Privacy Preserving Vertical Federated Learning: A Literature Study

Andrei Titu, Kaitai Liang, Rui Wang
Delft University of Technology

June 27, 2021

Abstract

Federated learning (FL) is a new paradigm that allows several parties to train a model together without sharing their proprietary data. This paper investigates vertical federated learning, which addresses scenarios in which collaborating organizations own data from the same set of users but with differing features. The survey provides an overview of how five alternative Vertical Federated Learning frameworks function, as well as a description of their performance and security assurances. A thorough comparison of how each of the alternatives handles the trade-offs between data privacy, framework performance, and model performance is extracted based on the examined frameworks. This allows the reader to form an opinion about benefits and disadvantages of various techniques across the vertical federated learning landscape.

1 Introduction

Machine learning has grown in popularity in recent years as a result of the almost infinite amount of data accessible, inexpensive data storage, and the development of less costly and more powerful computing. Machine learning has exploded in popularity in recent decades, refining our algorithms, assisting us in obtaining better outcomes, and expanding into new industries. It has also been a big moneymaker for corporations like Facebook, Google, and others because to its usability. It all starts with data: the bigger the data set and the more high-quality data points there are, the more accurate these machine learning algorithms may be. The more profit an algorithm can make, the more successful it is, thereby turning data into a commodity. Many sectors are now creating increasingly powerful machine learning models capable of processing more and more complicated data while producing faster, more accurate results on massive sizes. Machine learning techniques help businesses discover valuable possibilities and possible dangers more rapidly.

1.1 Federated Learning

In order to achieve predictions, traditional machine learning requires a data pipeline that uses a central server (on-premise or cloud) to host the trained model. While this structure has frequently resulted in improved services and convenience, it also has a number of drawbacks. Most notably, there has been strong opposition from individuals concerned about their privacy. All data gathered by local devices and sensors is transmitted to a central server for processing before being delivered to the devices. This round-trip invariably exchanges data with third parties, making it vulnerable to data leaks or assaults. It also restricts the capacity of a model to learn in real time.
At this point, federated learning [1] comes into play. Federated learning is a machine learning technique created by Google AI in 2016 in reaction to the GDPR [2] regulations, which made conventional

Centralized Training not only impracticable, but also unlawful in some cases. Federated learning (also known as collaborative learning) involves training an algorithm across several decentralized edge devices or servers that retain local data samples without having to exchange them. Federated learning improves algorithms by distributing the most recent version of a model to devices that are eligible. The algorithm's model then learns from the private data on a limited set of users' remote devices. When it's done, it sends a summary of the new information back to the central server; the data itself never leaves its owner. This method differs from typical centralized machine learning techniques, which need all local data sets to be uploaded to a single server, as well as more traditional decentralized alternatives, which frequently presume that local data samples are dispersed uniformly. Federated learning allows several players to construct a shared, effective machine learning model without sharing data, allowing important concerns like data privacy, security, data access rights, and access to heterogeneous data to be addressed. With this sort of learning, several industries including military, telecommunications, IoT, banking, and medical may employ more consistent, powerful, and scalable computing (Hadoop/Spark clusters, etc.) within each application. [26]

1.2 Data verticality

Federated learning can in turn be divided into multiple branches. This research paper will study in depth vertical federated learning. Vertical federated learning or feature-based federated learning - is applicable to the cases in which separate data sets share the same sample space but differ in feature space. This type of federated learning is looking to unlock the value of data that is more widely distributed, for example between different types organizations serving the same customers. Whilst the vertical federated learning has its goals set higher and including the removal of silos, there are arguably greater challenges on the security side. This paper will also provide context and descriptive overview over privacy enhancing methods, as well as high-level implementation details about how they could be integrated.

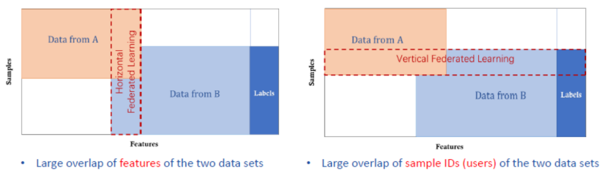

Data from A
Horizontal Federated Learning
Data from B
Labels
Samples
Features
Large overlap of features of the two data sets
Data from A
Vertical Federated Learning
Data from B
Labels
Samples
Features
Large overlap of sample IDs (users) of the two data sets

Figure 1: Vertical and Horizontal Federated Learning [3]

1.3 Purpose of the research

For an organization the problem now becomes how to choose the best approach and configuration of vertical FL which best fits their needs. This paper aims to present different existing modern VFL approaches in order to help readers form an opinion with regard to which method would be most suitable in various scenarios. The paper will approach these subject by answering the following questions:
1. What are the available technologies for vertical federated learning, and how they are implemented with respect to the performance vs. privacy trade-off?
2. How do these methods compare in terms of efficiency, complexity, security and scenarios in that they would perform best?

This paper summarizes the algorithms behind five different VFL frameworks and analyzes them in terms of performance and security assurances, in order to provide a clear picture of how existing vertical federated learning frameworks compare. This literature survey seeks to contribute to future research in this area by providing comparisons between VFL algorithms and a critical analysis of their implementations. The paper will begin with a part outlining the security terms used in the paper, followed by a section outlining the methodology used to answer the research questions, and subsequently a section outlining an analysis of each of the VFL implementations examined. Finally, a discussion will take place and the conclusion will mark the end of the study.

2 Privacy preliminaries

An introduction to privacy related terms and techniques is provided by this section as to familiarise the reader with some recurrent security topics of this paper.

2.1 Threats

Two threat models may be identified based on the predicted behavior of the parties involved in a federated learning environment:

- semi-honest (honest-but-curious): a genuine participant who will not stray from the established protocol but will make every effort to get as much information as possible from properly received messages
- malicious: an unauthorized participant who may depart from the agreed-upon protocol at any time.

Another problem discussed in this paper is one of colluding attackers which describes whether multiple adversaries can coordinate an attack . This dimension can be split into three cases:

- Non-colluding: there is no capability for participants to coordinate an attack.
- Cross-update collusion: past client participants can coordinate with future participants on attacks to future updates to the global model.
- Within-update collusion: current client participants can coordinate on an attack to the current model update.

2.2 Privacy-preserving techniques

2.2.1 Homomorphic encryption [4]

Homomorphic Encryption allows ciphertexts to be subjected to mathematical operations. Cryptography is used in this protocol to prevent eavesdroppers or even the server from obtaining the individual information summaries. The summary can only be accessed by the server once it has been combined and aggregated with the findings of hundreds or thousands of other users. We distinguish between Partially Homomorphic Encryption (PHE), which allows applying only one type of operation an unlimited number of times, Somewhat-Homomorphic Encryption (SWHE), which allows applying some operations only a limited number of times, and Fully Homomorphic Encryption (FHE), which allows applying all operations an unlimited number of times. Homomorphic encryption has the advantage of being able to separate data ownership and processing rights. In the context of federated learning, it is employed as a safeguard against attackers obtaining access to the data owners' personal information by looking at the gradients or parameters they send back to the server. These parameters can be encrypted and aggregated by the server using homomorphic encryption without the possibilty of being decrypted by the server or any attacker. The Paillier technique is a homomorphic encryption algorithm that is frequently used in FL. The product of two ciphertexts is decrypted to the sum of the two corresponding plain text values, which is the homomorphic feature of this scheme. To put it another way, for a given public key p, plaintexts m and encryption E:

$$E_p(m1) * E_p(m2) = E_p(m1 + m2)$$

2.2.2 Differential privacy [5]

Differential privacy can be used to add random data noise to an individual's summary, obscuring the results. This random data is added before the summary is sent to the server, giving the server a result that is accurate enough for algorithmic training, without the actual summary data being revealed to it. This preserves the individual's privacy. Typically Gaussian or Laplacian noise is being utilised. The quantity of noise introduced is determined by the desired level of privacy (ϵ) and the summarizing function's sensitivity. The summarizing function translates each user's private data to the set of gradients returned to the main server. By introducing noise to these gradients, it becomes more difficult to draw inferences about each participant's underlying private samples. More precisely,δ-differential privacy ensures that it is only possible to determine whether or not an element is part of a dataset with probability of at most δ based on an exposed summary of the dataset. According to [6], differential privacy has the following formal definition: A randomized mechanism M provides (ϵ, δ)-differential privacy if for all datasets D_1 and D_2 differing on at most one element, and all possible outputs of M representing the set O, the following holds:

$$P[M(D_1 \in O)] \leq e^{\epsilon} * P[M(D_2 \in O)] + \delta$$

3 Methodology

Different vertical federated learning frameworks are investigated in order to solve the questions raised by this research article. Every vertical federated learning strategy is examined by looking at the scientific publication that first introduced it. These research articles not only introduce the technique's algorithm, but also, in most cases, act as a security white paper. Furthermore, data on efficiency and accuracy is supplied, most frequently in the context of an experimental setup. As a result, the process for examining each VFL framework will be based on extracting or deriving the following major points from the research materials:

1. overview of the framework's algorithm
2. computational overhead
3. communication costs
4. accuracy
5. security model and guarantees
6. benefits and downsides
7. potential improvements

In addition, based on the seven discussion points outlined above, a comparison of the examined frameworks and privacy-preserving strategies will be made with the goal of recommending each procedure to its most suitable scenario.

4 Privacy-preserving VFL frameworks

4.1 FedBCD [7]

This study offers a new cooperatively learning approach for distributed features based on block coordinate gradient descent, in which participants change gradients locally multiple times before communicating. The number of communication rounds and overall communication overhead are considerably reduced with this algorithm. The algorithm is built on top of FedSGD and FedAvg, which will be discussed next, bringing several optimizations in order to reduce communication overhead. FedBCD's literature theoretically demonstrates that with a decay learning rate and correct choice of the number of consecutive local gradient updates in parallel done before sharing the intermediate results among distinct participants, the method achieves global convergence.

4.1.1 FedSGD and FedAVG

Due to the similar characteristics they share with their well-known centralized equivalents, these two frameworks are typical federated learning techniques that provide a thorough grasp of how federation works in this context. They will be briefly addressed here because of this, as well as the fact that FedBCD uses them for numerous comparison points.

FedSGD is based on the standard stochastic gradient descent algorithm, which is a well-known method in the field of statistical optimization. FedSGD is an extended SGD that forms the global objective function assuming k participants in the training data and n items in the input data. When using FedSGD, each edge device must communicate gradients or parameters to the server, which then averages these and applies them to new parameters. FedSGD requires that devices and servers communicate often. So much so that at each repetition, interim findings must be communicated. This might be inefficient, especially if K is big or the task requires a lot of communication. If a job necessitates pair-wise communication, the number of communications each round can be $K^2 + K$. We should remark that, because FedSGD uses the same iteration as the standard SGD algorithm, it converges at a rate of $O(1/T)$, independent of K [8]. Because each iteration necessitates one round of communication between all participants, T rounds are necessary to achieve an error of $O(1/T)$.

In FedSGD, each client uses local data to conduct gradient descent on the deployed model, and the server then calculates the average of the generated models. The FedAvg [9] is meant to increase the amount of computation performed by each client. FedAvg, in particular, iterates the local update several times before the averaging step. FedAvg, unlike FedSGD, allows each edge device to train and update parameters repeatedly using gradient descent. As a consequence, despite the fact that FedAvg has a greater demand for edge devices, it outperforms FedSGD.

4.1.2 Overview

FedBCD tackles the same problem definition as FedSGD, namely the one of K data parties collaboratively train a machine learning model based on N data samples and the feature vector are distributed among K parties The objective is for each party k to find its own optimal training parameter without sharing it or its data to other parties and therefore not using any of this information owned by the others.

In the parallel variant of the proposed method, FedBCD-p, each party conducts x successive local gradient changes in parallel at each iteration before sharing the intermediate results. The practical implementation of the paper, which found that performing multiple local steps can significantly reduce overall communication cost, is a strong motivator for such a "multi-local-step" strategy. Furthermore, such a technique is similar to the FedAvg algorithm [9], which involves each participant doing numerous local stages before aggregation. Because in each step the gradient is represented by intermediate information from the most recent synchronization, it may contain staled information and therefore no longer be an unbiased approximation of the real partial gradient. In other words, finding an unbiased estimate for the local stochastic gradient is not trivial due to the fact that following each synchronization step, each agent k executes a series of deterministic actions based on the same data set S, while simultaneously fixing the remainder of the variable blocks. This is in contrast to FedAvg-style algorithms, which sample a fresh mini-batch at each node with each iteration. On the other hand, no inter-party communication is necessary during the execution of local updates. As a result, certain intriguing tradeoffs between communication efficiency and computing efficiency are possible. On the same note, a sequential variant of the algorithm, known as FedBCD-s, allows the parties to change their local training parameters sequentially, with each update containing several local modifications and no inter-party communication.

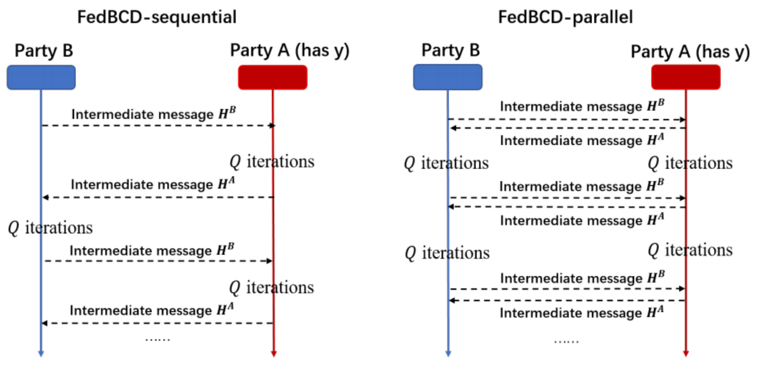

FedBCD-sequential
Party B
Party A (has y)
Intermediate message H^B
Q iterations
Intermediate message H^A
Q iterations
Intermediate message H^B
Q iterations
Intermediate message H^A
......
FedBCD-parallel
Party B
Party A (has y)
Intermediate message H^B
Intermediate message H^A
Q iterations
Q iterations
Intermediate message H^B
Intermediate message H^A
Q iterations
Q iterations
Intermediate message H^B
Intermediate message H^A
......

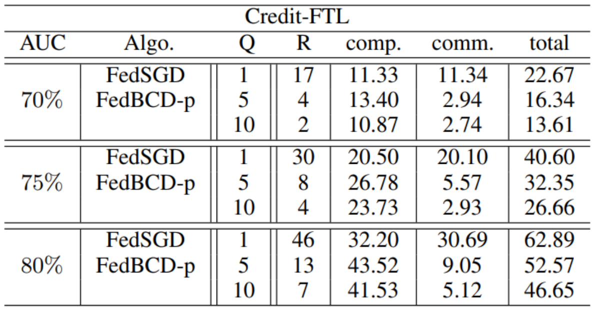
Figure 2: High-level FedBCD

4.1.3 Computational and communication complexity

The computational complexity per iteration is bounded by $O(KQ)$, since for each client node Q constant update operation are performed. This method with numerous local stages intended for the feature-partitioned jointly learning problem provides an O(1/T) bounded convergence rate when selecting the correct learning rate and batch size of local operations. Furthermore, when compared to previous distributed stochastic coordinate descent algorithms [10], our work achieves superior results. It demonstrates that, despite utilizing stochastic gradients and executing numerous local updates with staled information, just $O(\sqrt{T})$ communication rounds (out of total T iterations) are necessary to achieve an $O(1/\sqrt{T})$ rate. The convergence speed of the method is $O(\sqrt{K}/\sqrt{T})$ if we consider the influence of the number of nodes K and pick the values for the learning rate and local update number correspondingly. This means that the suggested method slows considerably as the number of parties involved increases. In reality, however, because the total number of parties engaged in a feature-partitioned cooperatively learning issue is generally not high, this aspect is minor.

4.1.4 Security guarantees

The goal here is to see if one party can learn the data of the other from a set of messages exchanged during training. Other studies looked at data leakage from exposing the entire set of model parameters or gradients [11], but in his technique, only the intermediate outcomes (such as the inner product of model parameters and feature) are revealed. The security definition states that for each party with an undisclosed dataset and FedBCD-based training settings, there are unlimited solutions that provide the identical set of contributions. That is, regardless of the total number of iterations, it is impossible to determine a specific party's data from its exchanged messages. Prior security definitions in privacy-preserving machine learning and secure multiple computing (SMC), such as [12], are in accord with this one. When an opponent has some previous knowledge of the data, he or she may be able to rule out certain potential solutions or expose some derived statistical information [13], but it is still difficult to deduce the actual raw data ("deep leakage"). This realistic and heuristic security approach, on the other hand, allows for a variable trade-off between privacy and efficiency, allowing for far more efficient solutions. Our challenge is made more difficult by the fact that the observations made by other parties are serial FedBCD algorithm outputs that are all coupled via updating equations .Although it is simple to demonstrate security when sending one round of contributions because of the decreased dimensionality, it is unknown if raw data will be exposed over hundreds or millions of iterative exchanges. The FedBCD-p algorithm's efficiency is studied when homomorphic encryption (HE) is used. While using HE to secure transmitted data provides greater security, doing calculations on encrypted data is highly computationally costly. In this case, properly choosing Q may minimize communication rounds while simultaneously introducing computational cost due to the increased total number of local iterations (Q number of communication rounds). FedBCD-p algorithm was incorporated into FATE1's existing FTL implementation and ran two-party learning simulations on two Intel Xeon Gold models with 20 cores, 80 GB of memory, and a one TB hard disk [7].



Credit-FTL						
AUC	Algo.	Q	R	comp.	comm.	total
70%	FedSGD	1	17	11.33	11.34	22.67
	FedBCD-p	5	4	13.40	2.94	16.34
		10	2	10.87	2.74	13.61
75%	FedSGD	1	30	20.50	20.10	40.60
	FedBCD-p	5	8	26.78	5.57	32.35
		10	4	23.73	2.93	26.66
80%	FedSGD	1	46	32.20	30.69	62.89
	FedBCD-p	5	13	43.52	9.05	52.57
		10	7	41.53	5.12	46.65

Figure 3: Experimental analysis of FedBCD [7]

Figure 3 summarizes the experimental findings. It indicates that FedBCD-p with a bigger Q requires fewer communication rounds and overall training time to attain a given AUC [14], with a slight increase in computation time but a communication round reduction of more than 70% when compared to FedSGD.

4.1.5 Accuracy impact

Based on MIMIC-LR and MNIST-CNN, the influence of different local iterations on the communication efficiency of both FedBCD-p and FedBCD-s algorithms is investigated (Figure 2). For varying values of the number of local updates in between communication rounds, FedBCD-s and FedBCD-p show similar convergence. Due to sequential execution, the running time of FedBCD-s is double that of FedBCD-p for the same communication round. As the number of local iterations grows, the number of communication rounds required decreases substantially. Participants' parallelism can therefore be utilized to minimize overall communication costs by lowering the number of total communication rounds necessary as a result of increasing the number of local iterations.

4.1.6 Considerable benefits and limitations

FedBCD performs extremely well in terms of communication costs but the trade-off is being made with model accuracy. In order to make the most out of this solution, various practical experiments need to be conducted for getting a feel of how this method fits a certain problem. Thorough understanding on the requirements of the problem solved with FedBCD is necessary for choosing the right size Q of the batch of operations happening in between communication rounds.

4.2 MMVFL[3]

MMVFL stands for 'Multi-participant Multi-class Vertical Federated Learning'. This framework builds on the concept of multi-view learning (MVL), in which several models are simultaneously learned for tasks involving multiple different views of the same input data, to provide a VFL framework that is suitable for multi-class classification problems involving numerous participants. To make the learning process more personalized, MMVFL, like the multi-task FL framework introduced in, learns a separate model for each participant instead of a single global model for all participants. MMVFL also allows the label owner to share labels with other participants, making federated model training easier. MMVFL is privacy-preserving, which means that data and labels do not leave their owners' possession during training. MMVFL can also calculate how much distinct attributes from each participant contribute to the FL model, since the communication, compute, and storage costs of a VFL system can be decreased for further training under incremental learning settings by removing redundant and detrimental characteristics during first training periods. MMVFL is the first VFL framework designed specifically for multi-class problems involving numerous players. It can successfully transfer label information among multiple VFL participants and match multi-class classification performance of existing techniques, according to rigorous experimental assessment.

4.2.1 Overview

The main problem with distributed multi-class classification is to ensure that all participants use the same labels. On a high-level the algorithm works as follows. It defines an order for its participants and assumes without loss of generality that the training labels are owned by the first participant. It then proceeds with all participants locally training a pseudo-label matrix on their own data-sets. These pseudo-label matrices will be the only ones leaving the participants and communicated to the server. The algorithm seeks convergence by trying to satisfy two defining constraints. So, in each iteration local and global updates are being made in order to satisfy them. The first constraint condition is global and checks that the locally learned pseudo-label matrices are equal across all participants. The other constraint condition is local and ensures that the pseudo-labels learned by the first participant are equal to the true labels. The combination of the two constraint conditions indirectly assures that the label set from all participants are equal to the true label set of the first.

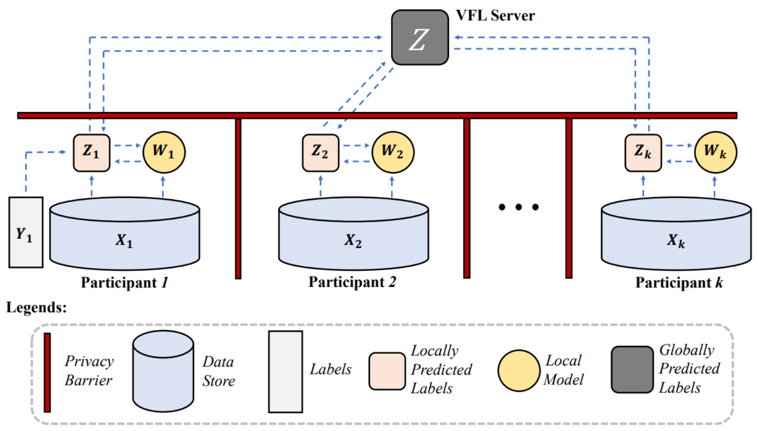


Figure 4: High-level MMVFL

4.2.2 Computational and communication complexity

The procedures involved in the optimization of the pseudo-label matrix are the most time expensive aspect of local training under MMVFL for the k-th participant. This step takes $O(d_k^3)$ time, where d_k is the dimensionality of the k-th participant's data-set. From here it can be understood why MMVFL focuses on reducing the training set dimensions as much as possible. On average, the dimensionality reduction algorithm succeeds in truncating one third of the feature space. Because of this and the fact that the proposed model necessitates per-iteration communications among all participants, each iteration of federated learning has a temporal complexity of $O(max_k(8d_k^3/27))$, implying that the time spent for FL training under MMVFL is determined by the slowest participant in each round (referred to as stragglers). Communication costs per participant are proportional to the amount of relevant features of its data-set. MMVFL strives in this aspect as it makes use of sparse learning-based unsupervised feature selection algorithms for reducing the number of features which are deterministic for the classification, and therefore usually halving the volume messages per commuication round. In terms of communication rounds, each local step will have to end with sending the updated pseudo-label matrix to the server. So the overhead is linearly proportional to the amount of steps needed for reaching convergence.

4.2.3 Security guarantees

MMVFL's key principle is that each participant learns their own model parameters locally, while the global pseudo-label matrix is updated federatedly. Only the aforementioned matrix from all participants must be transferred to the FL server in this process, whilst features and labels must be maintained locally by their owners. As a result, MMVFL allows privacy-preserving label sharing because the transformation matrices are insufficient to deduce the original data even when intercepted several times by a malicious entity.

4.2.4 Accuracy impact

In most circumstances, MMVFL's performance is comparable to, and occasionally even greater than, that of its supervised counterparts. MMVFL's classification results, which are comparable to those of the two competitors, show that it can effectively transmit label information from the label owner participant to other participants in VFL situations to train a global FL model. MMVFL can achieve equivalent or even higher performance with a lower set of essential features in some circumstances than other algorithms that use the entire dimensionality.

4.2.5 Considerable benefits and limitations

The resources required, such as transmission capacity, processing equipment, and memory requirements, can be decreased by discarding dimensions that are less relevant to the FL system based on the feature significance evaluation technique described in this work. This is especially helpful for VFL systems in incremental learning scenarios.

Without additional work, this framework is relatively inefficient with regard to communication rounds. Besides that, MMVFL has incomplete security guarantees in the case which the data owners are reticent to letting even the smallest trace of labeling information become known to potential attackers.

4.2.6 Improvements

In order to improve communication efficiency, techniques like those described in FedBCD [7] can be implemented. Multiple updates can be batched together on both server and clients in order to reduce the number of communication rounds.
Even though only pseudo parameters are sent to the central server, the cloud may be untrustworthy and allow for the theft of sensitive data from data owners. For example, [15] proved that even if only a tiny part of the gradients collected by malicious cloud is relevant information released, malicious-cloud may still exploit it. In most cases, the assault increases the amount of noise in the model. The elegant answer to this problem is Fully Homomorphic Encryption (FHE), which tries to retain the structure of ciphers such that addition and multiplication operations may be done after encryption. In reality, FHE seems to be deemed too theoretical, because additively homomorphic encryption [5] is frequently employed to assess non-linear functions in machine learning algorithms that must balance data privacy

and prediction accuracy trade-offs. [15] have developed an improved method to ensure that no data is lost to the server. All stochastic gradients can be encrypted using homomorphic encryption and saved on the cloud server, as inspired by [15]. The encrypted gradients may then be applied to models, which can make use of them in this form during computations thanks to homomorphic characteristics (addition and multiplication).

4.3 VAFL [16]

4.3.1 Overview

This novel technique enables each client to perform stochastic gradient algorithms independently of other clients, making it appropriate for customers with sporadic connectivity. This approach also makes use of a novel perturbed local embedding method to ensure data privacy while also enhancing communication efficiency. Client data is not shared with other clients nor the server in order to protect data privacy. Instead, each participant learns a local (linear or nonlinear) embedding that maps the high-dimensional data vector to a low-dimensional one, allowing the embedding vector to completely capture the local knowledge of its client. The embedding vectors and the stochastic gradients associated with these embedding vectors will be shared between the server and the clients. During the learning process, the server waits for a message from an active client, which may be either a query of the loss function's gradient in relation to the embedding vector, or a new embedding vector computed using the updated local model parameter. In response to first the query, the server computes the gradient for a certain client using its current embedding vector and sends it to the participant; and, in response to the second query, the server computes the new gradient for its model using the embedding vectors it currently has from other clients and updates its model.Each active client picks a datum at random for each interaction with the server, queries the associated gradient with reference to the embedding vector from the server, and then securely uploads the updated embedding vector. After that, it updates the local model.

4.3.2 Computational and communication complexity

Computational complexity is loosely bounded by the time to perform constant updates over T iterations for K participants, so $O(TK)$.
This technique considers two settings of the update methods in order to achieve convergence:
1. Uniformly bounded delay D. This may be accomplished by altering the server's behavior. Whenever the delay exceeds D(> 0) during the training phase, the server requests a new embedding vector from the client before continuing the server update procedure. Under the assumption that all delays are bounded by a certain constant D, the algorithm converges in $O(1/\sqrt{T})$ steps, where T is the number of total iterations done by the algorithm. Under the additional assumption of gradient convexity the convergence performance will be $O(1/T)$ bounded.
2. Stochastic unbounded delay. Each client's activation is a stochastic process in this scenario. The hitting times of the stochastic processes influence the delays. If all clients are activated using independent Poisson processes, for example, if the delays will be geometrically distributed, the time complexity is similar to the ones under bounded delay, namely $O(1/\sqrt{T})$, and $O(1/T)$ respectively if the gradient is strongly convex.

4.3.3 Security guarantees

This method provides a local perturbation mechanism that each client uses to ensure differential privacy of local information, as well as smoothing out the otherwise non-smooth mapping of local embedding. Because local clients continue to send out embedded information, it is critical to prevent any attacker from using this observation to track down a specific individual. VAFL uses the Gaussian differential privacy (GDP) introduced in [17] to provide a better compromise between privacy and accuracy. Intuitively, μ-GDP ensures that differentiating two neighboring data-sets using information given by all clients is at least as difficult as distinguishing two normal distributions with the same variance but one having mean equal to 0 and the other having mean equal to μ. Less privacy is lost when μ is smaller.

4.3.4 Accuracy impact

VAFL is integrated and tested on both federated logistic regression and federated deep learning settings, on data sets MNIST and MIMIC-III respectively. In all situations, VAFL trains a federated model with accuracies equivalent to the centralized model that requires raw data collection.

4.3.5 Considerable benefits and limitations

VAFL is fast-performing since it was built for asynchronous workloads. Its communication costs can be further reduced since, at the time, communication rounds increase linearly with the number of iterations until convergence. In the case, of lengthy training periods when the asynchrony is used mostly for dealing with low availability of client nodes, communication does not make a considerable impact. Nevertheless, in the case where asynchrony is used to speed up training, communication should not be a bottleneck.

4.3.6 Improvements

FedBCD's approach for limiting communication rounds can be employed in order to batch some local updates made by VAFL.

4.4 Pivot [18]

Pivot is a revolutionary approach for privacy-preserving vertical decision tree training and prediction that ensures no information is released except that which the clients have consented to share (i.e., the final tree model and prediction output). Pivot protects against a semi-honest adversary who may compromise k-1 out of k clients without relying on a trustworthy third party.

4.4.1 Overview

Pivot has two distinct protocols: basic and enhanced. The basic protocol ensures that no intermediate data is revealed. However, after getting the public model, colluding clients can use their own data-sets to extract private information from a target client's training dataset. This problem is addressed by the enhanced protocol. Initialization, model training, and model prediction make up the three steps of the protocol.

Initialization: In this step, the clients agree to execute a pre-defined algorithm (e.g., the decision tree model) on their shared data and distribute the pre-defined information (e.g., the trained model). The joint samples are determined and aligned cooperatively by the clients. Some hyper-parameters, such as security parameters (e.g., key size), pruning thresholds, and so on, are also agreed upon by the clients. The threshold homomorphic encryption keys are generated jointly by the clients, and each client receives the public key and a partial secret key.

Model training: Iteratively, the clients construct the designated tree model. Every iteration, the super client broadcasts some encrypted data to make it easier for the other clients to compute encrypted statistics locally. The clients then use secure calculations to turn those statistics into MPC-compatible inputs, i.e., covertly shared values, in order to calculate the best split of the current tree node. Finally, clients can update the model by revealing (in the Pivot basic protocol) or converting the privately shared best split back into an encrypted form (in the Pivot enhanced protocol). No intermediate data is revealed at any point during the process.

Model prediction: The clients receive a tree model after model training. The entire tree is released in plaintext in Pivot's basic protocol. The split threshold on each internal node and the prediction label on each leaf node are hidden from all clients in anonymously shared form in the Pivot improved protocol. The clients can collectively make a prediction given an input sample with distributed feature values. During the prediction process, Pivot ensures that no information other than the expected label is disclosed.

4.4.2 Computational and communication complexity

Pivot's both basic and enhanced versions are theoretically analysed with regard to time complexity in the original paper. This work will walk through that analysis and offer additional explanations. Let C_e and C_s be the costs of computing a homomorphic encrypted value and a secretly shared value, respectively. Because threshold decryption (which entails decryption of each client and combination via network communication) and secure comparison (which entails multi-round network communications among clients) take longer than the other computations, we analyze these two operations separately and denote their costs as C_d and C_c, respectively. Let d represent the maximum number of features a client may have, b represent the maximum number of splits a feature can have, and c represent the number of classes. Additionally, n is the number of data samples and m represents the number of separate clients.
For the model training, in each iteration of the basic protocol, a client's computational cost includes:
1. a local computation step: the encrypted label vectors computed by the super client, i.e., $O(nc)C_e$, and the encrypted statistics computed by the clients, i.e., $O(ncdb)C_e$, where db is the number of local splits;
2. MPC computation step: the MPC conversion for encrypted statistics of total splits, i.e., $O(cdb)C_d$, and the best split determined using $O(cdb)$ statistics, i.e., $O(cdb)C_s + O(cdb)C_c$, where db is the number of total splits;
3. model update step: the update of encrypted mask vectors, i.e., O(n)Ce. The overall cost is $O(ncdbi)C_e + O(cdbi)(C_d + C_s) + O(dbi)C_c$, where i is the number of internal nodes. The sole change in the enhanced protocol is two more calculations in the model update step: private split selection on b split indicator vectors, i.e., O(nb)Ce, and encrypted mask vector updating, which primarily needs n threshold decryption operations, i.e., O(n)Cd. Here the total cost is $O(ncdbi)C_e + O(cdbi + ni)C_d + O(cdbi)C_s + O(dbi)C_c$.
For model prediction, the interim computational costs with the basic protocol are $O(mi)C_e$, $O(i)C_e$, and $O(1)C_d$, which encompass updating an encrypted prediction vector with size (i+1) in a Robin round, the homomorphic dot product between the encrypted prediction vector and the plaintext label vector, and the threshold decryption of the final prediction output, respectively. The overall cost is therefore $O(mi)C_e + O(1)C_d$. The computation complexity of the enhanced protocol comprises the secure comparison of i internal nodes as well as the secure dot product of the prediction vector and the label vector, i.e. $O(i)(C_s + C_c)$.
In terms of communication, messages are being sent during both described stages, so on average communications rounds are $O(2T)$ bounded. After 2T communication rounds the algorithm will converge with an asymptotic rate of $O(2/\sqrt{T})$.
In summary, because the two additional calculations constitute extra expenses, the computational cost of the enhanced protocol is always higher than that of the basic protocol when it comes to model training. In terms of model prediction, the number of clients and the connection between cipher-text computation cost and secure computation cost determine whether the basic protocol is superior.

4.4.3 Security guarantees

Pivot assumes a semi-honest model [19], in which each client follows the protocol precisely as stated, but may attempt to deduce private information from other clients based on the messages received. The super client is treated in the same way as any other client. It is assumed that an adversary can corrupt up to k-1 clients and that the adversary's corruption technique is static, in the sense that the set of corrupted clients is fixed before the protocol execution and remains unaltered during it.
While accomplishing the desired features of the model training and model prediction stages, Pivot's fundamental protocol safeguards against a semi-honest adversary who can statically corrupt up to k-1 out of k clients. The correct and full process in which clients send their data to a trusted third party for calculation and receive the final output from that party is referred to as perfect functionality. Each client's data set provides the input to the model training stage, and the output is the trained model that everyone has agreed to release. The model prediction stage's inputs are the released model and a sample, and the predicted label of that sample is the output.
Because each node may be computed individually assuming that its result is public, the proof can be reduced to computations on one tree node for model training [20]. There are two possibilities. When a given node is an internal node, first: 1. If the super client is corrupted, nothing about the honest client's data is revealed in the local computation step; the MPC computation step is secure because the MPC conversion [21] and additive secret sharing scheme [22] are secure; finally, in the model update step, the transmitted message is secure in the case of an honest client because the threshold Paillier scheme [23] is secure. 2. If the super client is not corrupted, the only distinction is the securely communicated encrypted label information. Second, when a node is a leaf node: 1. nothing is revealed if the super client is corrupted because the honest client lacks the labels; 2. if the super client is not corrupted, the transmissible messages are the encrypted sample number of each class (for classification) and the encrypted mean label (for regression), which are both secure. As a result, the opponent learns no new information as a result of the protocol execution, and the security is preserved. In the model prediction stage, because of the threshold Paillier technique, the adversary only sees an encrypted prediction vector updated by the honest client(s) and the encrypted prediction output, thus no further information is gained (the decrypted prediction output is public).
Regarding a target client's training data-set, two possible privacy leakages are mentioned in the basic protocol: the training label leakage and the feature value leakage. The leakages are thought to be caused by colluding clients being able to divide the sample set based on split information in the model and their own data-sets.
The enhanced protocol is built for mitigating this problems. The two extra calculations (private split selection and encrypted mask vector updating) in the model update stage, which are performed using the threshold Paillier method and MPC conversion, are the sole differences from the basic protocol for model training. As a result, security follows. The attacker knows nothing except the final prediction result since the additive secret sharing technique is safe because the clients calculate a secretly shared marker for every feasible path. In addition to this, differential privacy is being incorporated in the enchanced protocol by making sure the decision tree model satisfies DP in each iteration for three different queries: pruning condition query (checks if the number of samples n on a tree node is less than a given threshold), non-leaf query (determines the best split as a whole query), and leaf query (computes the leaf label).

4.4.4 Accuracy impact

On three real-world datasets, the proposed decision tree (Pivot-DT), random forest (PivotRF), and gradient boosting decision tree (Pivot-GBDT) algorithms are compared to their non-private baselines in terms of accuracy. Each experiment is tested ten times and the average result is reported. The Pivot algorithms are accurate enough to compete with non-private baselines.

Dataset	Pivot-DT	NP-DT	Pivot-RF	NP-RF	Pivot-GBDT	NP-GBDT
Bank market	0.886077	0.886188	0.888619	0.890497	0.891271	0.892044
Credit card	0.821526	0.821533	0.823056	0.823667	0.825167	0.827167
Appliances energy	212.05281	211.45229	211.55175	211.32113	211.35326	210.75291

Figure 5: Pivot's model accuracy

4.4.5 Considerable benefits and limitations

Pivot is the first work that provides strong privacy guarantees for vertical tree-based models. The experimental results demonstrate Pivot achieves accuracy comparable to non-private algorithms and is highly efficient.

4.5 FLOP [24]

FLOP stands for Federated Learning on Medical Datasets using Partial Networks. The paper it is introduced in presents it as both a vertical and horizontal framework. The vertical setting is employed when different sections or hospitals treat the same patients, therefore resulting in disjoint information referring to the same subjects. An image classifier that is divided into a shared feature extractor and a private classifier is the example provided and experimented with in the article. FLOP's training method is identical to that of the regular vertical federated learning algorithm, with the exception that the models learned on each client are divided into two disjoint models: private and shared. The private model is solely trained on the client's private data, whereas the shared model is trained on all customers' data via federated averaging. FLOP's main goal is that of preventing attacks consisting of reverse engineering private data based on shared gradient values, by having clients only disclose a portion of the gradient values they computed locally.

4.6 Overview

Clients receive the global model from the server, and train their own model locally. They send the gradients of a partial computed version of their model back to server. The server then performs secure aggregation on the received updates from the participating clients; The server sends the aggregated results of M to clients; Clients update their model with the results from the server.

4.6.1 Computational and communication costs

Due to the simplicity of the described algorithm, the time overhead is loosely bounded by the product between the number of iterations until convergence and number of participants ($O(TK)$). The communication rounds grow linearly with the number of iterations of this method.

4.6.2 Privacy guarantees

As it is essentially a variation of the classic federated learning approach, the only difference being that only a subset of the model is shared, it will have at least the same privacy guarantees as this algorithm. [24] claim that FLOP reduces privacy and security risks by only sharing a subset of the model, however the extent to which this is the case is not discussed or presented. More research is required to verify this claim and to quantify the impact using FLOP has on privacy as opposed to classically sharing the whole model.

4.6.3 Benefits and improvements

The main benefit of this platform is its applicability to the niche medical industry. Following the described practical studies of this framework, users with similar needs can easily construct such software.

The performance and security of this framework can be drastically improved. [25] used the spirit of safe aggregation protocol to solve the challenge of computing a multiparty sum in federated learning. Based on [25], it is determined that multi-party computing (MPC) and FHE are two essential methods to federated learning, and that the aforementioned federated learning problem can be handled via FHE-based MPC. In comparison to garbled circuit-based MPC, FHE-based MPC may be completed in less rounds. As a result, a constant (at most 3) rounds threshold FHE-based MPC protocol can be designed under the common reference string model against a semi-honest adversary by combining light-weight cryptographic primitives, such as secret sharing, authenticated encryption, and SWHE, to reduce communication and computation overhead. FHE can also ensure the secrecy and confidentiality of the updates, and threshold-FHE ensures that the method can withstand users dropping out of the protocol during the recovery phase.

5 Comparison and discussion

This section reflects on the extracted features of the frameworks under consideration. Its goal is to compare them upon several levels and conduct a discussion based on the factual advantages that one has over the others.
Table 1 summarizes the frameworks on complexity aspects while Table 2 compares them in term of privacy guarantees.

	Computational complexity	Convergence complexity	Communication costs	Model
FedBCD	$O(TKQ)$	$O(1/\sqrt{T})$	$O(\sqrt{T})$	SGD
MMVFL	$O(8Td^3/27)$	$O(1/T)$	$O(T)$	Classification
VAFL	$O(TK)$	$O(1/T)$	$O(T)$	SGD
Pivot	$O(Tncdbi)C_e$	$O(2/\sqrt{T})$	$O(2T)$	Decision Tree
FLOP	$O(TK)$	$O(1/T)$	$O(T)$	Classification

Table 1: Performance

In terms of time complexity required by the computations of each algorithm, VAFL tends to be the fastest due to its asynchronous design and basic approach for updating models. This framework was built for cases in which there is uncontrollable downtime on the clients' side and therefore coordination is impossible. In theory this framework could also be used to satisfy low latency expectations when client downtime is not an issue. Another aspect to be noted about this framework is that even if in theory the convergence times have the same asymptotic bound as synchronous frameworks, in practice it should perform much faster. Looking at provided experiments, convergence times are in the top, sometimes even outperforming centralized SGD. It is followed by MMVFL which is the first framework allowing for multi-class classification training for more than two participants, happening in record times due to the feature reduction strategy it employs, even halving the feature space in some cases. Both these frameworks, however, lack communication efficiency which could prove detrimental in cases where high synchronization is needed or when dealing with unstable networking (in the eventuality of physically dispersed participants). FedBCD is highly efficient in terms of communication and it also matches computational performance of the previous two frameworks. Its optimizations make it an ideal choice for the aforementioned communication reliant cases. Nevertheless, FedBCD loses some points in terms of accuracy since gradients can become staled as a result of waiting too much before talking to the aggregator. This is why its convergence rate tends to be slower. FLOP presents impressive results given the simplicity of the algorithm, its convergence times being comparable to that of centralized classifier frameworks. Pivot is the first federated approach to the more sophisticated technique of decision tree learning and from this reason its complexity is generally higher than other studied frameworks. In provides remarkable accuracy and privacy when training distributed gradient boosting decision tree and random forest models.

	Threat model	Privacy techniques	Improvements
FedBCD	semi-honest	HE	-
MMVFL	semi-honest	-	FHE(p), FedBCD(c)
VAFL	malicious	Gaussian DP	FedBCD(c)
Pivot	semi-honest; cross-update colluding participants	HE(Paillier)	-
FLOP	semi-honest	-	FHE(s),MPC(c)

Table 2: Privacy

Most frameworks assume a semi-honest threat model which they defend against by not letting revealing information exit the scope of the client node. Instead, the models only exchange partial or altered data, making use of creative properties for the aggregation steps. In the case of Pivot, Paillier encryption is used as an additional measure. Pivot also raises the problem of colluding participants and differential privacy is engaged to prevent participants from collaborating over numerous iterations for extracting private information. For VAFL, a malicious adversarial model is assumed and therefor Gaussian DP is employed for adding noise the messages. This adds to VAFL's speed performance since DP has negligible impact on complexity in comparison to HE.

6 Responsible research

There are two components of the study process that need to be explained in the spirit of complete disclosure:

- The popularity of the privacy-preserving approach influenced the frameworks selection criteria. As far as possible, we tried to give popular, cutting-edge privacy-preserving techniques. The quantity of citations in the article or the presence of the paper in the references of the materials initially provided by the supervisor and responsible professor may have influenced the choices.
- The analysis given in the study is completely theoretical, and the only experiments reported are from the reference publications, which have been correctly mentioned. The analysis is based on knowledge and critical thinking gained throughout bachelor's studies, and the results may be easily replicated by studying the cited pieces of literature or following the explanations offered in this work. However, there may be inconsistencies between the information given in the article and the real-world use of the proposed privacy-preserving strategies in practice.

The topic of maintaining privacy has its own ethical problems, such as how user data is maintained, what constitutes sensitive data, and how the removal of an object used in training from the data-set impacts the model, among other things. However, because this work does not intend to practically improve upon the approaches, these aspects will be examined outside of its scope. Furthermore, this project had no conflicts of interest that may have impacted the researcher's neutrality. To that purpose, it should be noted that the study was not funded, and all of the paraphrases were cited, with the sources of the citations appropriately referenced.

7 Conclusion and future work

Each of the five frameworks examined has its own set of benefits and drawbacks. They made trade-offs between three factors: privacy, performance/accuracy of the federated model, and costs of the algorithm (runtime and communication overhead). The different approaches utilized by the frameworks under study point out some of the FL's most relevant dilemmas. For example, the trade-off between model accuracy and data privacy may be examined by introducing various degrees of noise to the communicated gradients, with negligible influence on training time or communication cost. Homomorphic Encryption guarantees that no information can be leaked through the weight gradients, and it does that without harming the model's quality, therefore the framework's focus changes to privacy and model correctness while sacrificing overall speed. The comparison section shows how the frameworks under consideration fit into this paradigm and which of the three elements they stress or compromise. Without a doubt, no ideal framework exists for all FL use cases, since the relevance of each of the aforementioned characteristics changes according to the circumstances. The summary and analyses offered in this article are meant to aid in selecting whether FL framework is best for a certain use case. For future work, the theoretically proposed improvements to the discussed frameworks will be pursued in order to experimentally determine their performance in a real world scenario.

References

[1] In: URL: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.

[2] In: URL: https://gdpr-info.eu/.

[3] Siwei Feng and Han Yu. "Multi-Participant Multi-Class Vertical Federated Learning". In: *CoRR* abs/2001.11154 (2020). arXiv: 2001.11154. URL: https://arxiv.org/abs/2001.11154.

[4] Abbas Acar et al. "A Survey on Homomorphic Encryption Schemes: Theory and Implementation". In: *ACM Comput. Surv.* 51.4 (July 2018). ISSN: 0360-0300. DOI: 10.1145/3214303. URL: https://doi.org/10.1145/3214303.

[5] Kang Wei et al. "Federated Learning With Differential Privacy: Algorithms and Performance Analysis". In: *IEEE Transactions on Information Forensics and Security* 15 (2020), pp. 3454–3469. DOI: 10.1109/TIFS.2020.2988575.

[6] Cynthia Dwork. "Differential Privacy: A Survey of Results". In: *Theory and Applications of Models of Computation*. Ed. by Manindra Agrawal et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1–19. ISBN: 978-3-540-79228-4.

[7] Yang Liu et al. "A Communication Efficient Vertical Federated Learning Framework". In: *CoRR* abs/1912.11187 (2019). arXiv: 1912.11187. URL: http://arxiv.org/abs/1912.11187.

[8] Saeed Ghadimi and Guanghui Lan. "Stochastic First- and Zeroth-Order Methods for Nonconvex Stochastic Programming". In: *SIAM Journal on Optimization* 23.4 (2013), pp. 2341–2368. DOI: 10.1137/120880811. eprint: https://doi.org/10.1137/120880811. URL: https://doi.org/10.1137/120880811.

[9] Xiang Li et al. *On the Convergence of FedAvg on Non-IID Data*. 2020. arXiv: 1907.02189 [stat.ML].

[10] Zhimin Peng et al. "ARock: An Algorithmic Framework for Asynchronous Parallel Coordinate Updates". In: *SIAM Journal on Scientific Computing* 38.5 (2016), A2851–A2879. DOI: 10.1137/15M1024950. eprint: https://doi.org/10.1137/15M1024950. URL: https://doi.org/10.1137/15M1024950.

[11] Reza Shokri and Vitaly Shmatikov. "Privacy-Preserving Deep Learning". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. CCS '15. Denver, Colorado, USA: Association for Computing Machinery, 2015, 1310â1321. ISBN: 9781450338325. DOI: 10.1145/2810103.2813687. URL: https://doi.org/10.1145/2810103.2813687.

[12] Olvi L. Mangasarian, Edward W. Wild, and Glenn M. Fung. "Privacy-Preserving Classification of Vertically Partitioned Data via Random Kernels". In: *ACM Trans. Knowl. Discov. Data* 2.3 (Oct. 2008). ISSN: 1556-4681. DOI: 10.1145/1409620.1409622. URL: https://doi.org/10.1145/1409620.1409622.

[13] Wenliang Du, Yunghsiang S. Han, and Shigang Chen. “Privacy-Preserving Multivariate Statistical Analysis: Linear Regression and Classification”. In: *Proceedings of the 2004 SIAM International Conference on Data Mining (SDM)*, pp. 222–233. DOI: `10.1137/1.9781611972740.21`. eprint: `https://epubs.siam.org/doi/pdf/10.1137/1.9781611972740.21`. URL: `https://epubs.siam.org/doi/abs/10.1137/1.9781611972740.21`.

[14] Jin Huang and C.X. Ling. “Using AUC and accuracy in evaluating learning algorithms”. In: *IEEE Transactions on Knowledge and Data Engineering* 17.3 (2005), pp. 299–310. DOI: `10.1109/TKDE.2005.50`.

[15] Le Trieu Phong et al. “Privacy-Preserving Deep Learning via Additively Homomorphic Encryption”. In: *IEEE Transactions on Information Forensics and Security* 13.5 (2018), pp. 1333–1345. DOI: `10.1109/TIFS.2017.2787987`.

[16] Tianyi Chen et al. “VAFL: a Method of Vertical Asynchronous Federated Learning”. In: *CoRR* abs/2007.06081 (2020). arXiv: `2007.06081`. URL: `https://arxiv.org/abs/2007.06081`.

[17] Jinshuo Dong, Aaron Roth, and Weijie J. Su. “Gaussian Differential Privacy”. In: *CoRR* abs/1905.02383 (2019). arXiv: `1905.02383`. URL: `http://arxiv.org/abs/1905.02383`.

[18] Yuncheng Wu et al. “Privacy Preserving Vertical Federated Learning for Tree-based Models”. In: *CoRR* abs/2008.06170 (2020). arXiv: `2008.06170`. URL: `https://arxiv.org/abs/2008.06170`.

[19] Payman Mohassel and Yupeng Zhang. “SecureML: A System for Scalable Privacy-Preserving Machine Learning”. In: *2017 IEEE Symposium on Security and Privacy (SP)*. 2017, pp. 19–38. DOI: `10.1109/SP.2017.12`.

[20] Rong Ma et al. “Secure multiparty computation for privacy-preserving drug discovery”. In: *Bioinformatics* 36.9 (Jan. 2020), pp. 2872–2880. ISSN: 1367-4803. DOI: `10.1093/bioinformatics/btaa038`. eprint: `https://academic.oup.com/bioinformatics/article-pdf/36/9/2872/33180817/btaa038\_supplementary\_data.pdf`. URL: `https://doi.org/10.1093/bioinformatics/btaa038`.

[21] Ronald Cramer, Ivan Damgård, and Jesper B. Nielsen. “Multiparty Computation from Threshold Homomorphic Encryption”. In: *Advances in Cryptology — EUROCRYPT 2001*. Ed. by Birgit Pfitzmann. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 280–300. ISBN: 978-3-540-44987-4.

[22] Ivan Damgård et al. “Multiparty Computation from Somewhat Homomorphic Encryption”. In: *Advances in Cryptology – CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 643–662. ISBN: 978-3-642-32009-5.

[23] Pascal Paillier. “Public-Key Cryptosystems Based on Composite Degree Residuosity Classes”. In: *Advances in Cryptology — EUROCRYPT ’99*. Ed. by Jacques Stern. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 223–238. ISBN: 978-3-540-48910-8.

[24] Qian Yang et al. “FLOP: Federated Learning on Medical Datasets using Partial Networks”. In: *CoRR* abs/2102.05218 (2021). arXiv: `2102.05218`. URL: `https://arxiv.org/abs/2102.05218`.

[25] Keith Bonawitz et al. “Practical Secure Aggregation for Privacy-Preserving Machine Learning”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’17. Dallas, Texas, USA: Association for Computing Machinery, 2017, 1175â1191. ISBN: 9781450349468. DOI: `10.1145/3133956.3133982`. URL: `https://doi.org/10.1145/3133956.3133982`.