Universiteit Leiden

Master Computer Science

Federated Learning: literature review and hyperparameter importance.

Name:	Amin Moradi
Student ID:	s2588862
Date:	13/12/2021
Specialisation:	Computer Science and Advanced Data Analytics
1st supervisor:	Dr. Jan N. van Rijn
2st supervisor:	Dr. Olga Gadyatskaya
External supervisor:	Dr. Albert Wong
External supervisor:	Mark Kroon

Federated Learning: literature review and hyperparameter importance.

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract. Current approaches to centralising data gathering and creating larger data silos have emerged as a prominent approach to building robust, state-of-the-art models. However, centralised data gathering is not always possible due to data regulations on sensitive information like electronic health records. Federated learning can help us overcome such challenges by introducing a decentralised learning paradigm without transferring sensitive data. Under the federated setting, multiple organisations can jointly train a shared model by keeping the data local and only communicating an instance of the shared model. In this work, we will look at the fundamentals of federated learning and provide a literature study of this emerging field. We also designed a benchmarking application to study federated setting hyperparameters and their effect on the model accuracy and convergence.

Keywords: Federated Learning · Distributed Learning · Hyperparameter · Hyperparamete importance · Benchmarking

Acknowledgements

I have been a student most of my life. Having role models from day one will help you pursue science, and I have my parents to thank for. With their extraordinary experiences, my two older brothers taught me to dream big, and I'm grateful for their support all these years. Amir, without a doubt, it would have been impossible to get to this point without your wisdom, encouragement and financial support, and I'm thankful for it. I would like to thank my supervisor Dr Jan N. van Rijn, for his remarkable insights and guidance to complete this work.

Table of Contents

List of Figures

1 Introduction

We have managed to produce more data than ever and breaking records every year [32]. This tremendous flow of data directly affected the progress of machine learning and specially deep learning methods.
Large scale language models like GPT-3 [7] can adapt to software source codes and with its one-shot-learning ability learn various programming languages. On top of that, with the ever-increasing volume of publicly available source codes, machine learning models are surpassing human-level programming.
In the open source community the pace of model improvement is much higher due to the nature of data.
However when we look at other sectors to adapt such methods, we are faced with major data regulations and privacy issues. Without a doubt, regulations like General Data Protection Regulation (GDPR) and Health Insurance Portability and Accountability Act (HIPAA) are necessary and required. Technology is yet to adapt completely to elevate big data benefits in health and medical data.
Federated learning (FL) is a machine learning setting where many clients collaboratively train a model under a federated (central) server's orchestration while keeping the training data decentralised. It embodies focused collection and data minimisation principles and can mitigate many of the systemic privacy risks and costs resulting from traditional, centralised machine learning.
As per definition, "the learning task is solved by a loose federation of participating devices (clients) which are coordinated by a central server" [48].
By keeping data locally and preserving data privacy and regulations, federated learning trains a model across a series of distributed datasets by actively communicating the learned parameters of each dataset and accumulating it on a federated server. This technique creates a shared model without transferring the actual datasets or any data accumulation procedures. Federated learning methods emerge the potential of security and privacy-preserving distributed machine learning and provide a systematic approach for learning on a broader range of datasets. Federated learning has been applied extensively in medicine, personalised health, drug discovery, pandemic analysis, and many more areas where data privacy plays a crucial role or the data is highly distributed.

1.1 Problem Statement & AI in health-care

Without a doubt, machine learning in health and medicine has amplified the research in simulation for drug discovery and automating cumbersome experiments, prediction and prevention in cancer treatment and ailment classification. According to a recent research, only 10% of global health data was responsible for the vast majority of this improvement [56]. Electronic Health Records (EHR) are highly bound by data regulations and can not be shared. There has been several attempts using data augmentation techniques and generative networks [28] to synthesis medical datasets without exposing patient data in a controlled privacy preserving environment, but nonetheless adversarial attacks and distribution distortion [60] has limited the potentials and capabilities of such methods. Data anonymisation and data encryption techniques has also been implemented for sharing private datasets but the limitation in computational power of deep learning and in some cases adversarial data exposure [50] has limited the growth of such techniques [58].
Large data gathering methods and data accumulation techniques by governmental institutes have shown us the great potential of machine learning in health-care. Large-scale genetic databases such as the Cancer Genome Atlas (TCGA) and the UK Biobank with thousands of genetic sequencing data along with numerous other health information such as diseases, state, age of diagnosis, time of death, and much more [8]. Copy number variation (CNV) data from the UK Biobank's roughly 500,000 patients, which does not even contain the raw sequence reads, is almost 2 Terabytes alone in flat text files [64]. However the pace of data gathering is much larger than what we can control today. With the lack of domestic data gathering platforms and eco-systems we can conclude at a global scale it is almost impossible to reach an international and unified convention.
With the emergence of edge computation and Cloud Platforms, distributed machine learning is the most prominent solution to most of the problems described above. Distributed machine learning can simply be translated to keeping the data locally at the edge and only transmitting machine learning model or computation results (gradient values) over a secure and encrypted network connection. By leveraging distributed learning we can train a model in a secure and privacy preserving manner.
Taking advantage of distributed machine learning the centralised data gathering shifts to participants in a training phase which can result in increase in the total amount of data compared to traditional data gathering. Using distributed machine learning we can overcome challenges in data starvation and insufficient generalisation due to data limitation and leverage parallel computing and distributed hardware [70].

Taking advantage of distributed machine learning not only overcomes challenges in data starvation and insufficient generalisation of machine learning models but also contributes to parallelising the computations and distributed hardware [70].
The valuable advantages of standalone distributed machine learning solution unfolds against a backdrop of malicious and untrusted contributors in model training phase. By processing invert gradient and exploiting magnitude-invariant loss it is possible to reconstruct the original data with a very high accuracy even in deep networks [23].
To interpret a secure eco-system for distributed learning, several mathematical and cryptography techniques have been introduced. Secure computation, Homomorphic encryption and Differential Privacy are amongst the most commonly used methods. Encrypted computation enables us to introduce a level of trust between distributed parties and to some extent guarantee data privacy.

1.2 Structure of thesis

Federated learning tackles problems regarding distributed learning and security exposures. In this thesis we will dive deeper into fundamental concepts of federated learning and explore systematic methods and measurements to extend federated learning setting in health-care and specially cross-silo learning where limited number of clients collaborating to train a shared model.
In this thesis we will

- Review what are the available methods and architectures for cross-silo federated learning.
- Design and implement an end-to-end benchmarking software for federated learning evaluation
- Research what are the important experimental hyperparameters in the federated setting
- Study to how these hyperparameters affect each other and the global performance of the federated model.

In section 2, we present an introduction to distributed machine learning and explore federated learning and its algorithmic notations.
Sections 3-3.7 are dedicated to the literature study and exploring a wide range of approaches to federated problem solving and frameworks.
Federated learning can be applied to various domains and in section 3.7 we extend the vanilla implementation and explore the pros and cons of domain specific federated learning methods. Open-source federated learning libraries and frameworks has pushed federated learning research-community to scale and extend their knowledge-base further. In section 3.9 we explore several federated learning frameworks and compare them in terms of use cases and limitations.
To evaluate different federated learning settings we implemented an end-to-end benchmarking source-code using PyTorch and functional ANOVA (fANOVA) [35]. Using fANOVA framework we are able to gain a deep understanding of each hyperparameter importance and various effective factors on federated learning environmental parameters setting. In section 5 we explain our design paradim for the benchmarking source-code and explore how fANOVA works as a complementary framework to help us gain insight into hyperparameter importance. Finally in section 6 we introduces the final conclusions of the experiments and the literature study.

2 Preliminaries

In this section we will take a look into fully decentralised machine learning and study how federated learning emerged as a solution to overcome challenges of decentralised architecture. We will also further examine the basic structure of federated learning and explain the mathematical foundation of federated optimisation.

2.1 An introduction to non-data-centric machine learning

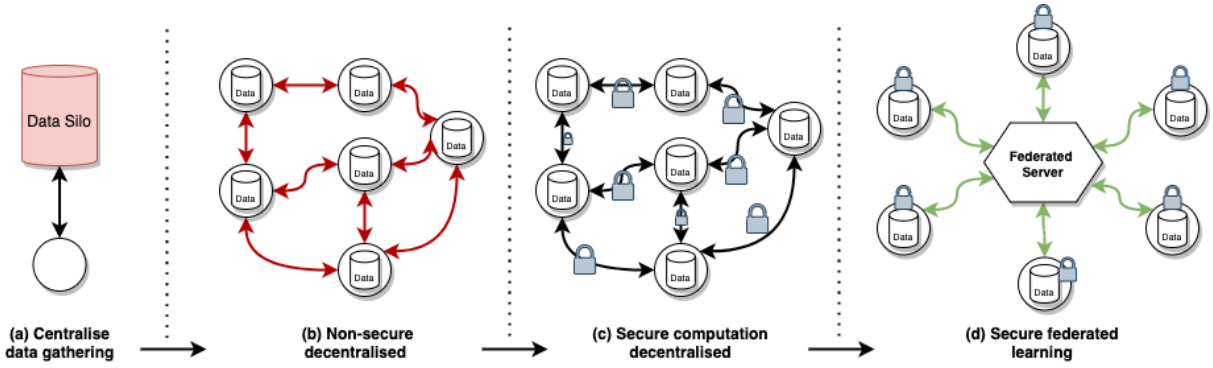


Fig. 1: From centralised data gathering (a) to decentralised federated learning (d). By securing communication and computation layers federated learning provides a robust and privacy preserving learning paradigm.

Artificial intelligence has seen a significant increase in demand over the last decade, both in designing novel methods in machine learning techniques and the ability to use hardware acceleration. However, a large amount of training data is necessary to improve prediction quality and make machine learning and specially deep learning systems practicable for complicated applications. While a limited number of machine learning models may be trained with minimal amounts of data, the input for bigger and deeper networks grows exponentially as the number of parameters is increased [78].

Nowadays, the most common method of utilising machine learning is to gather all data in a central location (figure 1a). After that, the model is trained on powerful servers. However, this data gathering procedure is often intrusive in nature. Many consumers are unwilling to share private data with businesses, which makes machine learning challenging to exploit in some circumstances, such as training on medical and health-care data. Even if privacy is not an issue, the collection of data may be impractical. For example medical institutes and hospitals in Europe can not easily share patient information even if we overlook data regulations. This is mainly because of the scale of data and lack of global standardisation.

Gathering and storing large volumes of data can be costly and adds risk factors such as single-point of failure and data leakage. For more advanced tasks, the volume of training data can easily be in the petabyte range, as a result system engineers turn to distributed systems to enhance parallelisation [70].

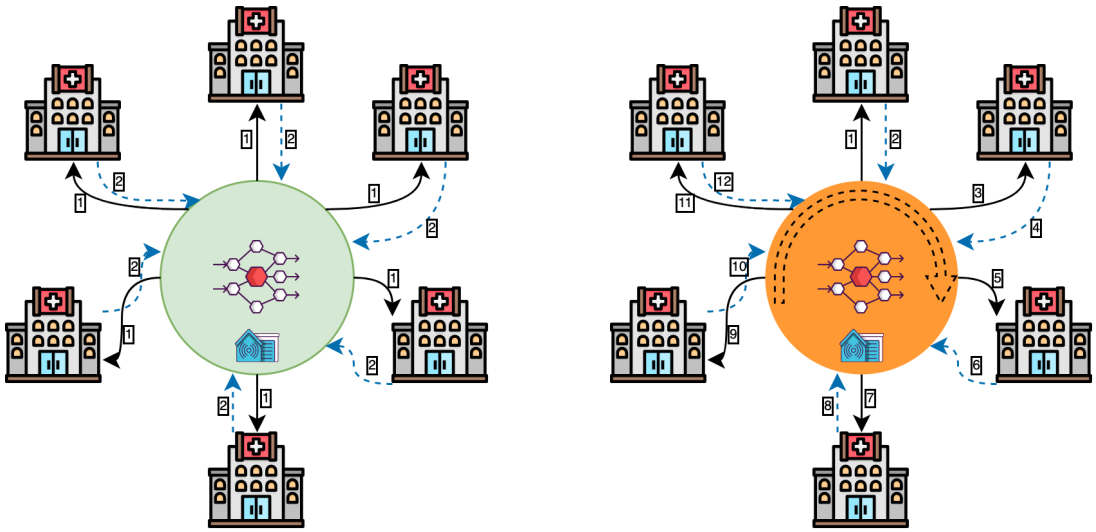
To overcome the challenges of centralising of large-scale datasets, we can take advantage of distributed architectures. Rather than gathering data from various sources (nodes), we keep the data at source and create a communication protocol (edges) to compose a decentralised and collaborative training process.

In a fully decentralised architecture, there will be no central server or a global unified model. An initial model is available at the beginning of the training phase and it will be shared between the participant nodes in the networks. Each node will be connected to minimum number of neighbour nodes and as a result they only communicate their learned parameters or update the shared model as depicted in figure 1b,c.

In this architecture a **round of communication** is refereed to each available node participating in updating the shared model with their corresponding neighbors in the decentralised graph.

One of the most common collaborative learning paradigms in distributed machine learning is to average local gradients steps with corresponding active neighbors in the training pool.

The development of completely decentralised SGD [3, 66] optimisation algorithms has allowed the introduction of a fully decentralised deep learning network through approaches such as Gossip Learning [4]. However, network



(a) (b)

Fig. 2: (a) Synchronous communication where each gradient update is sent back to central server. (b) Asynchronous communication where clients perform multiple gradient steps and share

convergence constraints, poor accuracy and communication limitation have ruled Gossip learning technique as a prominent solution to distributed machine learning [37].

Although decentralised systems follow a basic set of secure networking protocols, participants in the training process can reconstruct the original training data of their neighbour nodes. By using invert gradient and previous state of the model, a malicious attacker can reconstruct the training data and diverge the training process by adding adversarial data points [22].

A fully decentralised training process faces security, performance and model convergence bottlenecks which as of today rules it out as a prominent solution to non-data-centric problem [31]. A straightforward solution is to design and introduce a trusty central node to monitor and take authority between clients and preventing attackers from exposing data points or interfering with the network. In this approach, a global model is shared and controlled by the central node, which will also be responsible for the algorithmic optimisation process on the global model.

In federated learning, a central server orchestrates the training process and receives the contributions of all clients. The server is thus a central player which also potentially represents a single point of failure. While large companies or organisations can play this role in some application scenarios, a reliable and powerful central server may not always be available or desirable in more collaborative learning scenarios. Furthermore, the server may even become a bottleneck when the number of clients is very large, as demonstrated by Lian et al.

Companies or governmental authorities can act as the central unit in the provided solution and uphold the training process. This is the beginning of Federated Learning methodology and collaborative training process. The federated server takes control of the learnt parameters flowing from each client. It is worth noting that the federated server can be a bottleneck in distributed training [55]. We will further explore the challenges of federated learning in section 2.2.

2.2 An introduction to vanilla FL

As a solution to distributed and decentralised machine learning, federated learning trains a model by establishing a communicating layer to share a global model between a set of participants without exposing the underlying training dataset. In a federated setting, the architecture of a learnable objective is defined and initialised at the central server. After receiving a copy of the global model, each participant proceeds in a training phase using their local data. The result of each local computation is later shared with the central server and is ultimately combined using a predefined aggregation technique. The training process takes place locally at each client, and as a result, data remains intact and unexposed. Based on the communication and availability of clients, the aggregation process can happen asynchronously or synchronously where the central server updates the global model iteratively or halts to receive all the local updates, respectively.

Figure 2 shows the two type of communication between the federated server and client nodes in cross-silo setting.
Training a neural network [47] is one of the most commonly used techniques in the federated setting. Federated learning was initially introduced by Google [29] in 2016 as a solution to personalised Smart Keyboard. A poll of available devices participate by jointly training a local gradient step and sharing their weights with the central server. Using Federated Averaging (FedAVG) method [48], the central server consequently updates the global model and deploys it to participant devices. The central server act as a trusted entity and a proxy to communicate models between clients.
Training a model across a large set of clients where limited communication and network bandwidth are major bottlenecks is referred as *cross-device* federated learning. Internet of things (IoT), mobile phones, autonomous vehicles are few examples of such settings.
On the other hand, *cross-silo* federated learning takes place with fewer clients and a reliable communication layer. Banks, hospitals and private companies can collaboratively train a shared model to improve user experience, better fraud detection and track patients lifestyle activities.
Throughout the whole thesis, we assume federated learning in *cross-silo* setting unless stated otherwise. This section will take a deeper look into *Federated-Averaging* and *Federated-SGD*, two of the more conventional aggregation methods in FL.

2.3 Federated-SGD

Stochastic Gradient Descent (*SGD*) is one of the most commonly used methods in neural network optimisation. Compared to *Gradient Descent*, *SGD* is significantly more efficient as it performs gradient update on a subsample of training points. By randomly sampling a subset of the training data x_k, y_k we can define the weight update (w_{t+1}) methods as:

$$w_{t+1} \leftarrow w_t - \eta \nabla f\left(w_t; x_k, y_k\right) \tag{1}$$

Where ∇f is the gradient value with respect to x_k and y_k and η the learning rate [43].
For a training dataset containing n samples $\left(x_i, y_i\right), 1 \leq i \leq n$, the training objective is:

$$\min_{w \in \mathbb{R}^d} f(w) \quad \text{where} \quad f(w) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w) \tag{2}$$

$$f_i(w) = l\left(x_i, y_i, w\right) \text{ is the loss of the prediction on example } \left(x_i, y_i\right)$$

By extending on the equations above we can define federated learning stochastic gradient descent optimisation problem as n training samples that are distributed to K clients, where P_k is the set of indices of data points on client k and $n_k = |P_k|$ As an objective function we need to minimise f:

$$\min f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad F_k(w) \stackrel{\text{def}}{=} \frac{1}{n_k} \sum_{i \in P_k} f_i(w) \text{ [43]} \tag{3}$$

As a naive base-line, in federated SGD, the central server synchronously selects a client and performs a local update on the transferred model weights. The model is then sent back to the server and is ready for the next round of weight update using equation 2.
To clearly formulate federated SGD implementation, suppose in round t the central server broadcasts current model weights w_t to each client; each client K computes gradient: $g_k = \nabla F_k\left(w_t\right)$, on its local data. Each client k submits g_k; the central server aggregates the gradients to generate a new model [37]:

$$w_{t+1} \leftarrow w_t - \eta \nabla f\left(w_t\right) = w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k \tag{4}$$

As Algorithms 1and 2 shows, each client update the model weight only one time (1 epoch) in a synchronous setting which can be very costly in terms of communication bandwidth as well as excessive number of local iterations. It can be imagined each client is a selected batch in a training round. Several experiments have shown ¡Batch normalisation: Accelerating deep network training by reducing internal covariate shift, Communication-Efficient Learning of Deep Networks from Decentralised Data¿ the low convergence and unstable training of FedSGD as a result of non-IID assumption of distributed data on each client.

Algorithm 1 Federated SGD

initialize x_0
for each Communication round $t = 1, 2, ...$ **do**
    $S_t \leftarrow$ random set of M clients
    for each $k \in S_t$ in parallel do **do**
        $w_{t+1}^i \leftarrow$ ClientUpdate (i, w_t)
    end for
    $w_{t+1} \leftarrow \sum_{k=1}^{M} \frac{1}{M} w_{t+1}^i$
end for

Algorithm 2 Client Federated SGD

ClientUpdate(k, w): // Run on client k
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for local step $j = 1, \ldots, K$ do **do**
    $x \leftarrow x - \eta \nabla f(x; z)$ for $z \sim \mathcal{P}_i$
end for
return x to server

2.4 Federated Averaging

To reduce the number of communication and accelerate the training loop, we can perform multiple local updates on each client asynchronously and averaging the weights on the central server. In comparison to FedSGD, where we only transmit the gradients to the central server, in FedAVG, we send over the updated weights after E number of local updates [44].
Gradient descent guarantees convergence, however in a non-convex setting averaging weights of two models trained on non-overlapping IID sample sets can result in model divergence [27]. Moreover, Yann N et al. demonstrated that in high dimensional spaces, the non-convex error surface does not suffer from local minima with high error rate ,relaxing the saddle point problem for non-convex high-dimensional optimisation [16]. Accordingly, optimising two neural networks starting with the same initial weights can be averaged and result in higher accuracy. As Algorithms 3 and 4 shows, the FedAVG depends K clients indexed by k with E - number of local epochs, C - Number of client in each communication round and B - the local batch size.

Algorithm 3 Federated Averaging

initialize w_0
for each round $t = 1, 2, ...$ **do**
    $S_t \leftarrow$ random set of m clients
    for each $k \in S_t$ in parallel do **do**
        $w_{t+1}^k \leftarrow$ ClientUpdate (k, w_t)
    end for
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
end for

Algorithm 4 Client update for federated averaging optimisation.

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

3 Literature Study

This section will study different types of federated learning, data distribution challenges and explore several federated techniques to improve model accuracy. In the last subsection we will also look at security aspects of federated learning and study their effect on the training process.

3.1 IID and non-IID assumption

As per initial definition of federated learning by McMahan, "An unbalanced and non-IID (identically and independently distributed) data partitioning across a massive number of unreliable devices with limited communication bandwidth was introduced as the defining set of challenges." [48]
When optimizing per-client functions, one of the most significant distinctions is between IID and non-IID assumption. IID assumption for a client imply that each mini-batch of data used to update a client's local state is statistically equivalent to a randomly selected sample from the whole training dataset. As data collection for each client is different, the IID assumption is strongly violated. Although by apply this assumption to federated optimisation algorithms, it significantly simplifies and provides a framework for analysing the effect of non-IID data distribution. Therefore IID assumption institutes a stable baseline for federated optimisation [37].
Several optimization methods have been introduced to tackle the non-IID phenomenon in federated learning[45, 80, 82]. This non-indenticalness and dependence can be cause of each client collecting a particular set of sample at different time windows or locations. Non-IID can also take place between client selection as not always all participants in a training pool are available[71]. However throughout this thesis we only consider non-IID among clients data and assume clients preserve time ordered datasets such as consecutive frames in a video. Designing solutions for federated learning is highly influenced by how the data is shaped and distributed among clients. Clients can share features, samples or have no overlapping points. Based on the data distribution several optimisation techniques like FedProx [44] and Split Learning [62] have been introduced.

3.2 Horizontal federated learning

Categorising data distribution between clients help us implement robust optimisation techniques and improved aggregation methods. Collaborative learning between clients requires clarity on data-type stored on each client local dataset. This information must be available prior to implementation. This information can easily be shared through secure computation or Practical Private Set Intersection Protocols(PSI) [18] without exposing any sensitive data.
Horizontal federated learning [75] is referred to scenarios where all clients share the same feature space (columns) however clients have different set of samples (rows).
For example, two regional banks may have very different user groups from their respective regions, and the intersection set of their users is very small. However, their business is very similar, so the feature spaces are the same. For instance, two hospitals in the same region can have many different users but collect a similar set of features like body temperature, blood type and weight. Using horizontal federated learning a shared model can be trained on the overlapping features on discrete samples.
Figure 3 shows an overview of samples sharing the same feature set. Federated learning in horizontal setting is the most common method and we can use algorithm 3 to train a shared model.

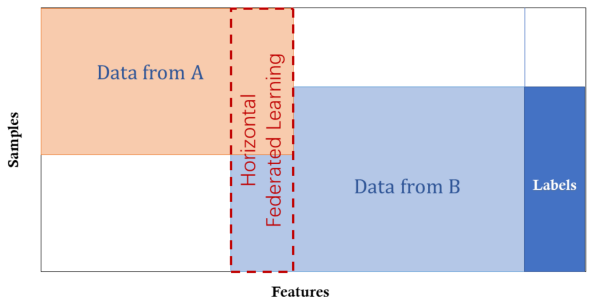


Fig. 3: Horizontal federated learning where clients share feature space with different data entry. [75]

3.3 Vertical federated learning

Clients do not always collect the same set of features however those features can belong to the same entry points. In such cases we introduce vertical federated learning [75] where clients collect non-overlapping set of features for the same sample. By using methods like Private Set Intersection [18] we can identify similar entry points using their unique identifier.
For example hospital that offer specialisation on cancer treatments can collaboratively learn from the data of local clinics or other hospital for the exact same set of patients. Vertical federated learning is one of the most prominent and promising methods for medical research [56].
Compared to horizontal federated learning, vertical federated learning is much more compute intensive and following vanilla implementations of federated learning can be insufficient. Solutions like split learning which we will look at in section 3.6 provided a robust implementation for such cases.
As we observed previously data collection and data distribution plays a crucial in definition of federated problem statement. Datasets that do not share features or sample identifiers (neither horizontal or vertical) can also collaborate. Figure 4 depicts high-level view of vertical federated learning.

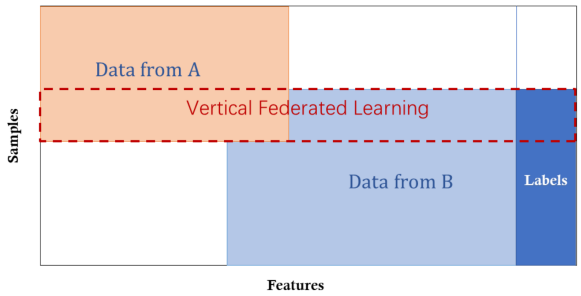


Fig. 4: Vertical federated learning where clients hold different features of the same users. [75]

3.4 Federated Transfer Learning

In the real-world cases, organisations and institutes not always follow a unique set of practices for data collection. This results into a non-overlapping of data points either in sample IDs or feature columns. For these cases we can use federated transfer learning [75]. Although clients have no mutual points, but the domain knowledge is transferable [65]. For examples a hospital that use MRI scans for brain tumor and an clinic that uses MRI scans for skull damages can collaboratively share a transfer model. Predictions for samples with just one-sided features are generated by learning a common representation across the two feature spaces. As a significant improvement over previous federated learning systems, federated transfer learning addresses different challenges compared to vertical or horizontal federated learning.

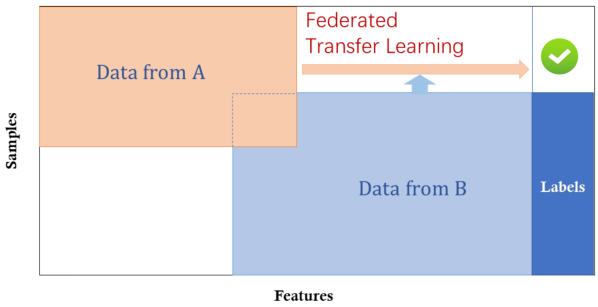


Fig. 5: Federated transfer learning where clients have no overlap of data but the domain knowledge is transferable.

3.5 Privacy - Methods, Aspects and Risks

Distributed training does not completely resolve data regulation by itself. Several methods like inverse gradient [22] can be used to reconstruct the original data. Malicious attackers can simply use this technique to access participants' training data. Although we presume a set of trustees to collaborate in the cross-silo scenario, the issue changes to privacy-preserving training data. For example, General Data Protection Regulation (GDPR) does not allow sensitive health records to be shared even between hospitals. It is often challenging to measure privacy or fully guarantee a distributed training process. Secure computation, homomorphic encryption and differential privacy are among the most conventional methods to preserve data privacy in the federated setting. We will provide a high-level overview of these security measurements in this section. It is worth mentioning that not all security risks can be solved by secure federated learning techniques and requires systematic design either at the hardware layer or communication protocols which are beyond the scope of this thesis.
To better understand the risk aspects of federated learning, we will introduce potentially vulnerable components:

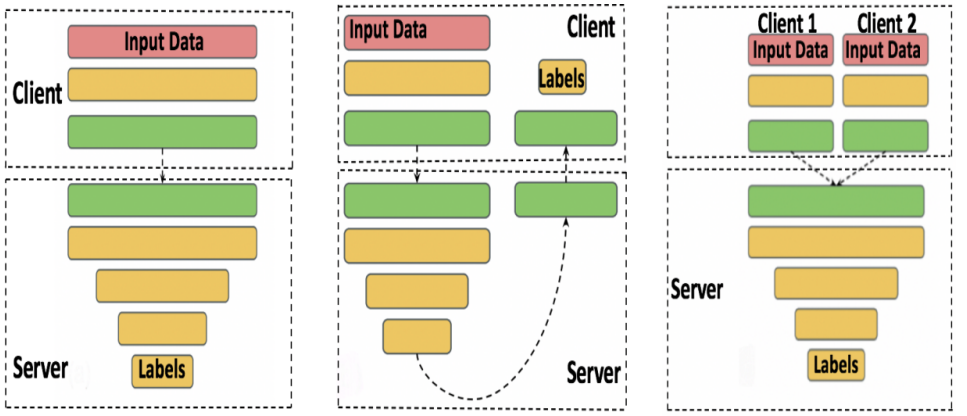
Clients: An entity with root access to the client device and its data. As mentioned before, in cross-silo we presume clients and the central server are in a trusted environment and communication are honest. However an honest-but-curios client can extract original training data of other clients or target labels of the federated server. Methods like invert gradient [22], adversarial weight update [81] and data masking [37] can reveal private information. It is challenging to solve such security threats from federated learning perspective as it requires deeply technical solutions like Trusted Execution Environments (TEE) [54].

Server An entity with root access to the central server and can read and process incoming model update requests.
The central server carries the most vulnerability as a malicious server can reconstruct all clients data, diverge model training and misusing the inference model. Although in cross-silo federated learning, central server is usually a trusted organization, however, they must be transparent about model objectives and the purpose of training. The central entity should not be able to store or use the weight updates as they reach the server. This can cause in misuse of training data and bring ambiguity for data ownership [49].
Model inference The entity responsible for exposing the training result as an executable application. An inference model can be deployed on millions of devices or limited institutes. Nonetheless, adversarial attacks, misuse and model exposure are among the most critical security risks in model inference [37].

To tackle vulnerability issues mentioned above for different components of federated settings, we can use methods like secure computation, homomorphic encryption and differential privacy. These methods affect all aspects of federated training and inference as they introduce additional computational encryption and masking methods over the data. Here we will explore an overview of mentioned techniques.

Secure Computation One of the main objectives and advantages of using federated is to bring security and privacy to data owners. Secure computation or Secure Multi-party Computation (MPC) [26] is the notion where multiple participants can jointly perform arithmetic operation without exposing the original data. Multi-party computation uses Shamir's secret sharing technique [17] where non of the participants of the computational graph can be identified. By dividing one secret value across participants of the computation scheme in such a way that neither of the parties can learn about the original data. To better understand multi-party computation, we can introduce the problem where n parties are sharing their private data point to participate in a computational method. To secure this computation it can only take place if and only if all parties data are available and none can have access to the other data point. To do such encrypted computation we use the Lagrange theorem where we define a polynomial of degree n (number of shared data points). We want to find an answer to how can we define a polynomial function $h(X)$ where it crosses all the data points available in multi-party computation?
By definition we need at least $n+1$ points to create a polynomial (f) of degree n which passes over all $n+1$ points. As initial assumption we define $f(0)$ as the secret value. Using Lagrange interpolation and having n other points on this polynomial we can find $f(0)$. $C \subset \mathbb{F}$ s.t. $|C| = n+1$ where $h(i)$ is value of the polynomial at data point i. Then it holds [37]:

$$h(X) = \sum_{i \in C} h(i)\delta_i(X) \tag{5}$$



(a) Simple vanilla split learning (b) Split learning without label sharing (c) Split learning for vertically partitioned data

Fig. 6: Different types of communication between client/s and server[69]

where $\delta_i(X)$ the degree n polynomial s.t.: $\delta_i(j) = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases}$ In other words:

$$\delta_i(X) = \prod_{j \in C, j \neq i} \frac{X - j}{i - j} \tag{6}$$

Homomorphic Encryption Although multi-party computation encryption method brings a layer of security, it requires all parties to participate and be available at the same time. This limitation also increases computation and communication across all parties. To overcome such issues, federated learning can be secured using the homomorphic encryption method (HM) [24].
Enables a party to compute functions of data to which they do not have direct access, by allowing mathematical operations to be performed on *ciphertexts* without decrypting them.

Differential Privacy Both homomorphic encryption and secure computation add a substantial overhead to computation required for the training process. In case specific situations we can achieve privacy by only masking the data and adding random noise from a tractable distribution. Formally we can define it as the amount of information that could be transferred from an individual data point where the training objective does not expose or get access to sensitive information. Algorithms with differential privacy necessarily incorporate some amount of randomness or noise, which can be tuned to mask the influence of the user on the output [19].

3.6 Split Learning

As mentioned in section 3.3 vanilla federated learning is not suitable for vertical federated learning. To further extend the collaborative capability of federated learning, several complimentary distributed learning paradigms have been introduced. Split learning is a distributed learning method which splits the execution logic of a model based on transferring the input data to an embedding layer known as the cut-layer. Each client trains their local model up to the cut-layer and share the results with each other or the central server. This process continues until all clients participate in training up to the cut-layer and passing it forward which results in one complete round of forward propagation. In general there are three types of Split learning. Vanilla split learning which is used for distributed training, U-shaped split learning which creates a trusted party to act as an accumulator of weights without having the target labels for one single client and Vertical split learning which multiple clients participate to train a shared global model. In this section we will take a look at each split learning and explore use cases of each implementation. Further more [62] compared split learning and federated learning from communication and security perspective.
One of the main drawback of split learning, is the ability to reconstruct the original data. Nonetheless, depending on the use case, the data exposure can be tolerated and there several methods which can reduce such data exposures while sacrificing the performance. By decreasing the distance correlation between activations and the raw data, a variant of split learning known as NoPeek SplitNN [68] is able to minimise the possible leakage through communicated activations while retaining excellent model performance via categorical cross-entropy.

Vanilla split learning The most straightforward implementation of split learning where each client completes a forward pass up to the cut-layer and share its output with the federated server. At the central server the gradient values are calculated using the backpropagation method up to the cut-layer. These gradient values are again shared with the clients to update their weights accordingly. This process continues until network convergence. Figure 6a shows a high-level view of vanilla split learning. In this scenario the client does not know the target label on the central server and depending on the cut-layer the central can no originally reconstruct the initial input to the network.

U-Shape split learning Training on sensitive data is not always straightforward. Clients do not always share labels with the central server. To solve such cases, we can use U-shape split learning, where the final layer of the central server is sent to the client to calculate loss and gradient values for backpropagation [69]. This data is again shared with the central server to update weights up to the cut-layer and client weights respectively. Figure 6b shows an example of a network configuration for U-shape split learning.

Vertical split learning As mentioned in section 3.3, vertically distributed data where client gather different features for the same entry points provide great value for for health-care. However vanilla federated learning struggles with convergence, accuracy and efficient communication for vertically distributed data [11, 46]. As a solution to vertically partitioned data, vertical split learning can jointly train a shared model between a set of clients even if clients data differ entirely. Clients asynchronously train their specific model at each step up to an embedding layer. This embedding layer is later shared with the central server, where using stack or average methods is fed for a forward pass of the federated model. As the federated server holds the labels, it calculates gradients and backpropagates weight updates for each client. This process continues until model convergence. To test the accuracy of a vertically distributed network, we need to break down input's features and send them for a forward pass at each client up to the embedding layer. Again this embedding is shared with the central server to make a prediction. Figure 6c shows a high-level view of vertical federated learning.

Stack & Averaging the cut-layer In vertical split learning, there we can take two main approaches in dealing with the incoming weights from each client. Similar to federated averaging (FedAVG), we can average the weights and pass them through the central server or stack on top of each other and train the global model.

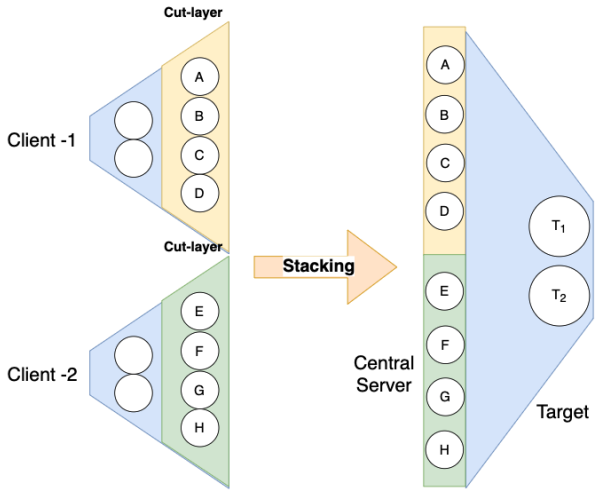


Fig. 7: Stacking incoming weights in the central server.

3.7 Federated aggregation algorithms

As we divided federated learning based on data distribution amongst clients, we can take a look at different aggregation algorithms that focus on improving model accuracy and increase communication efficiency. These solutions appear both for client local training as well as the central server. Table 1 puts together a list of different

aggregation algorithms and their objective target. These algorithms can be divide in two main category based on the aggregation paradigm of the central server. Algorithms that aggregate based on parameters or derived parameters of client models, indicated by (P) in table 1 and algorithms that aggregate based on client's model, indicated by (M).
Algorthms such as FedSGD, PFNM, RF, ILTM, MAP and DT where initially design for optimisation of the trainable objective function such as neural networks, random forest, maximum-a-posteriori, decision trees and probabilistic federated neural matching respectively.
Secondary batch of algorithms focus on improvements and optimisation of the base aggregation algorithms. Methods like FOCUS, FedMA, FedProx, FUALA, PMA, OFMTL, SCAFFOLD target data quality and efficiency of clients, more effective matching average, weighted contribution of clients, remove noise and bias and control clients model-drift respectively.
Algorithms like SMA improve both the convergence and model quality by leveraging split learning [69] techniques. NDW and ASTW are methods build on top of blockchain which calculate the contribution of each client and optimises based on improving the global model accuracy.

Aggregation algorithm	**FL Setting**	**Design**	Type	Considerations of clients	Article
FedSGD	H	Basic aggregation algorithm	P	Gradient, Quantity of data	[48], [40], [9]
RF	H		M	Model	[41]
ILTM			P	Model similarity, Training parameters	[36]
MAP			P	Prior distribution	[53]
DT			P	Parameters of decision tree	[15]
PFNM			M	Model	[77]
QNA	V		P	Curvature	[74]
FOCUS	H	Improve model quality	M	Model, Quality of model	[14]
FedMA			P	Weight of each network layer	[72]
FedProx			M	Model, Quantity of data	[44]
FUALA			M	Model, Quality of model	[5]
PMA			M	Quantity of data, Diversity of label, Differences of model	[2]
OFMTL			M	Model, Quality of model	[42]
SCAFFOLD			M	Model	[38]
FedAvg	H, T	Converge faster	M	Model, Quantity of data	[45]
SMA	H	Converge faster and improve model quality	M	Data quality, Computing power, type	[76]
NDW			M	Model, Quantity of data and model, Frequency of participation	[39]
ASTW			M	Model, Quantity of data, Sequence of recently updated models	[12]

Table 1: An overview of different aggregation algorithms.

One of the advantages of federated learning is its ability to combine recent machine learning developments in a distributed and secure way: Transfer-learning [65], meta-learning [10], multi-task learning [63] and multi-view learning [34] to name a few.

3.8 Federated fusion

Novel approaches in machine learning can be adapted for the federated setting where clients can train a model collaboratively on distributed silos of data. Fundamentally federated learning requires combining learnt components gathered from a pool of clients and training a centralised or personalised model. Learning paradigms that can satisfy federated base objectives can be used for collaborative training. Here we will explain two of the most commonly used methods for multi-objective machine learning and client personalisation. We assume these methods are used amongst trusted parties and do not require an encryption layer. To further study the encryption methods, please check the original papers.

	Models	FedML	FATE	PySyft	TFF	PaddleFL	LEAF	Nvidia Clara	IBM Federated	Flower
Open-source	-	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes
Number of Stars	-	835	3500	7600	-	340	460	-	-	550
Number of Contributors	-	29	55	372	-	15	5	-	-	22
ML Engine	-	PyTorch	TensorFlow/ Pytorch	PyTorch	Tensorflow	Paddle	Tensorflow	PyTorch	Tensorflow	Agnostic
Supported Models	RF/ DT	No	No	Yes	No	No	No	No	No	Yes
	NN/ CNN	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	LM	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes
Privacy	HE	Yes	No	Yes	Yes	Yes	No	No	Yes	Yes
	DP	No	Yes	Yes	No	No	No	No	Yes	Yes
	CM	No	No	Yes	No	No	Yes	Yes	Yes	Yes
SL Support	-	Yes	No	Yes	No	No	No	No	No	No

Table 2: Federated learning framework comparison. Frameworks with higher number of stars and contributors are more reliable and guarantee long time support.

Multi-Task Learning Multi-task learning can enable machine learning models to learn multiple objectives at the same time. Using multi-task learning we can use a single model to solve multiple task which as a result increases efficiency and better generalisation [79]. By sharing the backbone of the model and using multiple heads, a multi-task architecture can optimise for several objective hence better generalisation.
Leveraging such architectures in federated learning can increase the productivity of clients. For example, hospitals can train a model to extract knowledge from brain MRIs and adapt it for tumour detection, disease prevention, or brain damage from a single multi-task model. Multi-task learning can be used only for horizontally distributed data in a cross-silo setting. Although cross-devices can also use this technique, the multi-task architecture assumes all tasks are always available in the training process, which cross-device settings can violate. In addition, clients require high computation and storage to train for multiple tasks, making it suitable for cross-silo federated learning.

Federated Meta-learning In the vanilla federated learning setting, the global model benefits from clients participation and each positive training contribution of clients can ultimately result in improvement of the global model. In cross-silo federated learning settings, clients can also benefit from the global model for internal usage. For example, several hospitals can jointly train a general tumour detection deep learning model, and specialised practitioners can use it internally for regional tumour detection, like brain or lounge. Although the shared model obtained a global higher accuracy on a wide range of data points across all clients, it does not necessarily entail a better accuracy on an individual client's dataset. This phenomenon exacerbates as heterogeneity increases between datasets [10].
To address this problem, we can use a Meta-Learning based formulation for training the global model. In such a case, the global model will be trained so each client can adapt to their local dataset with few gradient steps. Meta-learning models in federated learning can introduce personalisation and domain adaptation across a broader range of use cases. Fallah et al. used Personal FedAvg (Per-FedAvg) [10] algorithm to demonstrate Model agnostic meta-learning [21] implementation for distributed training.

3.9 Federated Learning Frameworks

This section will take a look at some of the most famous federated learning frameworks and libraries. Researchers and data scientists can use these tools to explore and simulate federated learning in a wide range of scenarios. Frameworks like Tensorflow Federated (TFF) [1] can be used as a production-grade toolset to manage and implement federated learning at a large scale. However, most of the frameworks introduced in this section are under active development and are often not recommended for large scale platforms. Recent

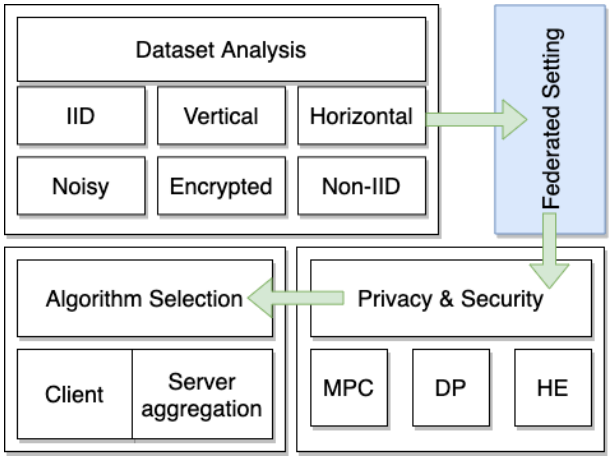


Fig. 8: Macro view of a federated learning eco-system.

support of Cloud Providers like Google and Microsoft for federated learning has increased the development pace of such frameworks and libraries. Federated learning libraries or frameworks follow a systematic set of objectives and requirements. As Figure 8 shows, by determining dataset distribution type and the IID assumption, a federated learning framework can structure client communication protocols and training/inference of individual clients. Characteristics of each client's data will be processed under the federated setting, where the central server evaluates the training contribution of each client. As privacy and security play a crucial role in a federated learning framework, it must be optimised and designed based on training objectives and use-cases. The aggregation method on the server-side, along with encrypted computation, completes a federated learning framework.

In table 2 we compare nine federated learning frameworks based on open-source availability, number of stars on Github.com, number of active contributors from the open-source community, the core machine learning engine, supported learning paradigms, privacy aspects and whether they support split learning. Framework with more stars and contributors are often more reliable and can be trusted for long-term support. Here we will explore three of the most prominent frameworks based on the number of stars on Github.

By taking advantage of frameworks in table 2 medical researchers are able to solve problems like Mortality prediction [33], Hospitalization prediction [6], Preterm-birth prediction [5], Activity recognition [13]. Jie Xu et al. [73] takes an in-depth look at medical problem solving in federated learning context.

FedML [30] is an open-source library that focuses on benchmarking and evaluation of different federated learning methods. FedML provides structural API for cross-silo distributed training, cross-device distribution training and standalone simulations. The modular building block of this federated learning benchmarking library enables researchers and developers to compare and explore new methods by simply extending basic training and connectivity classes. FedML follows a worker/client architecture to simulate a wide range of diverse network topologies. FedML-Core is a set of low-level API where the model architecture is initialised on top of PyTorch. The optimiser class can be extended easily to support custom loss-function objectives for the global network. The Core layer also manages the communication and security layer between clients. The communication can be initialised on top of homomorphic encryption [24] or multi-party secure computation [26]. Each message is later on dispatched to a message-bus service like RabbitMQ. By extending the based algorithm classes, users can implement custom aggregation and local training methods using the FedML high-level API. The data distribution and model architecture can easily be controlled from the high-level API.

FATE (Federated AI Technology Enabler) [51] is an open-source platform which provides a secure computing framework to support the federated AI ecosystem. It implements secure computation protocols based on homomorphic encryption [24] and multi-party computation [26]. FATE supports federated learning architectures and secure computation of various machine learning algorithms, including Logistic regression, Tree-based algorithms, deep learning and transfer learning. FATE platform can be deployed on top of container orchestration services like Kubernetes, which provides elasticity and scalability for production-grade deployment. FATE pro-

vides both an inference system and an end-to-end training pipeline. Several monitoring tools and its high-level API makes FATE a reliable platform for production-grade federated learning.

PySyft [59] With a focus on data ownership and user data privacy, PySyft provides an end-to-end framework for federated learning training and inference. By providing client-side application service, the central server can only read data from clients who agreed to participate in a specific objective in each round of training. The computation is performed on top of the multi-party computation [26] and homomorphic encryption [24] security layers. PySyft also provides a robust development process by keeping the data local to each client but assigning a remote tensor value to each data point. Using homomorphic encryption, PySyft library can perform arithmetic operation on single entries. This functionality can help researchers to examine sample points from clients datasets without exposing private information. PySyft is under active development and is not recommended for production-grade deployment.

4 Benchmarking and fANOVA

To evaluate hyperparameters importance of of federated learning we designed a benchmarking application on top of PyTorch and PySyft, leveraging secure computation with GPU parallalisation support. We further used functional ANOVA, a hyperparameter importance framework, to calculate the effect of individual hyperparameters. In this section we breakdown the design and implementation method of our benchmarking application and study how functional ANOVA can calculate hyperparameter importance using a tree-based algorithm. Our benchmarking implementation is open-source and available online[1].

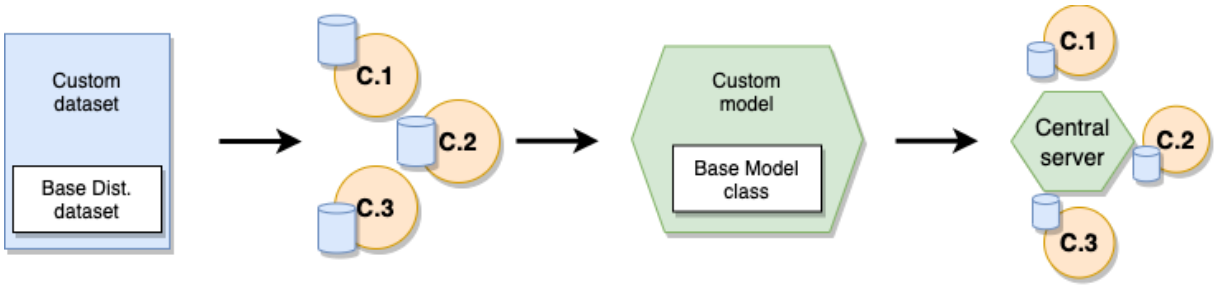


Fig. 9: High-level overview of our federated benchmakring application.

4.1 Benchmarking Application Design

In section 3.9 several frameworks and libraries were introduced which at time of writing this thesis, none of them provided a comprehensive analysis and benchmarking setup for vertical federated learning using Split learning. Our benchmarking application was designed with three main factors in mind.

1. Dynamic and adaptive to a wide range of datasets and data types.
2. Modular training and test methods
3. Parallelisation and GPU support

To support a wide range of data types, we create a custom dataset class which can easily be extended and only requires to specify how to read features and target labels. Using an API based approach, we are able to support image, text and tabular datasets. We used the same approach in designing the training process. Clients can each have a unique network design or process on a shared model design. This flexibility enables combining multiple data sources such as image and tabular data [25]. Our experiments are mainly on tabular data using the OpenML CC18 benchmarking dataset [67]. To automate the data loading process we used OpenML python library to read, pre-process and clean the data [20]. Figure 9 shows a high-level overview of the benchmarking framework, where by extending the base distributed data classes we can send the data to virtual clients and construct a federated learning enabled machine learning model. This model is then deployed on the central server and is orchestrated for training.
One of the main challenges in the federated setting is the distribution of data amongst the participating clients in the training process. One client can hold the most important features and as a result other clients affect on the global model would be insignificant. To simulate such cases and overcome bias challenges, we designed a data distribution algorithm using Dirichlet distribution [52]. Dirichlet is the generalised version of Beta distribution.

$$x_1, \ldots, x_K \text{ where } x_i \in (0,1) \text{ and } \sum_{i=1}^{K} x_i = 1 \tag{7}$$

$$\mathrm{Dir}(\theta \mid \alpha) = \frac{1}{\mathrm{Beta}(\alpha)} \prod_{i=1}^{K} \theta_i^{\alpha_i - 1}, \text{ where Beta } (\alpha) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)} \text{ and } \alpha = (\alpha_1, \ldots, \alpha_k) \tag{8}$$

[1] https://github.com/maminio/federated-benchmarking

Imagine there are four courses that students can select from. In the beginning, the first student will randomly pick a course. For the second student, it is more likely to choose the previously selected course by the first student as it has at least one student. This process continues until all students pick a course. The probability of students choosing the most dominating course is parameterised with the alpha value of the Dirichlet distribution.

4.2 functional ANOVA

Functional analysis of variance is a framework which measures the contribution of hyperparameters on the performance of functions. It provides insight on how hyperparameters effect each other and the overall performance of the network.
By using tree-based surrogate models, functional ANOVA is able to calculate how a hyperparameter performs, averaged over a wide range of continues or discrete variables.
To manually analyze the value of hyperparameters, algorithm designers generally look at the local surroundings of a particular hyperparameter configuration: they alter one hyperparameter at a time and see how performance changes. Please keep in mind that just one instance of the other hyperparameters can be used in this research to determine how different hyperparameter values perform. For algorithm designers, the best case scenario is to know how their hyperparameters effect overall performance, not simply in the setting of a single fixed instantiation of the remaining hyperparameters[35, 57].
Let A's (true) performance be $y : \Theta \mapsto \mathbb{R}, U \subseteq N$, and $T = N \backslash U$. A's marginal performance $a_U(\theta_U)$ is then defined as

$$\begin{aligned} a_U(\boldsymbol{\theta}_U) &= \mathbb{E}\left[y\left(\boldsymbol{\theta}_{N|U}\right) \mid \boldsymbol{\theta}_{N|U} \in X(\boldsymbol{\theta}_U)\right] \\ &= \frac{1}{\|\boldsymbol{\Theta}_T\|} \int y\left(\boldsymbol{\theta}_{N|U}\right) d\boldsymbol{\theta}_T. \end{aligned} \tag{9}$$

Similarly, A's marginal predicted performance $\hat{a}_U(\boldsymbol{\theta}_U)$ under a model $\hat{y} : \Theta \to \mathbf{R}$ is

$$\hat{a}_U(\boldsymbol{\theta}_U) = \frac{1}{\|\Theta_T\|} \int \hat{y}\left(\boldsymbol{\theta}_{N|U}\right) d\boldsymbol{\theta}_T \tag{10}$$

It is worth mentioning that if the predictive model $\hat{y}$ has low error on average across the configuration space, the difference between predicted and true marginal performance will also be low.

5 Experiments

Federated machine learning performance is affected by a series of training and external hyperparameters. To discover the effect of such hyperparameters, we demonstrate a series of benchmarking experiments and analyse the impact of each hyperparameters on model convergence and network accuracy.
As mentioned in sections 3.6 and 3.3, vertically distributed data are the most common cause for hospitals and medical institutes. RIVM (National Institute for Public Health and Environment) requested that we aimed our experiments towards vertically distributed tabular data across a wide range of clients (hospitals).
In this section, we initially explain how we constructed a variety of parameters and hyperparameters from tractable distributions and study the effect of each hyperparameter on the global network accuracy. All our experiments are compared to one another using functional NOVA framework to calculate the marginal contribution of each hyperparameter on the accuracy.
We got inspired by federated machine learning (FedML) [30] and PySyft [59] benchmarking libraries and used them to design our experiments.
Throughout our experiments, we faced with several challenges using the basic functionality of functional NOVA to extract the marginal effect of hyperparameters from one set of varying hyperparameters experiment as hyperparameters were highly dependent on each other and varying hyperparameters like cut-layer and number of linear layers would drastically increase bias in the evaluation of functional NOVA. Therefore, we designed each experiment with a limited number of varying hyperparameters to get the best result of functional NOVA framework.

5.1 Hyperparameters and sampling

Basic network hyperparameters like learning-rate (lr), weight-decay (wd), number of epochs, depth of the network are bound to the learning objective. On the other hand, the number of communication rounds, client drop off, distribution of data points among clients, cut layer, aggregation methods are parameters of the federated architecture.
Table 3 generally describes each hyperparameter and its distribution.

Param	**FL Type**	**Distribution**	**Range**
Epoch	H / V	Uniform	15-100
Partition Alpha (Dist. clients)	H / V	Dirichlet	0.1 - 20
Batch Size	H / V	Uniform	3-7
lr / wd	H / V	Log Uniform	0.0001 - 0.1
Number of Clients	H / V	Uniform	1-20
Cut Layer	V	Uniform	2-9
No. Linear layers	V	Uniform	3-10
Aggregation Type	V	Uniform	0,1
Random Seed	H / V	Uniform	0-100

Table 3: Hyperparameters and their respective distribution, used in the benchmarking experiments

5.2 Data distribution among clients

In a central setting, it can be inferred that there is only one client, and it has all the data points. By extending this notion to federated learning, we can define n clients that each hold a portion of the data. The way data points are distributed between clients has a direct impact on the training process.
For example in vanilla federated learning, if one client holds the majority of the training datapoints, its contribution to the global model is significantly larger compared to other participants in the network. However, as the number of clients increase, depending on the aggregation type of the central server, the contribution of data holders tends to a balanced state. Our initial experiments with data distribution demonstrated that the number of clients and their share of training data is one of the key factor of federated learning setting. We used the Dirichlet distribution $Dir(\alpha)$ which is the general form of the Beta distribution for continues multivariate probability distributions parameterised by α of positive real numbers. High value Partition-alpha(PA) indicates

Param	Distribution	Range
Cut Layer	Uniform	[2-9]
partition-alpha	Log Uniform	[0.1-20]
Batch Size	Const.	64
lr / wd	Const.	0.01
Number of Clients	Const.	5
Epoch	Const.	15
No. Linear layers	Const.	10
Aggregation Type	Const.	Stack
Random Seed	Const.	0

Table 4: Split learning experiment on the effect of cut-layer and varying PA.

the dataset is almost evenly distributed and a low partition-alpha means distribution of data is unbalanced.

5.3 Effect of Cut Layer on performance

In this experiment we look at the effect of cut-layer on the performance of the model. During the training phase each client trained a local model upto the cut-layer and the values of the cut-layers are shared with the central server. If a client has a very low cut-layer, it means the shared weights carry significant information from the input data. This can introduce several risk factors like reverse gradient to get hold of the input data. A lower cut-layer also means the computation-load is mostly on the central server and client do not need high capacity compute machine for training. In our experiment we sample cut-layer from a uniform distribution, partition-alpha from log-uniform and preserve the rest of the hyperparameters. The result in Figure 10 ran on 720 experiments over 10 datasets.

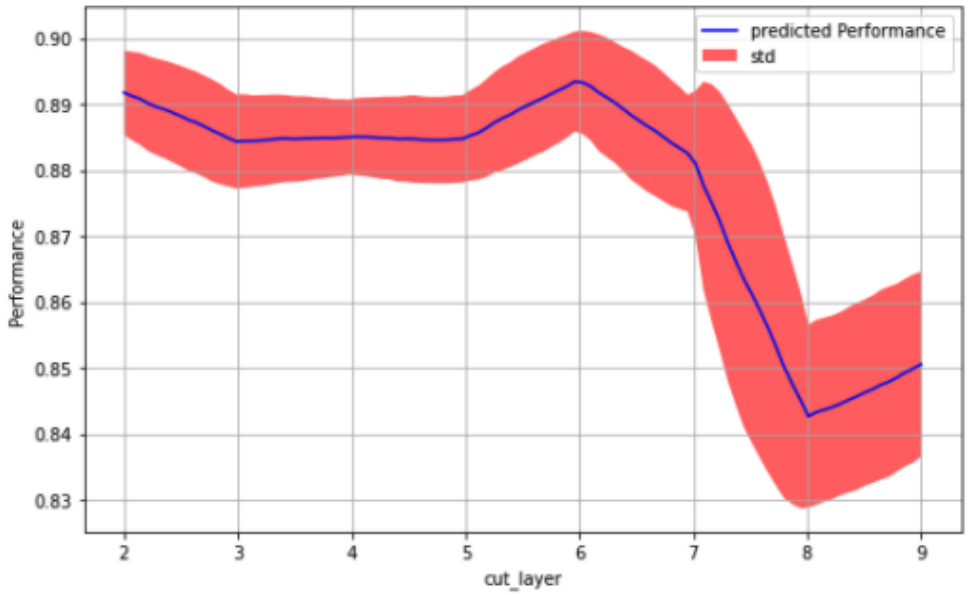


Fig. 10: The effect of cut-layer on performance. Lower cut-layers share closer representation of the input data with the central server.

Figure 10 shows lower cut-layers result in higher accuracy. Lower cut-layer means the stacked weights at the central server have a closer representation of the input data. In other terms, lower cut-layer increases the similarity of the federated learning setting to central learning. From the figure above we can also observe that the difference between cut-layers 2 to 6 is significantly smaller than 6 to 9. This means we can select an intermediary cut-layer which can provide high accuracy, higher security for weight sharing and distributed training(More gradient steps at the edge).

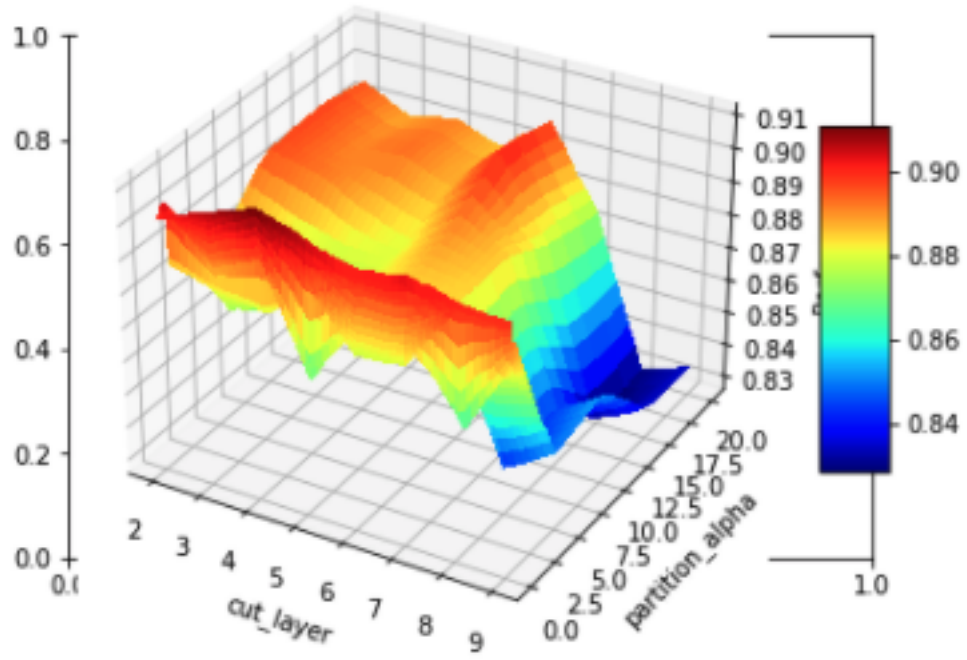


Fig. 11: Pair-wise marginal effect of cut-layer and partition-alpha. Lower cut-layers neutralise the effect of larger partition-alphas.

Following our experiment, we examined the pair-wise marginal effect of partition-alpha and cut-layer. This comparison can show us how much relevant information each client is sharing with the central server. Figure 11 shows lower partition-alpha (One client holding majority of features) has the highest accuracy. However the higher partition-alpha, we see a significant drop for higher cut-layers. Tishby et. al. shows higher layers in a neural network transfer high-level features [61] of the input data which in our experiment means the shared weights carry more information about each clients local features rather than information about the input data.

5.4 Effect of clients and data distribution

Param	Distribution	Range
Number of Clients	Uniform	[2-20]
partition-alpha	Log Uniform	[0.1-20]
Batch Size	Const.	64
lr / wd	Const.	0.01
Cut Layer	Const.	3
Epoch	Const.	15
No. Linear layers	Const.	10
Aggregation Type	Const.	Stack
Random Seed	Uniform	[0-4]

Table 5: Hyperparameter space to examine the effect of data distribution over participants in a federated learning network.

One of the most challenging problems in federated learning, is the convergence of non-iid [71] datasets and how clients contribute to the global model. In this experiment we take a look at how number of participants effect the convergence of the network. We further examine the network performance by varying data distribution between clients. This experiment ran on 1000 different settings sampled from uniform and log-uniform distributions(Check Table 5).

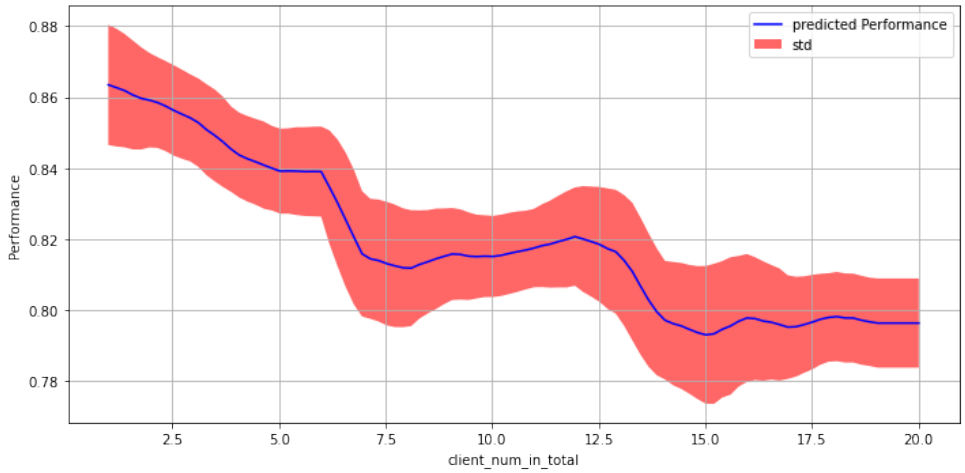


Fig. 12: Effect of number of clients on performance. As the number of clients increase the performance decreases.

Our experiments show the fewer the number clients, the higher accuracy we will get from the global model. Fewer number of clients means, each client hold larger portion of the features. If we only have one client it can be interpret as central learning and as we increase the number of clients we have drop in network performance. In this experiment the amount of data is fixed and we only demonstrate the variation in performance by changing the number of clients. Although in a real-world scenario, adding new clients means more data and can potentially increase network accuracy. It is worth mentioning adding new clients to the poll of participants not always result in higher accuracy. Model divergence is highly probable as data heterogeneity increases in non-iid data distribution [45, 80].

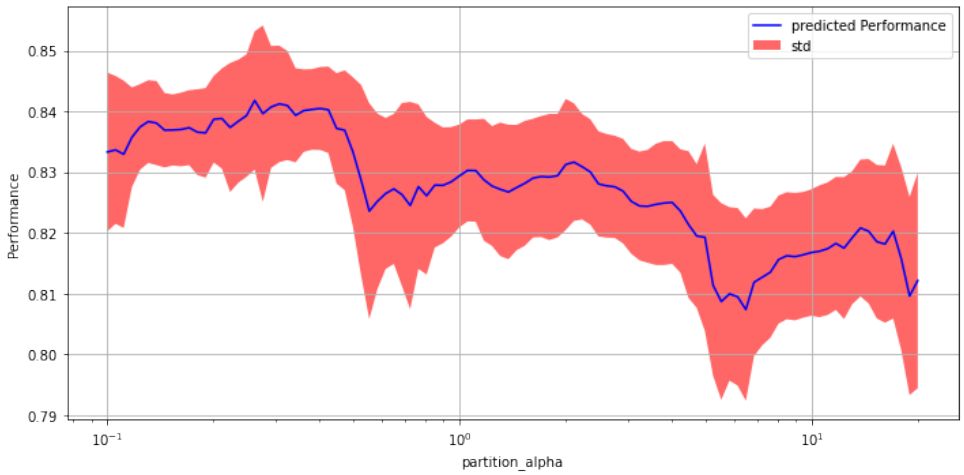


Fig. 13: Effect of partition-alpha on performance over 1000 runs.

We further look at the effect of partition-alpha on the performance. As Figure 13 shows our network has the best performance for lower values of partition-alpha. As mentioned in the previous experiment, lower values for partition-alpha means one client holds the majority of the features. Our results show, regardless of number of clients, if there exists one client with significantly larger portion of the features, we can expect higher performance of the global model.

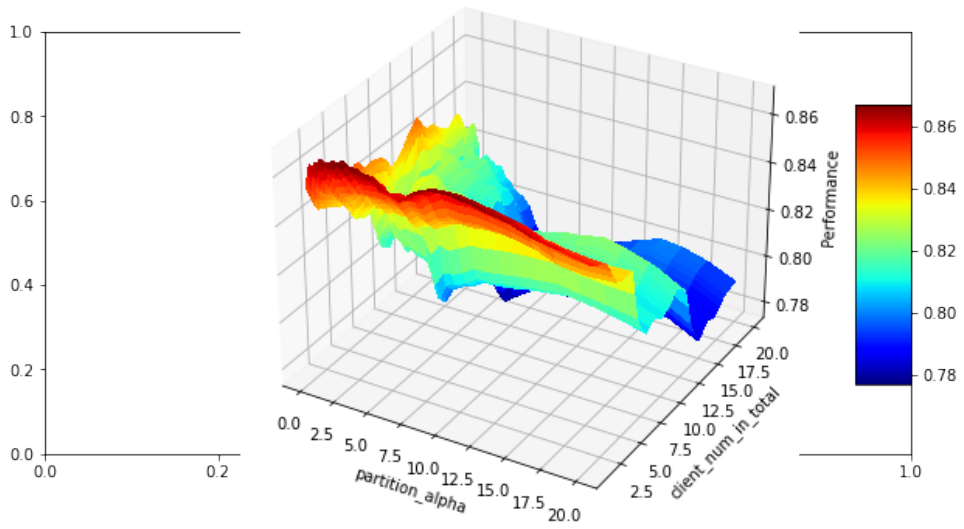


Fig. 14: Pair-wise marginal comparison between partition-alpha and total number of clients. Lower partition-alpha works almost similar to central training.

As we saw in Figure 13 higher value of partition-alpha result in lower accuracy. We can also demonstrate the same phenomenon in the marginal diagram of two hyperparameters of partition-alpha and number of clients. It is worth noting fewer clients and high partition-alpha does not severely affect performance. We can see a drop of accuracy as the number of clients increase.

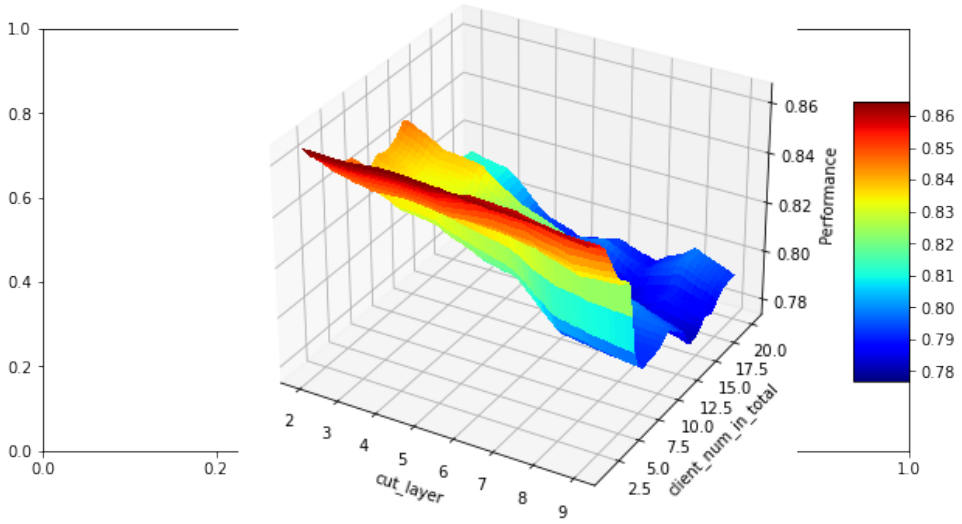


Fig. 15: Pair-wise marginal comparison between cut-layer and number of clients. Number of clients has a major effect on the performance, although low cut-layers can hold an average result.

As mentioned in experiment 5.3, lower cut-layer result in better performance of the network. In Figure 15 we can observer that the number of clients in comparison to cut-layer have a drastic effect on the performance of the network.

5.5 Effect of aggregation methods in Split Learning

Param	Distribution	Range
Number of Clients	Uniform	[2-20]
Partition-alpha	Log Uniform	[0.1-20]
Batch Size	Const.	[8-128]
lr / wd	Const.	0.01
Cut Layer	Uniform	[2-9]
Epoch	Const.	15
No. Linear layers	Const.	10
Aggregation Type	Const.	Stack, Avg
Random Seed	Uniform	[0-4]

Table 6: Hyperparameter space to examine the effect of aggregation in the central server.

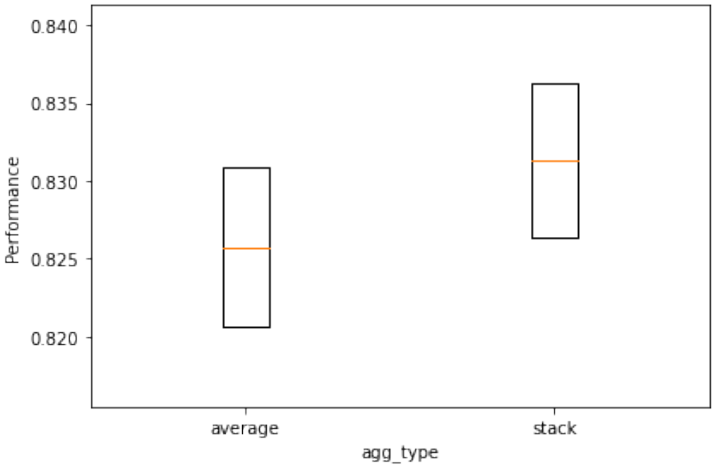
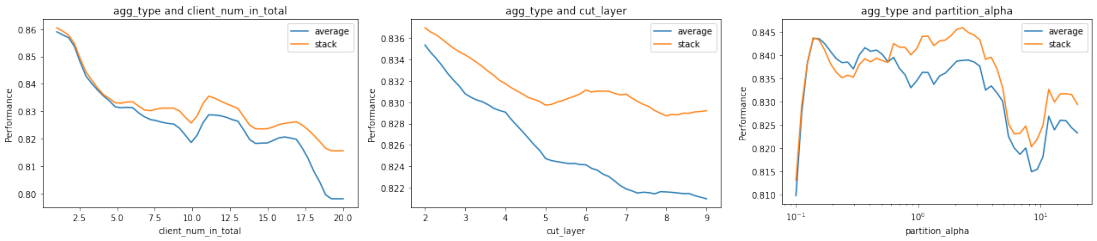


Fig. 16: Stacking shared weights in the central server has higher average accuracy. Experimented over 600 run.

As mentions in section 3.6 there are two main types of aggregation method for split-learning. The central server can either stack the weight vertically and pass them as input to the central server model or average incoming weights and pass them through the central server as input values.

In section 3.6 we argued pros and cons of each method, and in this experiment we examine the difference in performance of the mentioned implementations.



(a) (b) (c)

Fig. 17: Effect of averaging and stacking weights in the central server. (a) As the number of clients increase stacking weights result in higher accuracy. (b) Averaging higher cut-layer losses characteristics of its clients features. (c) The effect of aggregation method in unbalanced distribution of data is negligible.

Our experiments results in Figure 17 shows the pair-wise marginal values of aggregation method and data distribution hyperparameters (partition-alpha and number of clients).
As in all our experiments, we divided datasets features between clients, the data type and its homogeneity was preserved. One of the advantages of split learning is the ability of multiple clients with different data types to collaborate to train a global model. For example, patient tabular data and brain CT scans can be fed through their client-side neural network and only share the embedding layer(cut-layer) [69]. This flexibility of split learning is only valid when using the stack method of cut-layer aggregation in the central server.

5.6 Overall evaluation

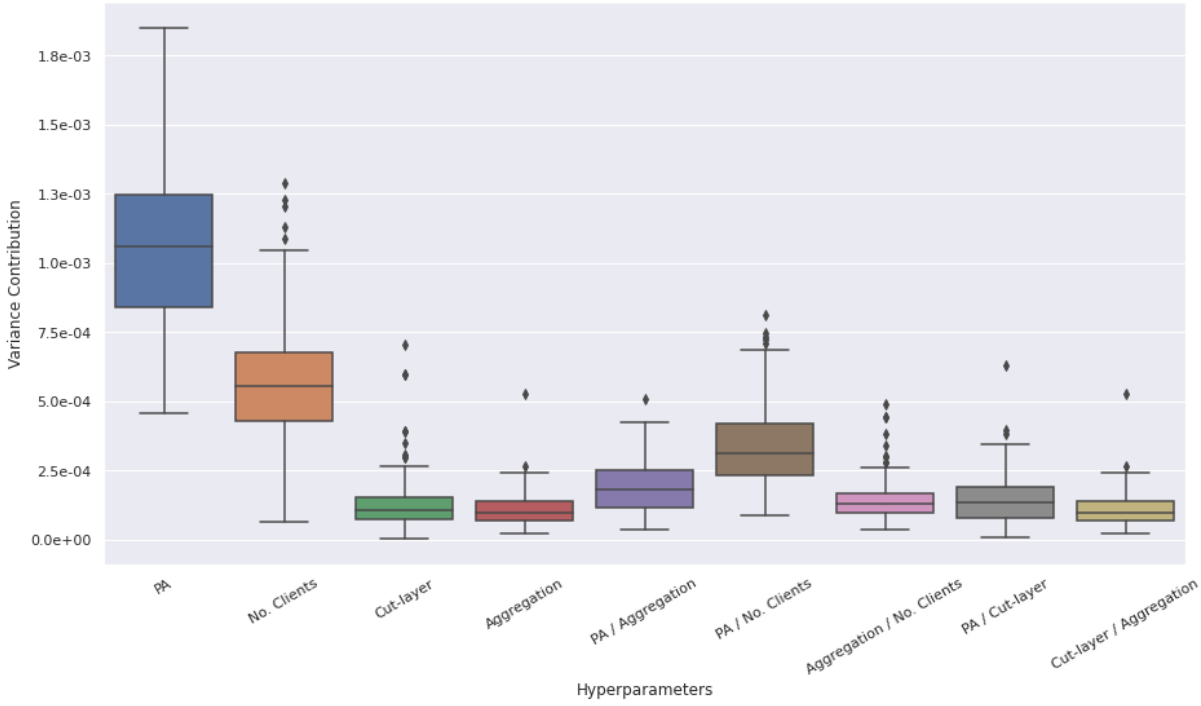


Fig. 18: Variance Contribution importance per hyperparameters across 10 datasets

6 Conclusion

Machine learning has emerged in almost every field we interact with daily, especially those with large-scale attainable datasets. We are particularly good at collecting big data and scaling up deep learning models where data regulations are loose, and most of the data is publically available. However, machine learning in health care resides on the opposite spectrum of data-full operations. Despite large-scale medical data gathered in data silos by governmental organisations, privacy and regulation concerns restrict practitioners to take advantage of its potentials. Although medical researchers are actively interested in leveraging machine learning paradigms into their research, they face very complicated, time-consuming and paperwork-heavy regulations and standards that eventually stagnate machine learning in healthcare. Data anonymisation and data masking is often proposed as a solution to remove privacy barriers, which is now well established that removing meta information from datasets does not guarantee privacy.

Federated learning can help us overcome such challenges by introducing a decentralised learning paradigm without transferring sensitive data. Under the federated setting, multiple organisations can jointly train a shared model by keeping the data local and only communicating an instance of the shared model.

This thesis took a thorough look at federated learning in a cross-silo setting. We explored Federated aggregation methods and potential improvements to increase accuracy and efficiency. The Federated averaging algorithm is often a strong baseline for optimising federated networks that work significantly well, both on vanilla federated learning and split learning for horizontally and vertically distributed data.

One of the primary concerns in the federated setting is user privacy and data regulations. Without a doubt, overcoming data regulations like GDPR and HIPAA has increased the interest of medical researchers to take advantage of distributed learning. By keeping the data local to each client, federated learning trains a global model by only combining the trained weights shared by participants in a training poll.

Several challenges regarding data distribution among participating clients were introduced. The none independent and identically distributed (non-IID) assumption and issues derived from unbalanced data distribution is the main bottleneck of federated learning. We assume a satisfied IID data distribution as a baseline for federated optimisation and further implement methods to overcome such challenges.

Several methods were also introduced to solve the non-IID phenomenon and reduce bias and instability in the training process. FedProx [44] and FOCUS [14] use the weight matching technique to normalise client contribution over the global model and reduce bias and improve accuracy.

Dividing federated data distribution settings in horizontal and vertical help us improve and design better aggregation and communication methods to train robust models. We explored federated averaging and federated stochastic gradient descent algorithms in detail.

We also introduced split learning as a solution to vertical federated learning and explored several drawbacks of this solution. Medical data is often distributed in vertical data silos. We designed our experiments to explore algorithms and implementations to tackle learning from tabular data over a wide range of distributed clients. Split learning algorithm showed prominent results and scalability for vertically stored data and overcame the limitation of vertical federated learning.

Using split learning methods, hospitals and medical institutes can jointly train for drug discovery, cancer prediction, and disease prevention models in a comprehensive privacy-preserving manner. We designed benchmarking software on top of PyTorch to get a deeper insight into split learning's solution for distributed training and explore its hyperparameter importance and model limitations. We ran over 10,000 experiments across ten benchmark datasets from OpenML CC18.

Our experiments demonstrated that the degree of unbalanced data distribution between the participating clients in a training pool plays a crucial role in the convergence and accuracy of the global network. As one client possess most of the data, it will be more similar to a data-centric training process. The cut-layer also highly affects the data distribution and how we share client weights with the central server. Low cut-layer means clients only forward pass for the very few layers and share weights with the central server. The lower the cut-layer, the more the central server becomes similar to a data-centric approach. A lower cut-layer for edge and IoT devices are also advantageous as they delegate computation to an external entity.

Bibliography

[1] Abadi, M., Agarwal, A., Barham, P.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://www.tensorflow.org/, software available from tensorflow.org

[2] Anelli, V.W., Deldjoo, Y., Di Noia, T., Ferrara, A.: Towards effective device-aware federated learning. In: International Conference of the Italian Association for Artificial Intelligence. pp. 477–491. Springer (2019)

[3] Assran, M., Loizou, N., Ballas, N., Rabbat, M.: Stochastic gradient push for distributed deep learning. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 344–353. PMLR (09–15 Jun 2019), https://proceedings.mlr.press/v97/assran19a.html

[4] Blot, M., Picard, D., Cord, M., Thome, N.: Gossip training for deep learning. arXiv preprint arXiv:1611.09726 (2016)

[5] Boughorbel, S., Jarray, F., Venugopal, N., Moosa, S., Elhadi, H., Makhlouf, M.: Federated uncertainty-aware learning for distributed hospital ehr data. arXiv preprint arXiv:1910.12191 (2019)

[6] Brisimi, T.S., Chen, R., Mela, T., Olshevsky, A., Paschalidis, I.C., Shi, W.: Federated learning of predictive models from federated electronic health records. International Journal of Medical Informatics **112**, 59–67 (2018). https://doi.org/https://doi.org/10.1016/j.ijmedinf.2018.01.007, https://www.sciencedirect.com/science/article/pii/S138650561830008X

[7] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 1877–1901. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[8] Cancer Genome Atlas Research Network, Weinstein, J.N., Collisson, E.A., Mills, G.B., Shaw, K.R., Ozenberger, B.A., Ellrott, K., Shmulevich, I., Sander, C., Stuart, J.M.: The cancer genome atlas pan-cancer analysis project. Nat Genet **45**(10), 1113–1120 (Oct 2013). https://doi.org/10.1038/ng.2764, https://www.ncbi.nlm.nih.gov/pubmed/24071849?dopt=Abstract

[9] Chen, F., Luo, M., Dong, Z., Li, Z., He, X.: Federated meta-learning with fast convergence and efficient communication. arXiv: Learning (2018)

[10] Chen, F., Luo, M., Dong, Z., Li, Z., He, X.: Federated meta-learning with fast convergence and efficient communication. arXiv preprint arXiv:1802.07876 (2018)

[11] Chen, T., Jin, X., Sun, Y., Yin, W.: Vafl: a method of vertical asynchronous federated learning. arXiv preprint arXiv:2007.06081 (2020)

[12] Chen, Y., Sun, X., Jin, Y.: Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. IEEE Transactions on Neural Networks and Learning Systems **31**(10), 4229–4238 (2020). https://doi.org/10.1109/TNNLS.2019.2953131

[13] Chen, Y., Qin, X., Wang, J., Yu, C., Gao, W.: Fedhealth: A federated transfer learning framework for wearable healthcare. IEEE Intelligent Systems **35**(4), 83–93 (2020). https://doi.org/10.1109/MIS.2020.2988604

[14] Chen, Y., Yang, X., Qin, X., Yu, H., Chan, P., Shen, Z.: Dealing with label quality disparity in federated learning. In: Federated Learning, pp. 108–121. Springer (2020)

[15] Cheng, K., Fan, T., Jin, Y., Liu, Y., Chen, T., Papadopoulos, D., Yang, Q.: Secureboost: A lossless federated learning framework. arXiv preprint arXiv:1901.08755 (2019)

[16] Dauphin, Y.N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., Bengio, Y.: Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2. p. 2933–2941. NIPS'14, MIT Press, Cambridge, MA, USA (2014)

[17] Dawson, E., Donovan, D.: The breadth of shamir's secret-sharing scheme. Comput. Secur. **13**(1), 69–78 (feb 1994). https://doi.org/10.1016/0167-4048(94)90097-3, https://doi.org/10.1016/0167-4048(94)90097-3

[18] De Cristofaro, E., Tsudik, G.: Practical private set intersection protocols with linear complexity. In: Sion, R. (ed.) Financial Cryptography and Data Security. pp. 143–159. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)

[19] Dwork, C.: Differential privacy. In: International Colloquium on Automata, Languages, and Programming. pp. 1–12. Springer (2006)
[20] Feurer, M., van Rijn, J.N., Kadra, A., Gijsbers, P., Mallik, N., Ravi, S., Müller, A., Vanschoren, J., Hutter, F.: Openml-python: an extensible python api for openml. arXiv:1911.02490 (2019)
[21] Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning. pp. 1126–1135. PMLR (2017)
[22] Geiping, J., Bauermeister, H., Dröge, H., Moeller, M.: Inverting gradients - how easy is it to break privacy in federated learning? ArXiv **abs/2003.14053** (2020)
[23] Geiping, J., Bauermeister, H., Dröge, H., Moeller, M.: Inverting gradients–how easy is it to break privacy in federated learning? arXiv preprint arXiv:2003.14053 (2020)
[24] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 169–178 (2009)
[25] Gessert, N., Nielsen, M., Shaikh, M., Werner, R., Schlaefer, A.: Skin lesion classification using ensembles of multi-resolution efficientnets with meta data. MethodsX **7**, 100864 (2020). https://doi.org/https://doi.org/10.1016/j.mex.2020.100864, https://www.sciencedirect.com/science/article/pii/S2215016120300832
[26] Goldreich, O.: Secure multi-party computation. Manuscript. Preliminary version **78** (1998)
[27] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org
[28] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2. p. 2672–2680. NIPS'14, MIT Press, Cambridge, MA, USA (2014)
[29] Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., Ramage, D.: Federated learning for mobile keyboard prediction. arXiv preprint arXiv:1811.03604 (2018)
[30] He, C., Li, S., So, J., Zhang, M., Wang, H., Wang, X., Vepakomma, P., Singh, A., Qiu, H., Shen, L., Zhao, P., Kang, Y., Liu, Y., Raskar, R., Yang, Q., Annavaram, M., Avestimehr, S.: Fedml: A research library and benchmark for federated machine learning. arXiv preprint arXiv:2007.13518 (2020)
[31] Hegedűs, I., Danner, G., Jelasity, M.: Decentralized learning works: An empirical comparison of gossip learning and federated learning. Journal of Parallel and Distributed Computing **148**, 109–124 (2021). https://doi.org/https://doi.org/10.1016/j.jpdc.2020.10.006, https://www.sciencedirect.com/science/article/pii/S0743731520303890
[32] Hemingway, H., Asselbergs, F., Danesh, J., Dobson, R., Maniadakis, N., Maggioni, A., van Thiel, G., Cronin, M., Brobert, G., Vardas, P., Anker, S., Grobbee, D., Denaxas, S.: Big data from electronic health records for early and late translational cardiovascular research: Challenges and potential. European heart journal **39** (08 2017). https://doi.org/10.1093/eurheartj/ehx487
[33] Huang, L., Shea, A.L., Qian, H., Masurkar, A., Deng, H., Liu, D.: Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. Journal of Biomedical Informatics **99**, 103291 (2019). https://doi.org/https://doi.org/10.1016/j.jbi.2019.103291, https://www.sciencedirect.com/science/article/pii/S1532046419302102
[34] Huang, M., Li, H., Bai, B., Wang, C., Bai, K., Wang, F.: A federated multi-view deep learning framework for privacy-preserving recommendations. arXiv preprint arXiv:2008.10808 (2020)
[35] Hutter, F., Hoos, H., Leyton-Brown, K.: An efficient approach for assessing hyperparameter importance. In: Proceedings of International Conference on Machine Learning 2014 (ICML 2014). p. 754–762 (Jun 2014)
[36] Jiang, D., Tong, Y., Song, Y., Wu, X., Zhao, W., Peng, J., Lian, R., Xu, Q., Yang, Q.: Industrial federated topic modeling. ACM Transactions on Intelligent Systems and Technology **12**, 1–22 (01 2021). https://doi.org/10.1145/3418283
[37] Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R.G.L., Eichner, H., Rouayheb, S.E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P.B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konecný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Qi, H., Ramage, D., Raskar, R., Raykova, M., Song, D., Song, W., Stich, S.U., Sun, Z., Suresh, A.T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F.X., Yu, H., Zhao, S.: Advances and open problems in federated learning. Foundations and Trends® in Machine Learning **14**(1–2), 1–210 (2021). https://doi.org/10.1561/2200000083, http://dx.doi.org/10.1561/2200000083

[38] Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S., Stich, S., Suresh, A.T.: SCAFFOLD: Stochastic controlled averaging for federated learning. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 5132–5143. PMLR (13–18 Jul 2020), https://proceedings.mlr.press/v119/karimireddy20a.html

[39] Kim, Y.J., Hong, C.S.: Blockchain-based node-aware dynamic weighting methods for improving federated learning performance. In: 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS). pp. 1–4. IEEE (2019)

[40] Leroy, D., Coucke, A., Lavril, T., Gisselbrecht, T., Dureau, J.: Federated learning for keyword spotting. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 6341–6345. IEEE (2019)

[41] Li, J., Tian, Y., Zhu, Y., Zhou, T., Li, J., Ding, K., Li, J.: A multicenter random forest model for effective prognosis prediction in collaborative clinical research network. Artificial Intelligence in Medicine **103**, 101814 (2020). https://doi.org/https://doi.org/10.1016/j.artmed.2020.101814, https://www.sciencedirect.com/science/article/pii/S0933365719308553

[42] Li, R., Ma, F., Jiang, W., Gao, J.: Online federated multitask learning. In: 2019 IEEE International Conference on Big Data (Big Data). pp. 215–220. IEEE (2019)

[43] Li, T., Sahu, A.K., Talwalkar, A.S., Smith, V.: Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine **37**, 50–60 (2020)

[44] Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. Proceedings of Machine Learning and Systems **2**, 429–450 (2020)

[45] Li, X., Huang, K., Yang, W., Wang, S., Zhang, Z.: On the convergence of fedavg on non-iid data. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=HJxNAnVtDS

[46] Liu, Y., Kang, Y., Li, L., Zhang, X., Cheng, Y., Chen, T., Hong, M., Yang, Q.: A communication efficient vertical federated learning framework. Unknown Journal (2019)

[47] McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics **5**(4), 115–133 (1943)

[48] McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. pp. 1273–1282. PMLR (2017)

[49] Mothukuri, V., Parizi, R., Pouriyeh, S., Huang, Y., Dehghantanha, A., Srivastava, G.: A survey on security and privacy of federated learning. Future Generation Computer Systems (10 2020). https://doi.org/10.1016/j.future.2020.10.007

[50] Narayanan, A., Shmatikov, V.: How to break anonymity of the netflix prize dataset. ArXiv **abs/cs/0610105** (2006)

[51] Open-source: Fate (federated ai technology enabler) the world's first industrial grade federated learning open source framework. (Oct 2019), https://fate.fedai.org/

[52] Pitman, J., Yor, M.: The two-parameter poisson-dirichlet distribution derived from a stable subordinator. The Annals of Probability pp. 855–900 (1997)

[53] Puri, C., Dolui, K., Kooijman, G., Masculo, F., Van Sambeek, S., Boer, S.D., Michiels, S., Hallez, H., Luca, S., Vanrumste, B.: Privacy preserving pregnancy weight gain management: Demo abstract. In: Proceedings of the 17th Conference on Embedded Networked Sensor Systems. p. 398–399. SenSys '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3356250.3361941, https://doi.org/10.1145/3356250.3361941

[54] Quoc, D.L., Fetzer, C.: Secfl: Confidential federated learning using tees. arXiv preprint arXiv:2110.00981 (2021)

[55] Habib ur Rehman, M., Mukhtar Dirir, A., Salah, K., Svetinovic, D.: Fairfed: Cross-device fair federated learning. In: 2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR). pp. 1–7 (2020). https://doi.org/10.1109/AIPR50011.2020.9425266

[56] Rieke, N., Hancox, J., Li, W., Milletari, F., Roth, H., Albarqouni, S., Bakas, S., Galtier, M., Landman, B., Maier-Hein, K., Ourselin, S., Sheller, M., Summers, R., Trask, A., Xu, D., Baust, M., Cardoso, M.J.: The future of digital health with federated learning. npj Digital Medicine **3** (12 2020). https://doi.org/10.1038/s41746-020-00323-1

[57] van Rijn, J.N., Hutter, F.: Hyperparameter importance across datasets. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (Jul 2018). https://doi.org/10.1145/3219819.3220058, http://dx.doi.org/10.1145/3219819.3220058

[58] Rocher, L., Hendrickx, J., Montjoye, Y.A.: Estimating the success of re-identifications in incomplete datasets using generative models. Nature Communications **10** (07 2019). https://doi.org/10.1038/s41467-019-10933-3
[59] Ryffel, T., Trask, A., Dahl, M., Wagner, B., Mancuso, J.V., Rueckert, D., Passerat-Palmbach, J.: A generic framework for privacy preserving deep learning. ArXiv **abs/1811.04017** (2018)
[60] Sandfort, V., Yan, K., Pickhardt, P., Summers, R.: Data augmentation using generative adversarial networks (cyclegan) to improve generalizability in ct segmentation tasks. Scientific Reports **9** (11 2019). https://doi.org/10.1038/s41598-019-52737-x
[61] Shwartz-Ziv, R., Tishby, N.: Opening the black box of deep neural networks via information. CoRR **abs/1703.00810** (2017), http://arxiv.org/abs/1703.00810
[62] Singh, A., Vepakomma, P., Gupta, O., Raskar, R.: Detailed comparison of communication efficiency of split learning and federated learning. ArXiv **abs/1909.09145** (2019)
[63] Smith, V., Chiang, C.K., Sanjabi, M., Talwalkar, A.: Federated multi-task learning. arXiv preprint arXiv:1705.10467 (2017)
[64] Sudlow, C., Gallacher, J., Allen, N., Beral, V., Burton, P., Danesh, J., Downey, P., Elliott, P., Green, J., Landray, M., Liu, B., Matthews, P., Ong, G., Pell, J., Silman, A., Young, A., Sprosen, T., Peakman, T., Collins, R.: Uk biobank: An open access resource for identifying the causes of a wide range of complex diseases of middle and old age. PLOS Medicine **12**(3), 1–10 (03 2015). https://doi.org/10.1371/journal.pmed.1001779, https://doi.org/10.1371/journal.pmed.1001779
[65] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C.: A survey on deep transfer learning. In: International conference on artificial neural networks. pp. 270–279. Springer (2018)
[66] Vanhaesebrouck, P., Bellet, A., Tommasi, M.: Decentralized collaborative learning of personalized models over networks. In: Artificial Intelligence and Statistics. pp. 509–517. PMLR (2017)
[67] Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: Openml: Networked science in machine learning. SIGKDD Explorations **15**(2), 49–60 (2013). https://doi.org/10.1145/2641190.2641198, http://doi.acm.org/10.1145/2641190.2641198
[68] Vepakomma, P., Gupta, O., Dubey, A., Raskar, R.: Reducing leakage in distributed deep learning for sensitive health data. arXiv preprint arXiv:1812.00564 (2019)
[69] Vepakomma, P., Gupta, O., Swedish, T., Raskar, R.: Split learning for health: Distributed deep learning without sharing raw patient data. CoRR **abs/1812.00564** (2018), http://arxiv.org/abs/1812.00564
[70] Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., Rellermeyer, J.S.: A survey on distributed machine learning. ACM Comput. Surv. **53**(2) (Mar 2020). https://doi.org/10.1145/3377454, https://doi.org/10.1145/3377454
[71] Wang, H., Kaplan, Z., Niu, D., Li, B.: Optimizing federated learning on non-iid data with reinforcement learning. In: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications. pp. 1698–1707 (2020). https://doi.org/10.1109/INFOCOM41043.2020.9155494
[72] Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., Khazaeni, Y.: Federated learning with matched averaging. arXiv preprint arXiv:2002.06440 (2020)
[73] Xu, J., Glicksberg, B., Su, C., Walker, P., Bian, J., Wang, F.: Federated learning for healthcare informatics. Journal of Healthcare Informatics Research **5**, 1–19 (03 2021). https://doi.org/10.1007/s41666-020-00082-4
[74] Yang, K., Fan, T., Chen, T., Shi, Y., Yang, Q.: A quasi-newton method based vertical federated learning framework for logistic regression. arXiv preprint arXiv:1912.00513 (2019)
[75] Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Trans. Intell. Syst. Technol. **10**(2) (jan 2019). https://doi.org/10.1145/3298981, https://doi.org/10.1145/3298981
[76] Ye, D., Yu, R., Pan, M., Han, Z.: Federated learning in vehicular edge computing: A selective model aggregation approach. IEEE Access **8**, 23920–23935 (2020)
[77] Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., Khazaeni, Y.: Bayesian nonparametric federated learning of neural networks. In: International Conference on Machine Learning. pp. 7252–7261. PMLR (2019)
[78] Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning (still) requires rethinking generalization. Communications of the ACM **64**(3), 107–115 (2021)
[79] Zhang, Y., Yang, Q.: A survey on multi-task learning. arXiv preprint arXiv:1707.08114 (2017)
[80] Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-iid data. arXiv preprint arXiv:1806.00582 (2018)

[81] Zhou, Y., Wu, J., He, J.: Adversarially robust federated learning for neural networks (2021), https://openreview.net/forum?id=5xaInvrGWp

[82] Zhu, H., Xu, J., Liu, S., Jin, Y.: Federated learning on non-iid data: A survey. Neurocomputing **465**, 371–390 (2021). https://doi.org/https://doi.org/10.1016/j.neucom.2021.07.098, https://www.sciencedirect.com/science/article/pii/S0925231221013254