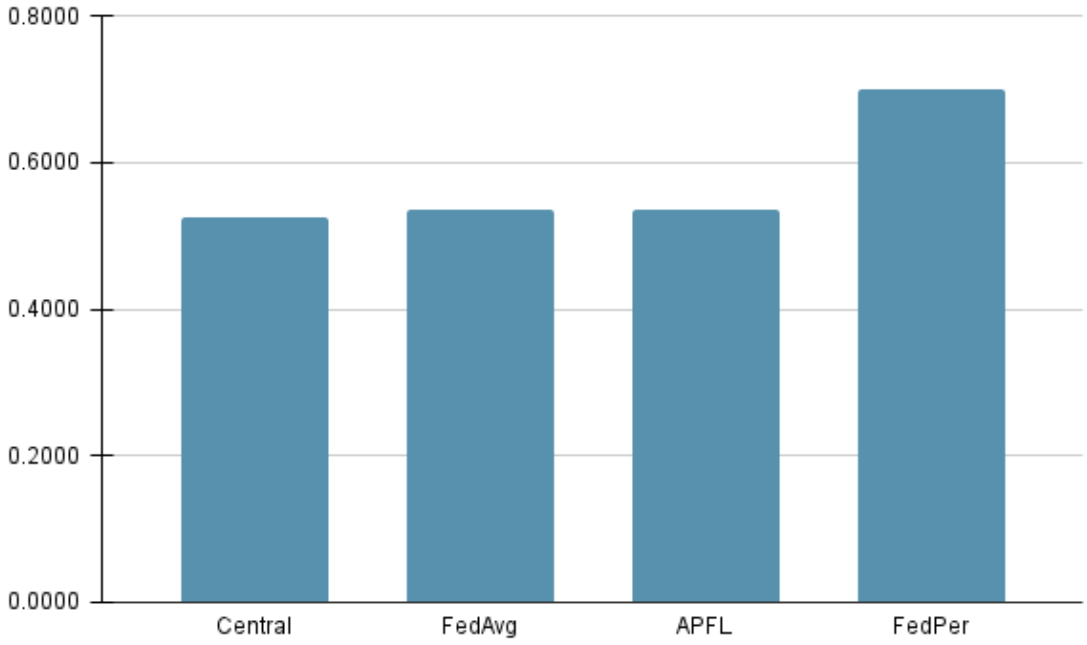


Figure 31: Median loss

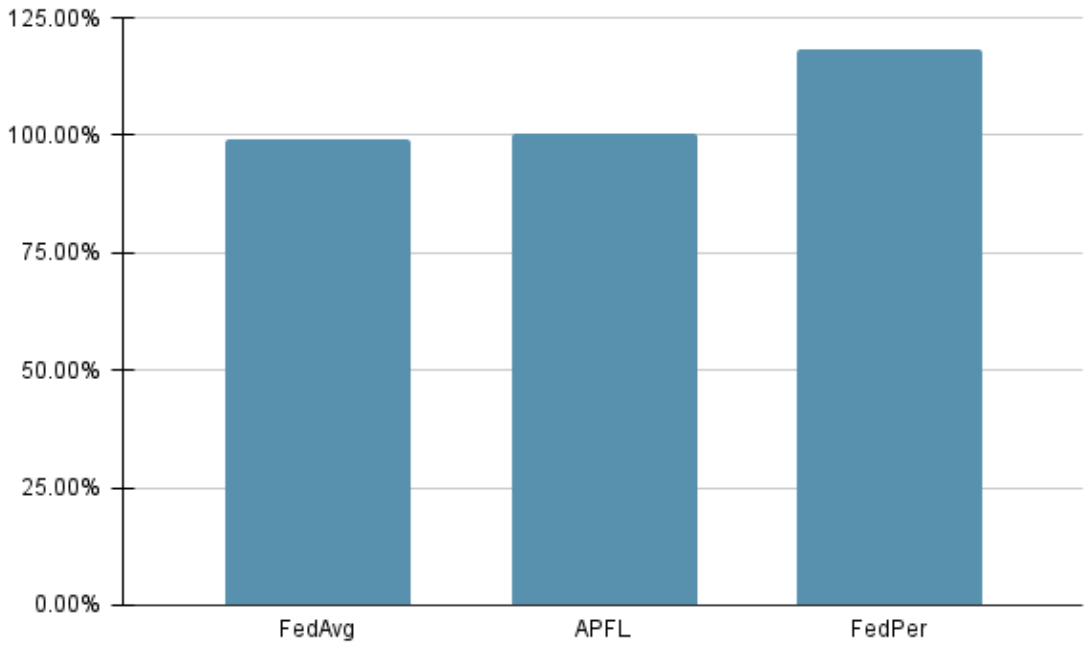


Figure 32: Median RelImp

A.8 Raw Results

ArchitectureVariant-Dataset-DataMod	Accuracy	AUC	Loss	RelImp
Central-Full-Regular	0.76	0.8347	0.5235	100%
Central-Full-Factor 2	0.7596	0.8341	0.5229	100%
Central-Full-Random Noise	0.7554	0.8295	0.5255	100%
Central-10%-Regular	0.7423	0.8149	0.5568	100%
Central-10%-Factor 2	0.758	0.8297	0.5257	100%
Central-10%-Random Noise	0.743	0.817	0.5565	100%
FedAvg1-Full-Regular	0.7508	0.8246	0.5409	99.27%
FedAvg1-Full-Factor 2	0.7548	0.8287	0.5417	101.29%
FedAvg1-Full-Random Noise	0.7667	0.8416	0.493	99.38%
FedAvg1-10%-Regular	0.757	0.8295	0.5211	96.90%
FedAvg1-10%-Factor 2	0.7437	0.8204	0.5339	100.87%
FedAvg1-10%-Random Noise	0.7422	0.8154	0.5433	100.16%
FedAvg2-Full-Regular	0.7574	0.8307	0.5374	99.57%
FedAvg2-Full-Factor 2	0.747	0.8195	0.5449	97.15%
FedAvg2-Full-Random Noise	0.7736	0.8505	0.4809	100.49%
FedAvg2-10%-Regular	0.7401	0.8163	0.5496	97.66%
FedAvg2-10%-Factor 2	0.7643	0.838	0.5101	98.73%
FedAvg2-10%-Random Noise	0.759	0.8316	0.5243	99.97%
APFL1-Full-Regular	0.7544	0.8274	0.5385	101.25%
APFL1-Full-Factor 2	0.7558	0.8289	0.5348	100.53%
APFL1-Full-Random Noise	0.7671	0.8417	0.4928	99.77%
APFL1-10%-Regular	0.7472	0.8233	0.5294	102.44%
APFL1-10%-Factor 2	0.7522	0.8188	0.536	100.58%
APFL1-10%-Random Noise	0.7456	0.8136	0.553	99.78%
APFL2-Full-Regular	0.7511	0.8241	0.5422	99.63%
APFL2-Full-Factor 2	0.7549	0.8273	0.5391	100.98%
APFL2-Full-Random Noise	0.7735	0.8478	0.4846	99.20%
APFL2-10%-Regular	0.741	0.8159	0.5443	99.06%
APFL2-10%-Factor 2	0.7607	0.8322	0.5189	100.53%
APFL2-10%-Random Noise	0.7579	0.8291	0.5225	99.56%
FedPer1-Full-Regular	0.7284	0.7819	0.7475	91.89%
FedPer1-Full-Factor 2	0.7365	0.7948	0.6709	95.52%
FedPer1-Full-Random Noise	0.773	0.8503	0.4807	123.85%
FedPer1-10%-Regular	0.7253	0.7888	0.7051	125.15%
FedPer1-10%-Factor 2	0.7251	0.7898	0.6933	151.58%
FedPer1-10%-Random Noise	0.71	0.7735	0.7199	125.39%
FedPer2-Full-Regular	0.7384	0.7915	0.7323	92.80%
FedPer2-Full-Factor 2	0.7328	0.7924	0.6634	92.17%
FedPer2-Full-Random Noise	0.7823	0.8602	0.4667	115.46%
FedPer2-10%-Regular	0.7325	0.7954	0.7374	136.16%
FedPer2-10%-Factor 2	0.7383	0.803	0.6902	136.63%

FedPer2-10%-Random Noise	0.7294	0.7904	0.7134	134.48%

Table 27: Raw results

References

[1] A. Trask, E. Bluemke, B. Garfinkel, C. G. Cuervas-Mons, and A. Dafoe, "Beyond Privacy Trade-offs with Structured Transparency," Dec. 2020, arXiv: 2012.08347. [Online]. Available: https://arxiv.org/abs/2012.08347v1 (visited on 09/01/2023).

[2] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated Learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 13, no. 3, pp. 1–207, 2019, Publisher: Morgan and Claypool Publishers, ISSN: 19394616. DOI: 10.2200/S00960ED2V01Y201910AIM043. (visited on 11/15/2023).

[3] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," vol. 216, p. 106 775, 2021. DOI: 10.1016/j.knosys.2021.106775. [Online]. Available: https://doi.org/10.1016/j.knosys.2021.106775 (visited on 09/11/2023).

[4] R. Shokri and V. Shmatikov, "Privacy-Preserving Deep Learning," 2015, ISBN: 9781450338325. DOI: 10.1145/2810103.2813687. [Online]. Available: http://dx.doi.org/10.1145/2810103.2813687. (visited on 09/21/2023).

[5] A. J. Brown, "'Should I Stay or Should I Leave?': Exploring (Dis)continued Facebook Use After the Cambridge Analytica Scandal," *https://doi.org/10.1177/2056305120913884*, vol. 6, no. 1, Mar. 2020, Publisher: SAGE PublicationsSage UK: London, England, ISSN: 20563051. DOI: 10.1177/2056305120913884. [Online]. Available: https://journals.sagepub.com/doi/full/10.1177/2056305120913884 (visited on 09/01/2023).

[6] R. Cummings and D. Desai, "The Role of Differential Privacy in GDPR Compliance Position Paper," 2018. DOI: 10.1145/nnnnnnn.nnnnnnn. [Online]. Available: https://www.semanticscholar.org/paper/The-Role-of-Differential-Privacy-in-GDPR-Compliance-Cummings-Desai/a277447b77a70ada6b343e74b6cd5974ea54a845 (visited on 09/01/2023).

[7] S. Zohra, E. Mestari, G. Lenzini, and H. Demirci, "Preserving data privacy in machine learning systems," *Computers & Security*, vol. 137, p. 103 605, 2024. DOI: 10.1016/j.cose.2023.103605. [Online]. Available: https://doi.org/10.1016/j.cose.2023.103605 (visited on 12/26/2023).

[8] K. Bonawitz, V. Ivanov, B. Kreuter, *et al.*, "Practical secure aggregation for privacy-preserving machine learning," *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 1175–1191, Oct. 2017, Publisher: Association for Computing Machinery ISBN: 9781450349468, ISSN: 15437221. DOI: 10.1145/3133956.3133982. [Online]. Available: https://dl.acm.org/doi/10.1145/3133956.3133982 (visited on 11/04/2023).

[9] Q. Li, Z. Wen, Z. Wu, *et al.*, "A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, pp. 3347–3366, Apr. 2023, arXiv:1907.09693 [cs, stat], ISSN: 1041-4347, 1558-2191, 2326-3865. DOI: 10.1109/TKDE.2021.3124599. [Online]. Available: http://arxiv.org/abs/1907.09693 (visited on 01/12/2024).

[10] Q. Li, Z. Wen, and B. He, *Practical Federated Gradient Boosting Decision Trees*, arXiv:1911.04206 [cs, stat], Dec. 2019. [Online]. Available: http://arxiv.org/abs/1911.04206 (visited on 01/13/2024).

[11] W. Li, Q. Xia, H. Cheng, K. Xue, and S.-T. Xia, *Vertical Semi-Federated Learning for Efficient Online Advertising*, arXiv:2209.15635 [cs], Jul. 2022. [Online]. Available: http://arxiv.org/abs/2209.15635 (visited on 02/15/2024).

[12] C. Su, J. Wei, Y. Lei, and J. Li, "A federated learning framework based on transfer learning and knowledge distillation for targeted advertising," en, *PeerJ Computer Science*, vol. 9, e1496, Aug. 2023, Publisher: PeerJ Inc., ISSN: 2376-5992. DOI: 10.7717/peerj-cs.1496. [Online]. Available: https://peerj.com/articles/cs-1496 (visited on 02/15/2024).

[13] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, *Three Approaches for Personalization with Applications to Federated Learning*, arXiv:2002.10619 [cs, stat], Jul. 2020. [Online]. Available: http://arxiv.org/abs/2002.10619 (visited on 01/21/2024).

[14] A. Ziller, A. Trask, A. Lopardo, *et al.*, "PySyft: A Library for Easy Federated Learning," en, in *Federated Learning Systems: Towards Next-Generation AI*, M. H. u. Rehman and M. M. Gaber, Eds., Cham: Springer International Publishing, 2021, pp. 111–139, ISBN: 978-3-030-70604-3. DOI: 10.1007/978-3-030-70604-3_5. [Online]. Available: https://doi.org/10.1007/978-3-030-70604-3_5 (visited on 03/22/2024).

[15] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless Monster: Exploring the Ecosystem of Web-based Device Fingerprinting," 2013. DOI: 10.1109/SP.2013.43. [Online]. Available: http://www.iovation.com (visited on 09/02/2023).

[16] A. Hevner and S. Chatterjee, "Design Science Research in Information Systems," pp. 9–22, 2010, Publisher: Springer, Boston, MA. DOI: 10.1007/978-1-4419-5653-8_2. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4419-5653-8_2 (visited on 09/21/2023).

[17] T. Ryffel, A. Trask, M. Dahl, *et al.*, "A generic framework for privacy preserving deep learning," 2018, arXiv: 1811.04017v2. (visited on 09/01/2023).

[18] S. Geyik, A. Dasdan, and K.-C. Lee, "User Clustering in Online Advertising via Topic Models," Jan. 2015.

[19] C. Borgs, J. Chayes, N. Immorlica, K. Jain, O. Etesami, and M. Mahdian, "Dynamics of bid optimization in online advertisement auctions," en, in *Proceedings of the 16th international conference on World Wide Web*, Banff Alberta Canada: ACM, May 2007, pp. 531–540, ISBN: 978-1-59593-654-7. DOI: 10.1145/1242572.1242644. [Online]. Available: https://dl.acm.org/doi/10.1145/1242572.1242644 (visited on 01/20/2024).

[20] J. Rhuggenaath, A. Akcay, Y. Zhang, and U. Kaymak, "Optimal display-ad allocation with guaranteed contracts and supply side platforms," en, *Computers & Industrial Engineering*, vol. 137, p. 106 071, Nov. 2019, ISSN: 03608352. DOI: 10.1016/j.cie.2019.106071. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0360835219305303 (visited on 01/20/2024).

[21] K.-c. Lee, B. Orten, A. Dasdan, and W. Li, "Estimating conversion rate in display advertising from past erformance data," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '12, New York, NY, USA: Association for Computing Machinery, Aug. 2012, pp. 768–776, ISBN: 978-1-4503-1462-6. DOI: 10.1145/2339530.2339651. [Online]. Available: https://doi.org/10.1145/2339530.2339651 (visited on 01/20/2024).

[22] A. Epasto, A. Munoz Medina, S. Avery, *et al.*, "Clustering for Private Interest-based Advertising," en, in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, Virtual Event Singapore: ACM, Aug. 2021, pp. 2802–2810, ISBN: 978-1-4503-8332-5. DOI: 10.1145/3447548.3467180. [Online]. Available: https://dl.acm.org/doi/10.1145/3447548.3467180 (visited on 01/20/2024).

[23] J. Yan, N. Liu, G. Wang, W. Zhang, Y. Jiang, and Z. Chen, "How much can behavioral targeting help online advertising?" en, in *Proceedings of the 18th international conference on World wide web*, Madrid Spain: ACM, Apr. 2009, pp. 261–270, ISBN: 978-1-60558-487-4. DOI: 10.1145/1526709.1526745. [Online]. Available: https://dl.acm.org/doi/10.1145/1526709.1526745 (visited on 01/20/2024).

[24] M. Kretschmer, J. Pennekamp, and K. Wehrle, "Cookie Banners and Privacy Policies: Measuring the Impact of the GDPR on the Web," *ACM Transactions on the Web*, vol. 15, p. 42, 2021. DOI: 10.1145/3466722. [Online]. Available: https://doi.org/10.1145/3466722 (visited on 09/01/2023).

[25] D. E. Bartholomew and M. Williamson, "Retail media networks," *Journal of Retailing and Consumer Services*, vol. 69, p. 103 119, Nov. 2022, ISSN: 0969-6989. DOI: 10.1016/j.jretconser.2022.103119. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0969698922002120 (visited on 01/12/2024).

[26] M.-C. Puffett and H. Mussard, "IAB Europe Guide to the Post Third-Party Cookie Era," Tech. Rep., 2020.

[27] Datatilsynet, "Artificial intelligence and privacy," Norwegian Data Protection Authority, Tech. Rep., 2018. (visited on 09/01/2023).

[28] D. Fujs, A. Mihelic, and S. Vrhovec, "Social Network Self-Protection Model: What Motivates Users to Self-Protect?" *Journal of Cyber Security and Mobility*, vol. 8, no. 4, pp. 467–492–467–492, Jan. 2019, Publisher: River Publishers, ISSN: 2245-4578. DOI: 10.13052/2245-1439.844. [Online]. Available: https://journals.riverpublishers.com/index.php/JCSANDM/article/view/5365 (visited on 09/01/2023).

[29] A. Narayanan and V. Shmatikov, "Robust De-anonymization of Large Datasets (How to Break Anonymity of the Netflix Prize Dataset)," 2008. (visited on 12/19/2023).

[30] S. Earley, "The Role of a Customer Data Platform," 2018. DOI: 10.1109/MITP.2018.011301803. [Online]. Available: https://www.researchgate.net/publication/323196693 (visited on 11/23/2023).

[31] IAB Technology Laboratory, "Data Clean Rooms Guidance and Recommended Practices," Tech. Rep., 2023. [Online]. Available: https://iabtechlab.com/datacleanrooms/.

[32] U. Majeed, L. U. Khan, A. Yousafzai, Z. Han, B. J. Park, and C. S. Hong, "ST-BFL: A Structured Transparency Empowered Cross-Silo Federated Learning on the Blockchain Framework," en, *IEEE Access*, vol. 9, pp. 155 634–155 650, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3128622. [Online]. Available: https://ieeexplore.ieee.org/document/9617624/ (visited on 04/26/2024).

[33] H. Nissenbaum, *Privacy in Context: Technology, Policy, and the Integrity of Social Life*. Stanford: Stanford University Press, 2009, ISBN: 978-0-8047-5236-7.

[34] Y. Lindell, "Secure Multiparty Computation (MPC)," *Cryptology ePrint Archive*, vol. 64, no. 1, pp. 86–98, Jan. 2020, Publisher: Association for Computing Machinery, ISSN: 15577317. DOI: 10.1145/3387108. (visited on 12/26/2023).

[35] R. L. Rivest and M. Dertouzos, "ON DATA BANKS AND PRIVACY HOMOMORPHISMS," 1978. [Online]. Available: https://www.semanticscholar.org/paper/ON-DATA-BANKS-AND-PRIVACY-HOMOMORPHISMS-Rivest-Dertouzos/c365f01d330b2211e74069120e88cff37eacbcf5 (visited on 01/14/2024).

[36] J. Sen, "Homomorphic Encryption: Theory & Applications," in arXiv:1305.5886 [cs], Jul. 2013. DOI: 10.5772/56687. [Online]. Available: http://arxiv.org/abs/1305.5886 (visited on 01/14/2024).

[37] C. Dwork, "Differential privacy: A survey of results," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4978 LNCS, pp. 1–19, 2008, Publisher: Springer Verlag ISBN: 3540792279, ISSN: 16113349. DOI: 10.1007/978-3-540-79228-4_1/COVER. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-79228-4_1 (visited on 09/11/2023).

[38] Apple Differential Privacy Team, "Learning with Privacy at Scale - Apple Machine Learning Research," 2017. (visited on 09/11/2023).

[39] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," Feb. 2016, arXiv: 1602.05629v4. [Online]. Available: http://arxiv.org/abs/1602.05629 (visited on 11/13/2023).

[40] Google, *Privacy Sandbox Learning Hub*, 2023. [Online]. Available: https://privacysandbox.com/learning-hub/ (visited on 11/01/2023).

[41] M. Gong, Y. Xie, K. Pan, K. Feng, and A. K. Qin, "A Survey on Differentially Private Machine Learning [Review Article]," *IEEE Computational Intelligence Magazine*, vol. 15, no. 2, pp. 49–64, May 2020, Publisher: Institute of Electrical and Electronics Engineers Inc., ISSN: 15566048. DOI: 10.1109/MCI.2020.2976185. [Online]. Available: https://fliphtml5.com/rtmqr/kknk/IEEE_Computational_Intelligence_Magazine_-_May_2020/51/ (visited on 01/11/2024).

[42] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning Differentially Private Language Models Without Losing Accuracy," *arXiv.org*, 2017. (visited on 11/03/2023).

[43] A. Hard, K. Rao, R. Mathews, *et al.*, "Federated Learning for Mobile Keyboard Prediction," Nov. 2018, arXiv: 1811.03604v1. [Online]. Available: https://www.researchgate.net/publication/328825912_Federated_Learning_for_Mobile_Keyboard_Prediction (visited on 09/01/2023).

[44] P. Kairouz, H. B. McMahan, B. Avent, *et al.*, *Advances and Open Problems in Federated Learning*, arXiv:1912.04977 [cs, stat], Mar. 2021. [Online]. Available: http://arxiv.org/abs/1912.04977 (visited on 01/12/2024).

[45] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar, *Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data*, arXiv:1610.05755 [cs, stat], Mar. 2017. [Online]. Available: http://arxiv.org/abs/1610.05755 (visited on 01/14/2024).

[46] T. Qi, F. Wu, C. Wu, Y. Huang, and X. Xie, "Privacy-Preserving News Recommendation Model Learning," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, T. Cohn, Y. He, and Y. Liu, Eds., Online: Association for Computational Linguistics, Nov. 2020, pp. 1423–1432. DOI: 10.18653/v1/2020.findings-emnlp.128. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.128 (visited on 01/14/2024).

[47] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, *Federated Multi-Task Learning*, arXiv:1705.10467 [cs, stat], Feb. 2018. [Online]. Available: http://arxiv.org/abs/1705.10467 (visited on 01/16/2024).

[48] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated Learning with Non-IID Data," 2018, arXiv:1806.00582 [cs, stat]. DOI: 10.48550/arXiv.1806.00582. [Online]. Available: http://arxiv.org/abs/1806.00582 (visited on 01/15/2024).

[49] H. Jiang, M. Liu, B. Yang, Q. Liu, J. Li, and X. Guo, "Customized Federated Learning for accelerated edge computing with heterogeneous task targets," *Computer Networks*, vol. 183, p. 107 569, Dec. 2020, ISSN: 1389-1286. DOI: 10.1016/j.comnet.2020.107569. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128620312135 (visited on 01/17/2024).

[50] H. Brendan McMahan Eider Moore Daniel Ramage Seth Hampson Blaise AgüeraAg and A. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," 2017, arXiv: 1602.05629v4. (visited on 11/15/2023).

[51] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, *D\"IoT: A Federated Self-learning Anomaly Detection System for IoT*, arXiv:1804.07474 [cs], May 2019. [Online]. Available: http://arxiv.org/abs/1804.07474 (visited on 01/17/2024).

[52] J. Zhang and Z. Li, "A Clustered Federated Learning Method of User Behavior Analysis Based on Non-IID Data," en, *Electronics*, vol. 12, no. 7, p. 1660, Jan. 2023, Number: 7 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2079-9292. DOI: 10.3390/electronics12071660. [Online]. Available: https://www.mdpi.com/2079-9292/12/7/1660 (visited on 02/14/2024).

[53] WeBank AI Department, "Federated learning white paper v1. 0," *WeBank Group*, vol. 9, 2018.

[54] V. Zantedeschi, A. Bellet, and M. Tommasi, *Fully Decentralized Joint Learning of Personalized Models and Collaboration Graphs*, arXiv:1901.08460 [cs, stat], Mar. 2020. [Online]. Available: http://arxiv.org/abs/1901.08460 (visited on 01/13/2024).

[55] K. Chang, N. Balachandar, C. K. Lam, *et al.*, *Institutionally Distributed Deep Learning Networks*, arXiv:1709.05929 [physics], Sep. 2017. [Online]. Available: http://arxiv.org/abs/1709.05929 (visited on 01/13/2024).

[56] S. Savazzi, M. Nicoli, and V. Rampa, "Federated Learning with Cooperating Devices: A Consensus Approach for Massive IoT Networks," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4641–4654, May 2020, arXiv:1912.13163 [cs, eess], ISSN: 2327-4662, 2372-2541. DOI: 10.1109/JIOT.2020.2964162. [Online]. Available: http://arxiv.org/abs/1912.13163 (visited on 01/17/2024).

[57] B. Minto, *The Minto Pyramid Principle: Logic in Writing, Thinking, & Problem Solving*, English, Expanded edition. London: Minto Intl, Jan. 1996, ISBN: 978-0-9601910-3-1.

[58] X. Liu, B. Twardowski, and T. K. Wijaya, *Online Meta-Learning for Model Update Aggregation in Federated Learning for Click-Through Rate Prediction*, arXiv:2209.00629 [cs], Aug. 2022. [Online]. Available: http://arxiv.org/abs/2209.00629 (visited on 03/28/2024).

[59] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards Personalized Federated Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 12, pp. 9587–9603, Dec. 2023, arXiv:2103.00710 [cs], ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2022.3160699. [Online]. Available: http://arxiv.org/abs/2103.00710 (visited on 02/14/2024).

[60] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, *SCAFFOLD: Stochastic Controlled Averaging for Federated Learning*, arXiv:1910.06378 [cs, math, stat], Apr. 2021. [Online]. Available: http://arxiv.org/abs/1910.06378 (visited on 04/05/2024).

[61] C. T. Dinh, N. H. Tran, and T. D. Nguyen, *Personalized Federated Learning with Moreau Envelopes*, arXiv:2006.08848 [cs, stat], Jan. 2022. [Online]. Available: http://arxiv.org/abs/2006.08848 (visited on 01/21/2024).

[62] P. P. Liang, T. Liu, L. Ziyin, *et al.*, *Think Locally, Act Globally: Federated Learning with Local and Global Representations*, arXiv:2001.01523 [cs, stat], Jul. 2020. [Online]. Available: http://arxiv.org/abs/2001.01523 (visited on 01/15/2024).

[63] Y. Deng, M. M. Kamani, and M. Mahdavi, *Adaptive Personalized Federated Learning*, en, arXiv:2003.13461 [cs, stat], Nov. 2020. [Online]. Available: http://arxiv.org/abs/2003.13461 (visited on 01/21/2024).

[64] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing Federated Learning on Non-IID Data with Reinforcement Learning," en, in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, Toronto, ON, Canada: IEEE, Jul. 2020, pp. 1698–1707, ISBN: 978-1-72816-412-0. DOI: 10.1109/INFOCOM41043.2020.9155494. [Online]. Available: https://ieeexplore.ieee.org/document/9155494/ (visited on 04/08/2024).

[65] D. Li and J. Wang, *FedMD: Heterogenous Federated Learning via Model Distillation*, arXiv:1910.03581 [cs, stat], Oct. 2019. [Online]. Available: http://arxiv.org/abs/1910.03581 (visited on 03/18/2024).

[66] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 3557–3568. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/24389bfe4fe2eba8bf9aa9203a44cdad-Abstract.html (visited on 04/08/2024).

[67] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, *Federated Learning with Personalization Layers*, arXiv:1912.00818 [cs, stat], Dec. 2019. [Online]. Available: http://arxiv.org/abs/1912.00818 (visited on 01/21/2024).

[68] Z. Ren, L. Yang, and K. Chen, "Improving Availability of Vertical Federated Learning: Relaxing Inference on Non-overlapping Data," *ACM Transactions on Intelligent Systems and Technology*, vol. 13, no. 4, 58:1–58:20, Jun. 2022, ISSN: 2157-6904. DOI: 10.1145/3501817. [Online]. Available: https://doi.org/10.1145/3501817 (visited on 04/05/2024).

[69] W. Li, Q. Xia, J. Deng, *et al.*, *VFed-SSD: Towards Practical Vertical Federated Advertising*, arXiv:2205.15987 [cs], Jun. 2022. [Online]. Available: http://arxiv.org/abs/2205.15987 (visited on 03/26/2024).

[70] Z. Liu, Y. Hui, and F. Peng, *Group Personalized Federated Learning*, arXiv:2210.01863 [cs, stat], Oct. 2022. DOI: 10.48550/arXiv.2210.01863. [Online]. Available: http://arxiv.org/abs/2210.01863 (visited on 04/08/2024).

[71] S. Wan, D. Gao, H. Gu, and D. Hu, "FedPDD: A Privacy-preserving Double Distillation Framework for Cross-silo Federated Recommendation," *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, Jun. 2023, Conference Name: 2023 International Joint Conference on Neural Networks (IJCNN) ISBN: 9781665488679 Place: Gold Coast, Australia Publisher: IEEE. DOI: 10.1109/IJCNN54540.2023.10191850. [Online]. Available: https://ieeexplore.ieee.org/document/10191850/ (visited on 03/27/2024).

[72] Y. He, Y. Kang, X. Zhao, *et al.*, *A Hybrid Self-Supervised Learning Framework for Vertical Federated Learning*, arXiv:2208.08934 [cs], Jun. 2022. DOI: 10.48550/arXiv.2208.08934. [Online]. Available: http://arxiv.org/abs/2208.08934 (visited on 04/05/2024).

[73] X. Rong, J. Zhu, and H. Xi, "A Collaborative Framework for Ad Click-Through Rate Prediction in Mobile App Services," en, in *Service-Oriented Computing*, J. Troya, B. Medjahed, M. Piattini, L. Yao, P. Fernández, and A. Ruiz-Cortés, Eds., Cham: Springer Nature Switzerland, 2022, pp. 567–574, ISBN: 978-3-031-20984-0. DOI: 10.1007/978-3-031-20984-0_40.

[74] P. Wei, H. Dou, S. Liu, *et al.*, *FedAds: A Benchmark for Privacy-Preserving CVR Estimation with Vertical Federated Learning*, arXiv:2305.08328 [cs], May 2023. [Online]. Available: http://arxiv.org/abs/2305.08328 (visited on 03/29/2024).

[75] M. Hejazinia, D. Huba, I. Leontiadis, *et al.*, *FEL: High Capacity Learning for Recommendation and Ranking via Federated Ensemble Learning*, arXiv:2206.03852 [cs], Jun. 2022. [Online]. Available: http://arxiv.org/abs/2206.03852 (visited on 03/29/2024).

[76] J. Bian, J. Huang, S. Ji, *et al.*, "*Feynman* : Federated Learning-Based Advertising for Ecosystems-Oriented Mobile Apps Recommendation," en, *IEEE Transactions on Services Computing*, vol. 16, no. 5, pp. 3361–3372, Sep. 2023, ISSN: 1939-1374, 2372-0204. DOI: 10.1109/TSC.2023.3285935. [Online]. Available: https://ieeexplore.ieee.org/document/10154552/ (visited on 02/15/2024).

[77] J. Xu, X. Tong, and S.-L. Huang, *Personalized Federated Learning with Feature Alignment and Classifier Collaboration*, arXiv:2306.11867 [cs], Jun. 2023. [Online]. Available: http://arxiv.org/abs/2306.11867 (visited on 02/14/2024).

[78] L. Yan, W.-J. Li, G.-R. Xue, and D. Han, "Coupled Group Lasso for Web-Scale CTR Prediction in Display Advertising," en, in *Proceedings of the 31st International Conference on Machine Learning*, ISSN: 1938-7228, PMLR, Jun. 2014, pp. 802–810. [Online]. Available: https://proceedings.mlr.press/v32/yan14.html (visited on 03/11/2024).

[79] A. Hall, M. Jay, T. Cebere, *et al.*, *Syft 0.5: A Platform for Universally Deployable Structured Transparency*. Apr. 2021.

[80] M. Naumov, D. Mudigere, H.-J. M. Shi, *et al.*, *Deep Learning Recommendation Model for Personalization and Recommendation Systems*, arXiv:1906.00091 [cs], May 2019. [Online]. Available: http://arxiv.org/abs/1906.00091 (visited on 04/16/2024).

[81] D. Donoho, "High-Dimensional Data Analysis: The Curses and Blessings of Dimensionality," *AMS Math Challenges Lecture*, pp. 1–32, Jan. 2000.

[82] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, 2nd ed. New York: Routledge, Jul. 1988, ISBN: 978-0-203-77158-7. DOI: 10.4324/9780203771587.

[83] D. Lakens, "Calculating and reporting effect sizes to facilitate cumulative science: A practical primer for t-tests and ANOVAs," *Frontiers in Psychology*, vol. 4, p. 863, Nov. 2013, ISSN: 1664-1078. DOI: 10.3389/fpsyg.2013.00863. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3840331/ (visited on 04/23/2024).

[84] J. Sun, Y. Yao, W. Gao, J. Xie, and C. Wang, *Defending against Reconstruction Attack in Vertical Federated Learning*, arXiv:2107.09898 [cs], Jul. 2021. [Online]. Available: http://arxiv.org/abs/2107.09898 (visited on 03/28/2024).

[85] A. Mahendran and A. Vedaldi, *Understanding Deep Image Representations by Inverting Them*, arXiv:1412.0035 [cs], Nov. 2014. [Online]. Available: http://arxiv.org/abs/1412.0035 (visited on 04/01/2024).

[86] L. Zhu, Z. Liu, and S. Han, *Deep Leakage from Gradients*, arXiv:1906.08935 [cs, stat], Dec. 2019. [Online]. Available: http://arxiv.org/abs/1906.08935 (visited on 04/01/2024).

[87] A. Mayer, M. Niemietz, V. Mladenov, and J. Schwenk, "Guardians of the Clouds: When Identity Providers Fail," in *Proceedings of the 6th edition of the ACM Workshop on Cloud Computing Security*, ser. CCSW '14, New York, NY, USA: Association for Computing Machinery, Nov. 2014, pp. 105–116, ISBN: 978-1-4503-3239-2. DOI: 10.1145/2664168.2664171. [Online]. Available: https://doi.org/10.1145/2664168.2664171 (visited on 01/12/2024).

[88] K. Corre, O. Barais, G. Sunye, V. Frey, and J.-M. Crom, "Why can't users choose their identity providers on the web?" *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 3, pp. 72–86, Jan. 2017, Publisher: Privacy Enhancing Technologies Symposium. DOI: 10.1515/popets-2017-0029. [Online]. Available: https://hal.science/hal-01611048 (visited on 01/12/2024).

[89] T. Herbrich, "Data Clean Rooms," *Computer Law Review International*, vol. 23, no. 4, pp. 109–120, Aug. 2022, Publisher: Verlag Dr. Otto Schmidt. DOI: 10.9785/CRI-2022-230404/HTML. (visited on 10/28/2023).

[90] M. Mihajlija, *Prepare for phasing out third-party cookies - Chrome Developers*, 2023. [Online]. Available: https://developer.chrome.com/en/docs/privacy-sandbox/third-party-cookie-phase-out/ (visited on 09/01/2023).

[91] P. Yadav, A. Feraudo, B. Arief, S. F. Shahandashti, and V. G. Vassilakis, "Position paper: A systematic framework for categorising IoT device fingerprinting mechanisms; Position paper: A systematic framework for categorising IoT device fingerprinting mechanisms," arXiv: 2010.08466v2. (visited on 11/01/2023).

[92] Z. Ji, Z. C. Lipton, and C. Elkan, "Differential Privacy and Machine Learning: A Survey and Review," 2014, arXiv: 1412.7584v1. (visited on 09/01/2023).

[93] K. Kollnig, A. Shuba, M. Van Kleek, R. Binns, and N. Shadbolt, "Goodbye Tracking? Impact of iOS App Tracking Transparency and Privacy Labels," en, in *2022 ACM Conference on Fairness, Accountability, and Transparency*, arXiv:2204.03556 [cs], Jun. 2022, pp. 508–520. DOI: 10.1145/3531146.3533116. [Online]. Available: http://arxiv.org/abs/2204.03556 (visited on 01/21/2024).

[94] J. Runge and E. Seufert, "Apple Is Changing How Digital Ads Work. Are Advertisers Prepared?" *Harvard Business Review*, Apr. 2021, Section: Sales and marketing, ISSN: 0017-8012. [Online]. Available: https://hbr.org/2021/04/apple-is-changing-how-digital-ads-work-are-advertisers-prepared (visited on 01/18/2024).

[95] M. Thomson, "An Analysis of Apple's Private Click Measurement," en, [Online]. Available: https://mozilla.github.io/ppa-docs/pcm.pdf.

[96] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010, ISSN: 1041-4347. DOI: 10.1109/TKDE.2009.191. [Online]. Available: http://ieeexplore.ieee.org/document/5288526/ (visited on 01/14/2024).

[97] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "Secure Federated Transfer Learning," *IEEE Intelligent Systems*, vol. 35, no. 4, pp. 70–82, Jul. 2018, arXiv:1812.03337 [cs, stat], ISSN: 1541-1672, 1941-1294. DOI: 10.1109/MIS.2020.2988525. [Online]. Available: http://arxiv.org/abs/1812.03337 (visited on 01/16/2024).