Antti Kumpumäki

Federated Learning

School of Science

Bachelor's thesis
Espoo 29.8.2019

Thesis supervisor:

Professor. Eero Hyvänen

Thesis advisor:

Doctoral Candidate. Diego Mesquita

Aalto University
School of Science

AALTO-YLIOPISTO
PERUSTIETEIDEN KORKEAKOULU

KANDIDAATINTYÖN
TIIVISTELMÄ

Tekijä: Antti Kumpumäki

Työn nimi: Yhdistetty oppiminen

Päivämäärä: 29.8.2019 Kieli: Englanti Sivumäärä: 0+21

Koulutusohjelma: Tietotekniikka

Vastuuopettaja: Professori. Eero Hyvönen

Ohjaaja: Tohtorikoulutettava. Diego Mesquita

Yksityisyydensuoja on kasvava trendi tietotekniikassa, eikä koneoppiminen ole poikkeus. Yhdistetty oppiminen on koneoppimisen tekniikka, jolla yritetään tehdä tarkkoja koneoppimismalleja hajautetusti monella eri laitteella. Nämä laitteet yhdessä muodostavat yhteisen koneoppimismallin, jonka tekemiseen jokainen laite on osallistunut käyttämällä niiden omaa dataansa ilman, että se on pitänyt lähettää keskitettyyn tietokantaan.
Yhdistetyssä oppimisessa on omat haasteensa. Miten luoda tarkkoja malleja, kun käytettävissä oleva dataa ei välttämättä pysty tarkistamaan ja sen jakaumat saattavat olla kaukana ideaalista. Myös osallistuvia laitteita saattaa olla useampia kuin yhdelläkään yksittäisellä laitteella on näytteitä. Haasteita tulee myös siitä, miten järjestelmä hallitsee laiteparvea, joka saattaa sisältää jopa satoja tuhansia laitteita, jotka usein voivat olla epäluotettavien ja hitaiden internetyhteyksien päässä. Tiedon salaaminen aiheuttaa myös omat haasteensa, sillä koneoppimismallin päivityksestä on jossain tapauksissa mahdollista selvittää tieto, jota on käytetty mallin kouluttamisessa. Vahvat salausmekanismit saattavat myös estää tunnistamasta vihamielisiä osallistujia tarkoituksenmukaisesti myrkyttämästä mallia.
Eräitä ratkaisuja on tehty jotta, mallin pystyvät saavuttamaan samankaltaisia tarkkuuksia verrattuna keskitettyihin ratkaisuihin. Näitä ovat muun muassa *federated averaging*, *multi-task federated learning*, ja *collaborative federated learning*. federated Averaging on eräänlainen mini-batch oppimisen muunnelma. Siinä jokainen laite käy läpi ennalta säädetyn kokoisen ryhmän näytteitä, ja tekee näiden perusteella uuden päivityksen malliin. Multitask federated learning taas käyttää perinteisempää multitask learning-algoritmia niin, että yksittäinen tehtävä vastaa yksittäistä laitetta, jolloin malli pyrkii oppimaan yhdessä laitteiden välillä yleistietoa halutusta kohteesta, mutta onnistuen kuitenkin tekemään laitekohtaisia eroavaisuuksia malleihin. Collaborative federated learning taas on menetelmä jota käytetään suosittelujärjestelmien tekemiseen. Tämän avulla pystytään tekemään hajautetusti personoituja suosituksia ilman, että laitteen pitää paljastaa muille omia tietojaan. Kommunikaation haasteisiin on olemassa myös joitakin ehdotettuja ratkaisuja. Yksi niistä on erilaiset tiedon pakkausmenetelmät, kuten muodostaa koneoppimismallista matriisihajotelma, jonka merkisevimmät tiedot lähetetään koostajalle pienempänä matriisinia. Toinen vaihtoehto on jakaa laitteet keskittymään vain joihinkin tiettyihin parametreihin, ja kun järjestelmässä on tarpeeksi monta laitetta, saadaan näiden päivityksiä yhdistelemällä koottua täysi päivitys koneoppimismalliin.
Tämän kirjallisuustutkielman julkaisuhetkellä yhdistetty oppiminen on monissa tapauksissa saatu samalle tarkkuudelle keskitettyjen mallien kanssa. Myös tämänhetkiset menetelmät kommunikaatioresurssien optimoimiseksi mahdollistavat yhdistetyn oppimisen käytön suurilla laitemäärillä. Mallien salauksen takaavat menetelmät eivät kuitenkaan vielä täysin toimi yhdessä rehellisten laitteiden tunnistamisen kanssa.

Avainsanat: Federated learning, Machine learning,
Artificial intelligence

AALTO UNIVERSITY
SCHOOL OF SCIENCE

ABSTRACT OF THE
BACHELOR'S THESIS

Author: Antti Kumpumäki

Title: Federated Learning

Date: 29.8.2019 Language: English Number of pages: 0+21

Degree programme: Computer Science

Supervisor: Professor. Eero Hyvänen

Advisor: Doctoral Candidate. Diego Mesquita

Privacy is a growing trend in computer science and machine learning is not an exception. Federated learning is a new field of machine learning, in which accurate models are tried to form by using a set of distributed devices. These devices collectively will make a shared model, each of them using their data without sending it to a centralized database.

Federated Learning also has its challenges. How to create accurate models when you may not be able to inspect the data to be used, and its distributions may not be ideal. There might also be more participating clients than there are data samples on any individual device. Additional challenges come when the device cluster consists of hundreds of thousands of devices, sometimes at the end of an unreliable internet connection. Privacy is also a separate challenge, as the models may need to be encrypted, all the while making sure that the clients are honest in their model training.

Some approaches and algorithms have been proposed to face these challenges. Algorithms such as Federated Averaging and Multi-Task Learning can in some cases achieve comparable results to those achieved with centralised methods.

Research has also been done on the privacy of Federated Learning. Solutions such as Secure Aggregation have been proposed, but they still face some open questions on how to combine them with methods that can identify and exclude malicious clients.

Keywords: Federated learning, Machine learning,
Artificial intelligence

Contents

1 Introduction

Over the last decade, machine learning algorithms have started to perform in certain tasks on a human-like performance level, even surpassing it in some domains (Hu et al. 2019). Modern machine learning applications produce effective results when they have access to vast amounts of data. But in order to train suitable machine learning models, traditional machine learning algorithms require that the training data is locally available in a centralised location. However, currently large amounts of data are being generated by multiple devices and machines in increasing fashion. Collecting this data from remote devices is often resource intensive, and a lot of the data can be very privacy sensitive. In addition to that, in recent years many countries have placed restrictions on how companies are allowed to collect private data from their users, for example, the European Union implemented the General Data Protection Regulation in 2018.[1]

Federated learning is a machine learning setting, in which the model training is done on the data that the devices themselves generate or hold and this data stays local, thus meaning that no sensitive data are sent out of the device's control. As the devices local data is used to train the model, federated learning thus goes beyond other machine learning settings where a multitude of workers are being used to train a machine learning model such as distributed machine learning (Reddi et al. 2016). This bachelor's thesis focuses on federated learning in a supervised machine learning setting, therefore the data used to train the model are labeled (Kotsiantis et al. 2007). In the most popular approaches the collective learning is done by transferring locally trained machine learning models to a centralized server, where the models from multiple devices are aggregated to a single machine learning model. The aggregated model is then sent to back to the devices, and the next iteration of the model update is made based on this new aggregated model. (Konečnỳ et al. 2016).

Although this type of algorithm doesn't send the user's data to external parties, it does not guarantee that the personal data cannot be extracted from the model that was sent for aggregation (Geyer et al. 2017). The privacy of the training data also brings challenges when it comes to malicious participants taking part in the model training.

2 Research question

This thesis consists of a literature review of the current state of federated machine learning, without an empirical study. This bachelor's thesis was made by finding and reading scientific articles from Google Scholar and Scopus. Articles were found using keywords such as machine learning, federated learning, privacy in machine learning, distributed learning, collaborative filtering and various combinations of them. I evaluated the found studies based on their title, abstraction and conclusion, as well as their publishing year and the number of times they had been cited. After reading

[1] https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:02016R0679-20160504

these selected studies I wrote initial drafts of them on my skeleton framework for this work, which ultimately led to this final bachelor's thesis.

3 Federated Learning, it's characteristics and challenges

Federated Learning has a few distinct characteristics that differentiate it from other machine learning approaches. One of these is that the data used to train and the samples being inferred are split between multiple machines (in this paper these will be referred to as clients). All of these clients may possess their distinct features on this data, which will be described in more detail in 5.1. Unlike in distributed learning, in federated learning, the amount of data between these devices may vary greatly, as well as the computing power locally available on the clients. One distinct characteristic of federated learning is also the fact that many of these clients will be remotely located from each other and often the connection between them can be slow, unreliable or both. Therefore the overall system must take these features into account. (McMahan et al. 2016)

There are several statistical and architectural related challenges that are unique in the federated learning setting compared to more tradition machine learning architectures. These can be issues such as how to handle the how the learning tasks are distributed in a setting where there might be thousands of client devices, or how to train models when the data on the client devices cannot be inspected and their distributions may be unknown. (McMahan et al. 2016)

3.1 Algorithmic challenges

As datasets grow larger, distributed learning systems have been developed to increase parallelism and therefore reduce the time it takes for the model training. But often in distributed systems such as federated learning, transmitting data between separate machines is much slower than using data from the machines own memory. This encourages the system to try to do more of the work locally between communication rounds. However, with added local batch training in gradient descent based algorithms the overall convergence rate of the model will decrease as well (Smith et al. 2016). This tradeoff will add additional challenges on how much the system should tolerate communication bottlenecks, and how much it is willing to sacrifice in the model accuracy.

There are several characteristics on the data used to train the models that bring additional challenges to the federated learning process:

L1 The data collected from various devices may often be very non-*iid* (independent and identically distributed).

L2 Different clients may possess very different amounts of local training data.

L3 There may be more participating clients than there are training samples on any individual client device.

The available data may not be independent and identically distributed (iid). That means that the training data on the devices usually will not come from the same distribution, and they will have characteristics related to the user of the device. For example, the text generated from one client may be used to write scientific texts whereas another one might be used for instant messaging, or image classification model is being trained when one users photos may be mostly of dogs whereas other likes to photograph flowers.

Konečnỳ et al. (2016) proposed four separate properties that the federated learning algorithm should have:

C1 *If the algorithm is initialized to the optimal solution, it stays there.*

C2 *If all the data is on a single node, the algorithm should converge in O(1) rounds of communication.*

C3 *If each feature occurs on a single node, so the problems are fully decomposable (each machine is essentially learning a disjoint block of parameters), then the algorithm should converge in* $\mathcal{O}(1)$ *rounds of communication.*

C4 *If each node contains an identical dataset, then the algorithm should converge in* $\mathcal{O}(1)$ *rounds of communication.*

Even though optimization algorithms for distributed datasets have long been generated, such as Distributed Approximation Newton (DANE), they do not work seamlessly with distributed data that is non-iid, and in some cases they may even diverge if the distribution is not optimal (Konečnỳ et al. 2016). The amount of data on separate devices can also vary drastically. If the data in classification problems is very skewed towards one label versus the other, then with very basic classifiers the *natural thing to do* will be to predict that most of the data in the test set will also be of that same label (Provost 2000). This presents challenges to the machine learning model on how it handles these unbalances; does it treat every client equally or does it promote some clients above others. Another possible obstacle is the separate availability times for variety of distributions of data. Consider the following scenario: One proposed use case for federated learning is to do next word prediction training on mobile phones, but to save the devices battery life, the training and sending of the intermediate models would only be done when the device is plugged into a charger. As many people charge their phones primarily during the night, speakers of separate English dialects will be available for the system at separate times, thus the model would be trained to one specific data distribution, before a new distribution would arrive in a large chunk, and possibly making the model to overlook the previous distribution.

3.2 Architectural challenges

Federated learning presents certain challenges to the system architecture, such as how to manage the communication between different clients and how to guarantee the privacy of each clients data.

3.2.1 Scale of the client base

It is possible that in some cases the number of clients taking part in the federated learning can be very large, and the number of clients can be significantly larger the number of training samples on any client device. This will bring challenges on how to select the clients that take part in the training in each round. If each client machine has a relatively small amount of training data, increasing the on-device computation between each aggregation cycle becomes more challenging. As the number of clients becomes larger, the possibility of hardware and communication problems increases as well. Therefore the system has to be robust to client dropout as well as delays in the communications between device and server (Konečnỳ et al. 2016).

3.2.2 Communication resources

Because of the nature of federated learning, the communication between the clients and the server becomes an instrumental part of the performance of the whole system. As opposed to more traditional distributed machine learning methods, where the system nodes reside in a data centre with high bandwidth network connections, in federated learning the communication between the clients and the server will often be done over much slower and more unreliable connections. This brings challenges on how to handle this deficiency in the network capacity, how much local computation is done in a single training iteration and which clients to choose in a given time (McMahan et al. 2016). Another obstacle comes from the ever-increasing complexity of machine learning models. Neural networks may be made of millions of parameters (Bonawitz et al. 2017), therefore they will utilize increasing amounts of available bandwidth.

Many proposed use cases for federated learning have focused on personal level applications, such as next word prediction, which happen on mobile phones. Therefore inevitably there will be variance in both the computing capabilities, as well as the network speeds associated with the available devices. If some of the client devices will spend more time generating local updates, it may slow down the update rate of the entire system. Likewise, if some clients take a longer time to upload their local parameters, the whole update cycle will again slow down. (Nishio & Yonetani 2018)

3.2.3 Privacy

In federated learning, the information sent to the server is only the latest update to the model, a new word in a text prediction algorithm for example. This updated information is much less sensitive than the whole dataset that the client has, but it is still something that the clients may want to hide. These updates are made of new weight parameters and not of individual data entries such as typed words. However, these models can be reverse-engineered so that the data responsible for producing the latest update can be found. (Bonawitz et al. 2017)

One of the key concepts of federated learning is the ability to keep the training data private. But this comes with certain drawbacks. If the client data cannot

be inspected, then an attacker can train their local model to include "backdoor" functionality into the model that is sent for aggregation (Bagdasaryan et al. 2018). Bagdasaryan et al. were able to include backdoor functionality into the model, and they were able to evade various methods designed to combat machine learning data poisoning.

4 Proposed solutions to the challenges of Federated learning

After the publishing of McMahan et al. (2016), several studies have come forwards with various solutions which take into account many of those initial concerns that McMahan et al. (2016) had in relation to both statistical challenges with the federated learning algorithms, as well as those that were architectural and privacy related.

4.1 Algorithms

Few separate algorithms have been proposed to work in federated learning. Of these the most cited in literature is the federated averaging that was introduced in the original paper by McMahan et al. (2016), but other methods such as federated multi-task learning have also been proposed.

4.1.1 Federated Averaging

McMahan et al. (2016) proposed Federated Averaging, a modified Stochastic Gradient Descent algorithm to solve the issues related to the non-iid data. Federated Averaging combines local stochastic gradient descent with a centralised server. With federated averaging the clients will calculate a new gradient with their data, passing through their data one or more times, and then sending the gradient up to the server for aggregation. The local training can be done with varying parameters for the number of epochs (how many times the whole data is looped over), as well as the local batch size can also be adjusted. With this method McMahan et al. (2016) were able to achieve 99% test-set accuracy for non-iid partitioned MNIST dataset. For next word prediction in non-iid partitioned dataset they built from The Complete Works of William Shakespeare(Shakespeare 2007), they compared Federated Averaging to Federated Stochastic Gradient Descent on how many communication rounds it would take to reach the 10.5% accuracy. With FederatedAVG they were able to achieve it in 35 communication rounds, compared to 820 rounds it took with federated stochastic gradient descent. Federated averaging was also used by Konečnỳ et al. (2016) in their study on reducing the uplink communication costs in federated learning. More on their work in 4.2.1. Bonawitz et al. (2019) used federated averaging as well in their large scale implementation of federated learning that we cover in 6.1. Nilsson et al. (2018) also found federated averaging to provide the best results in their comparison of federated learning algorithms in section 5. Zhao

et al. (2018) also studied federated averaging with more emphasis on the datasets that are highly skewed from each other. They found out that after a certain threshold of skewness Federated Averaging starts to lose it's accuracy. Their solution to tackle this problem was to train all of the initial models first with global dataset in order to achieve the weight parameters line up, before proceeding with traditional Federated Averaging. Their results showed that this method greatly increased the accuracy of Federated Averaging with highly skewed non-iid data. On the paper made by McMahan et al. (2016) and Zhao et al. (2018) they showed that federated averaging was able to cope with the learning challenges (L1), non-iid data, as well as with the unbalanced training set sizes (L2). However, they did not consider the third learning challenge concerned with the sample size related to the number of clients.

Algorithm 1 Federated Averaging. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Initialize w_0
for each round t = 1,2... **do**
    $m \leftarrow Max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of m clients)
    for each client $k \epsilon S_t$ **in parallel do**
        $w_{t+1}^k \leftarrow$ ClientUpdate(k, w_t)
        $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
ClientUpdate(**k**,**w**) :
$B \leftarrow$ (*split* P_k into batches of size B)
for each local epoch i from 1 to E **do**
    for batch $b \in B$ **do**
        $w \leftarrow w$ - $\eta \nabla\ l(w;b)$

return w to server

4.1.2 Federated recommender systems

Many applications are used to make various recommendations to individual users such as recommending movies that the user might like or a suitable medicine to a patient. Collaborative Filtering is a widely used matrix factorization model used to solve these optimization problems. Collaborative Filtering tries to find the link in the interactions between the user and a collection of items. This information is stored in a matrix, in which there may be millions of rows representing the users, and thousands of columns representing the items. However, one common feature of these matrices is that they often are very sparse, for example each user in a database has watched only a tiny fragment of all available movies, and their opinion on the rest of the movies is unknown.

In collaborative filtering the user-item interaction matrix $\mathbf{R} \in \mathcal{R}^{N \times M}$ is tried to estimate as a combination of two lower dimensions factors $\mathbf{X} \in \mathcal{R}^{K \times N}$ representing

user factor vectors and $\mathbf{Y} \in \mathcal{R}^{K \times M}$ representing item factor vectors as:

$$\mathbf{R} \sim \mathbf{X}^T \mathbf{Y}$$

Alternating least squares in one popular matrix factorization algorithm. However, in more traditional alternating least squares implementations the user factor vectors and user-item interactions need to be present when updating the item factor vectors. Therefore applying collaborative filtering to federated learning setting requires alternative ways to optimize the lower dimension vectors. Ammad-ud din et al. (2019) developed a stochastic gradient descent based model, that uses separate algorithms for updating both the user factor matrix as well as the item factor matrix. Their method allowed the user-item interactions as well as the user factor vectors to be kept locally, while still ending up to the same solution as with the standard model done without stochastic gradient descent.

Federated Recommender systems differ from other federated learning approaches in that each client does not iterate over multiple training samples, but instead makes the calculation over the whole dataset.

4.1.3 Multi-task learning

Multi-task learning is a machine learning process in which multiple different, but possibly related, tasks are being modeled by a single machine learning model. Learning multiple tasks that may be related can offer better results than learning all of these models individually, for example in cases where there isn't a lot of training data for each separate task, but "pooling" them together enables the model to be trained with a larger set of samples (Argyriou et al. 2007).

Multi-task learning algorithm can be reprsented in the following way:

$$\min_{\mathbf{W},\Omega} \left\{ \sum_{t=1}^{m} \sum_{i=1}^{n_t} \ell_t \left(\mathbf{w}_t^T \mathbf{x}_t^i, y_t^i \right) + \mathcal{R}(\mathbf{W}, \mathbf{\Omega}) \right\}$$

$\mathbf{W} := [\mathbf{w}_1, \ldots, \mathbf{w}_m] \in \mathbb{R}^{d \times m}$ represents the weight vectors for the tth task. $\mathcal{R}(\mathbf{W}, \mathbf{\Omega})$ in turn represents the regularization function that is used to keep the model from overfitting to the training data. Inside the regularization function is Ω, which is a matrix measuring the similarities between the various tasks.

Works by Jaggi et al. (2014) and Smith et al. (2017) have introduced multi-task learning methods which treat individual clients as distinct tasks in a combined model. However, the multi-task framework CoCoA introduced in Jaggi et al. (2014) doesn't take into consideration the varying network connections of the client nodes or their possibility of dropping out. Smith et al. (2017) developed *Mocha* to generalize the distributed optimization method CoCoA. It is based on a converging bi-convex alternating approach, and it uses weight parameters that change if client nodes are unresponsive or slow to connect, to deal with the changing availability and dataset sizes of separate nodes, thus making it more robust to work in real-world scenarios. Smith et al. (2017) argued that their multi-task learning for federated setting outperformed both the local model, which was made by learning a separate

model for each task, as well as a global model, which is based on learning a single model on the whole dataset, similar to other federated learning methods. However, it is unclear how this setup would suit a setting with a very large number of clients. Like Federated averaging, Multi-Task Learning works with the unbalanced and non-iid data (learning challenges L1 and L2), but like Federated Averaging, the third learning challenge is not considered.

4.2 Architecture and system solutions

Besides finding ways to improve the accuracy of federated learning models, there has also been research on the architecture surrounding the federated learning algorithm. Ways to decrease the amount of data needed for any individual update as well as various privacy related topics are an active field of research.

4.2.1 Strategies for improving communication efficiency

After the local machine learning model has been generated, it needs to be sent to the server responsible for aggregating the models. A naive approach to this would be to send the whole model to the server. But this approach could lead to a multitude of problems, for example very complicated machine learning models can be significantly large, and on mobile networks upload speeds are significantly slower than they are in download phase.

Konečnỳ et al. (2016) tried two separate ways of updating the model, structured and sketched updates, that use less communication bandwidth, as the whole model is not sent for aggregation, but without compromising the model accuracy. In structured update, they had a previously specified structure on the update, and thus the update was calculated only on these variables, and not on the whole matrix. Matrices can be approximated as a product of two or more matrices. This low rank matrix is an approximation of the initial parameter matrix, without the least meaningful values.

In the sketched update, they first computed the whole update matrix and then used lossy compression i.e. some of the new information is lost once the update is sent back for aggregation. One of these compressions was subsampling, in which each client sends a random subset of values of the complete update matrix, but with large enough set of clients these subsets make up of the whole parameter matrix.

4.2.2 Privacy

In Bonawitz et al. (2017) they propose a method called secure aggregation, which intends to prevent the client data from being reverse-engineered from the models. The secure aggregation process has 4 steps. In the first 2 steps, the client devices share secret values among themselves. In the 3rd step, the clients send their encrypted models for aggregation, and in the 4th step clients send their secret values, but only after the aggregation has been done, from which the secret values are removed. This way all of the added noise is removed from the final encryption, but the individual models were kept encrypted along the way (Bonawitz et al. 2017).

The secure aggregation method is used in the Gboard implementation of federated learning in 6.1.

However, Bagdasaryan et al. (2018) was able to create a model that was accurate with the rest of the data set, but it included few tampered sample groups that were made to be classified inaccurately. Even though the client models are aggregated together, the attacker can scale up the malicious model weights in a way that they survive the aggregation. This method is illustrated in the figure 1.

Fung et al. (2018) proposed a method that compares the cosine similarity between the gradients and penalises those that are too close to each other to prevent malicious backdoor attacks. This solution however cannot be used with secure aggregation, since with that the outlier gradients cannot be tracked to the suspicious clients.

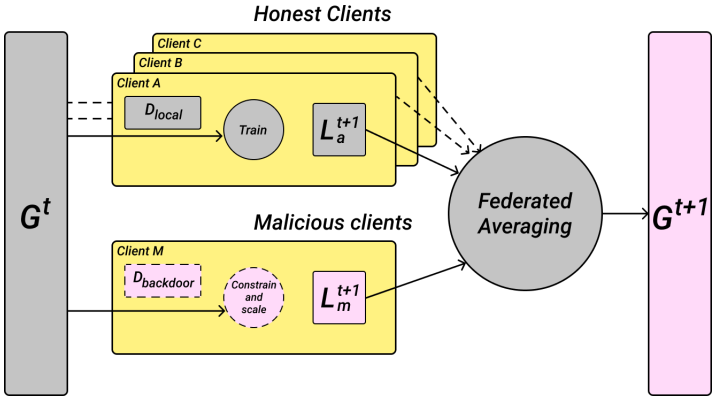

Figure 1: Malicious client inserts backdoor functionality into the aggregated model in light pink

5 Other developments of federated learning

Variational inference is a method that transfers complex inference problems into high-dimensional optimization problems (Hoffman et al. 2013). Variational inference has become one of the most popular methods for approximate inference, and it has been widely used for Gaussian process models, latent topic models, as well as deep generative models. (Bui et al. 2018).

In their paper Bui et al. (2018) the researchers presented a framework called Partitioned Variance Inference that unifies several variance inference algorithms under one framework, that can also be used to compute Bayesian Neural Networks in a federated learning environment.

Nilsson et al. (2018) compared three separate Federated learning algorithms to each other and a single central model. The algorithms used were Federated Averaging, Federated Stochastic Variance Reduced Gradient (FSVRG), as well as CO-OP. They measured the performance of these three algorithms on their accuracy after a *selected number of communication rounds*, as they argued that because the communication resources pay such an important role in federated learning, therefore stopping at converge would not be as useful criteria. In their study they used the

MNIST dataset of images of handwritten digits, and they tested the algorithms with both iid and non-iid partitioned training data. They found that Federated averaging performed the best of the three federated algorithms, and it had similar results to the central model with iid data, but not with the non-iid partitioned set. Although they too acknowledged that they used equivalently sized datasets on each client, and future work could be done on measuring the performance with unevenly sized training datasets.

5.1 Data partitioning in Federated Learning

Yang et al. (2019) introduced the concept of dividing federated learning into two separate branches based on how the data is partitioned between the clients. Most of the research cited in this bachelor's thesis has focused on what Yang et al. (2019) calls *Horizontal federated learning*, but they also proposed a method that could work when different features from the same sample are divided between separate clients.

Horizontal federated learning, figure 2, is used when all of the samples contain the same feature space, but the samples are distributed across various locations. For example, various people can have the same mobile phone and the pictures they take on them will include the same feature space, but the samples all possess a distinct sample-ID (Yang et al. 2019). In horizontal federated learning each edge node, which could be a mobile phone (Konečnỳ et al. 2016) or a server holding banks local branches customer data, will train a new model based on their local data (Yang et al. 2019).

In vertical federated learning illustrated in figure 3, the participants have the same sample-IDs, but their data contains a separate set of features. For example consider two businesses that have mostly the same customer base, but collect a separate kind of data from their clients. In vertical federated learning, the whole dataset is used to create a *"common wealth"* model which could produce information that the participants find useful, but which could be inaccessible without vertical federated learning. In federated learning the participating clients need a third-party collaborator that helps with the encryption of the model parameters.(Yang et al. 2019)

6 Frameworks and notable implementations

Several popular machine learning frameworks have implemented federated learning into the them, such as TensorFlow and PyTorch. And even though federated learning is fairly recent concept in the machine learning literature, Bonawitz et al. (2019) described how federated learning has been used in Googles intelligent keyboard Gboard.

6.1 Federated learning in Google Gboard

Perhaps the most comprehensive Federated Learning system to date is presented in a paper by Bonawitz et al. (2019), illustrated in figure 4. In the article they

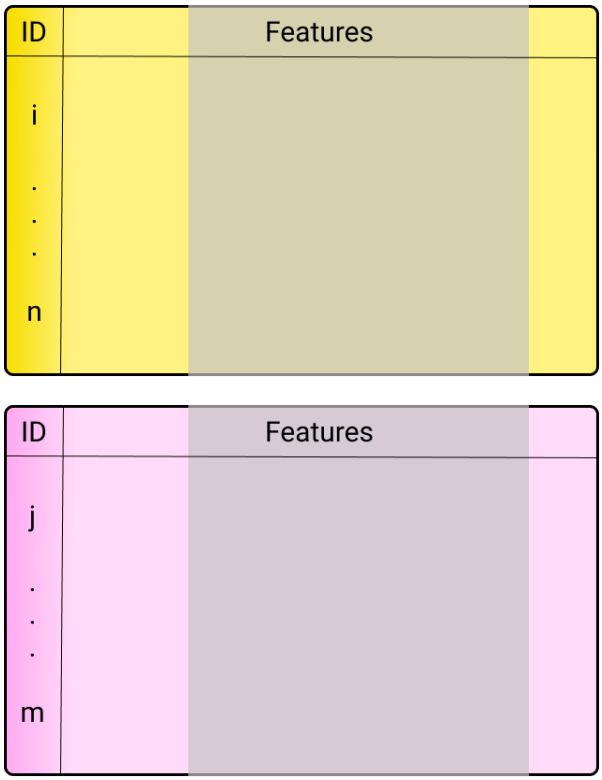

Figure 2: Horizontal federated learning, all participating devices contain data containing the same features. Data in grey is used in the model

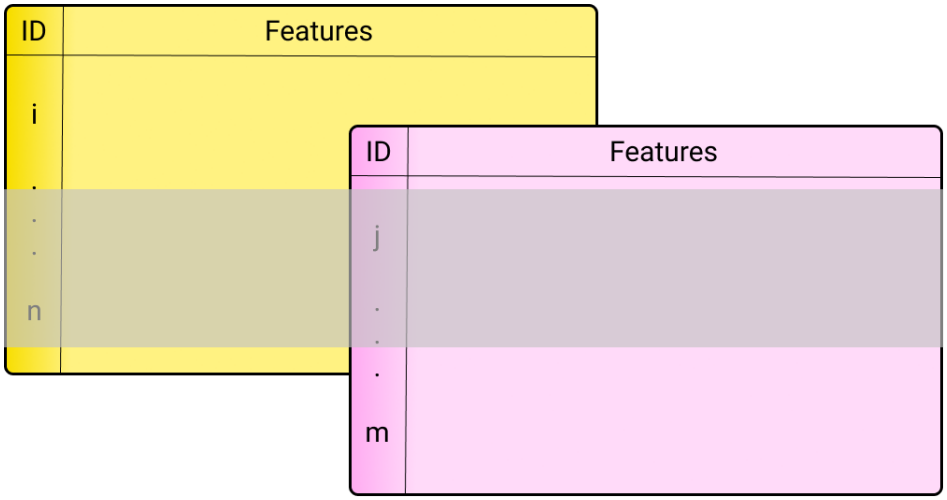

Figure 3: Vertical Federated learning, various devices may hold separate feature values from same sources. Data in grey is used in the model

present a setting currently in commercial use, which they argue has a solution to many of the initial challenges of federated learning, such as the threat of reverse-engineering the data from the models as well as possible communication resource problems. In their system, they used an architecture in which there is one central

coordinator, under which there is ephemeral master aggregator, which in turn creates multiple aggregators to which the client devices connect to. There are also persistent selectors, which forward clients to the right aggregator. Because communication costs related to secure aggregation grow rapidly and establishing communication between all clients becomes unattainable as the number of devices grows large, in their system a smaller group of devices are connected to a single aggregator, and in this layer secure aggregation is used. These aggregators then aggregate their devices models, which are in turn sent over to the master aggregator, which aggregates the full model update, which will be given to the coordinator.

Because of the ephemeral nature of the middle layers, the system isn't vulnerable to attacks that target distributed storages, because in their system there aren't any, because the models are not stored after they have been sent upwards. However, as Bagdasaryan et al. (2018) demonstrated, secure aggregation does not prevent attackers from inserting backdoor functionality into the model.

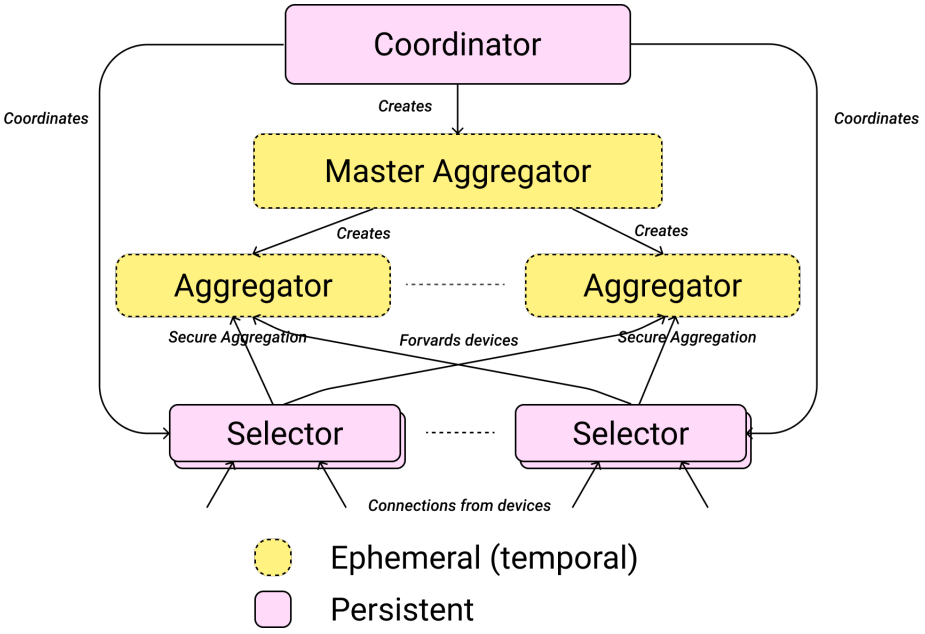

Figure 4: Actors in the FL server architecture in Bonawitz et al. (2019) implementation

6.2 LEAF

Leaf is an open-source benchmark framework which consists of datasets suitable for federated learning, a set of metrics and statistics to measure the performance of federated learning models, and a set of federated learning implementations (Caldas et al. 2018). Caldas et al. (2018) argue that most federated learning research has been done using datasets that are either not publicly available, or on datasets that are artificially made to mimic data from federated learning setting in a way that do not realistically represent the real-world data, such as splitting the MNIST (LeCun

1998) dataset by each digit. By the time of publishing this bachelor's thesis LEAF had three separate datasets:

- (1) *Federated Extended MNIST (FEMNIST)*, which is a modification on the popular MNIST digit classification dataset. In FEMNIST the data is split based on the individual who wrote the character.
- (2) *Sentiment140* is a dataset containing twitter messages, which are partitioned on the author of the tweet.
- (3) *Shakespeare* is based on The Complete Works of William Shakespeare, and it is partitioned based on each speaking role.

6.3 TensorFlow Federated

Tensorflow is an open-source symbolic math library, that is widely used in machine learning. Tensorflow started as a library to be used internally at Google but has since become publicly available to everyone. Tensorflow provides a high level federated learning interface, *TensorFlow Federated* which offers tools that can be used for example to wrap existing TensorFlow models to work in federated learning setting as well as builders that construct federated computations. The interface also gives access to datasets that are designed for federated learning scenarios, which can be used to test and evaluate federated learning models. The Tensorflow federated interface is built on top of Tensorflow Federated Core, which is a programming environment for distributed computations. One of the key advantages of TensorFlow is that it can be deployed in variety of platforms such as mobile devices.

6.4 PySyft

PyTorch is a popular machine learning library made for Python. It is a Python implementation of the Torch library, which was originally implemented with C. PyTorch is able to use GPUs in order to achieve faster computation. PyTorch differs from TensorFlow by being strictly made for Python. This brings the advantage that developers can use existing Python libraries to work with their models, whereas in Tensorflow the code written will often be translated to another language on which the TensorFlow is run. PySyft is an open-source project that allows the implementation of federated learning, secure multiparty computation and differential privacy into PyTorch. PySyft is built in a way that existing PyTorch commands can be used to work with PySyfts chain abstractions. (Ryffel et al. 2018)

7 Conclusions

Model accuracy in federated learning can be comparable to a centralized setting, as the results of federated averaging and multi-task learning demonstrate. However, as Zhao et al. (2018) demonstrated, significant asymmetries can make models less accurate. Also, the functionality of proposed solutions with uneven dataset sizes is something that requires some attention as Nilsson et al. (2018) suggests, but hasn't been fully studied in the materials cited in this thesis.

Secure aggregation has been shown to protect the clients' dataset from reverse engineering. However, there are still some issues that are only vaguely studied, such as the capability to prohibit malicious actors from inserting backdoor functionality while still being able to keep honest clients data secure. Also, the robustness of the algorithms in the presence of unreliable client devices is something that is still only vaguely studied. One active area of research is focusing on decreasing communication bottlenecks from algorithms.

Even more powerful mobile devices are spreading across the world and even more people have access to devices with a fast internet connection and relatively good computing power, and i can see that there will be a growing demand for on-device machine learning applications. Also, a lot of very personal data is being collected in hospitals and other healthcare facilities and services. Machine learning has made notable improvements in predictive healthcare, but a lot of this very valuable data is beyond the reach of other scientists because of the laws that prevent institutions from publishing this data. As many machine learning models can get higher accuracies as they have access to larger amounts of training data, federated learning could be used to develop better models while still keeping this data private.

The concept of federated learning is fairly recent, therefore most of the studies have been published in the past few years. As a result, there were few separate branches of studies made concerning federated learning such as the works on privacy and federated averaging, but there wasn't an abundance of material from which to filter the most relevant ones.

Many studies highlighted the importance of finding ways to reduce the communication overhead, which will most likely still play a major role in future research and implementations even in the age of 5G mobile networks, which has been predicted to move more computation outside of client machines. Bonawitz et al. (2019) suggested that federated learning could be extended to work in more general tasks such as federated analytics and other MapReduce-like functionalities, and Bui et al. (2018) proposed that federated learning could in the future be done without a dedicated model aggregator, but instead using peer to peer model sharing. Decreasing the required training time is also something that many studies planned to investigate.

Federated learning could also find usage in various IoT machines and network devices that generate a lot of sensory data. These devices may generate such amounts of data, that sending all of this to a remote server would use such a large part of the available network connection that it would dramatically limit the usability of these devices. On the other hand, a lot of this data can also be privacy sensitive, for personal or for business reasons. Therefore if one would like to be able to create a

model that would predict the health of the system in the future, as well as tag into the resources of other similar devices, federated learning could be a valid choice.

The results achieved with federated learning models are encouraging, however, there are still some open questions related to how these models perform with datasets that are more diverse and unbalanced than the model datasets used in most of the studies. Also, the privacy and model authenticity questions need more research before federated learning can fully be implemented in those applications that require a high level of privacy and model authenticity, such as those proposed for healthcare use.

References

Ammad-ud din, M., Ivannikova, E., Khan, S. A., Oyomno, W., Fu, Q., Tan, K. E. & Flanagan, A. (2019), 'Federated collaborative filtering for privacy-preserving personalized recommendation system', *arXiv preprint arXiv:1901.09888* .

Argyriou, A., Evgeniou, T. & Pontil, M. (2007), Multi-task feature learning, *in* 'Advances in neural information processing systems', pp. 41–48.

Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D. & Shmatikov, V. (2018), 'How to backdoor federated learning', *arXiv preprint arXiv:1807.00459* .

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecny, J., Mazzocchi, S., McMahan, H. B. et al. (2019), 'Towards federated learning at scale: System design', *arXiv preprint arXiv:1902.01046* .

Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A. & Seth, K. (2017), Practical secure aggregation for privacy-preserving machine learning, *in* 'Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security', ACM, pp. 1175–1191.

Bui, T. D., Nguyen, C. V., Swaroop, S. & Turner, R. E. (2018), 'Partitioned variational inference: A unified framework encompassing federated and continual learning', *arXiv preprint arXiv:1811.11206* .

Caldas, S., Wu, P., Li, T., Konečnỳ, J., McMahan, H. B., Smith, V. & Talwalkar, A. (2018), 'Leaf: A benchmark for federated settings', *arXiv preprint arXiv:1812.01097* .

Fung, C., Yoon, C. J. & Beschastnikh, I. (2018), 'Mitigating sybils in federated learning poisoning', *arXiv preprint arXiv:1808.04866* .

Geyer, R. C., Klein, T. & Nabi, M. (2017), 'Differentially private federated learning: A client level perspective', *arXiv preprint arXiv:1712.07557* .

Hoffman, M. D., Blei, D. M., Wang, C. & Paisley, J. (2013), 'Stochastic variational inference', *The Journal of Machine Learning Research* **14**(1), 1303–1347.

Hu, L., Bell, D., Antani, S., Xue, Z., Yu, K., Horning, M. P., Gachuhi, N., Wilson, B., Jaiswal, M. S., Befano, B. et al. (2019), 'An observational study of deep learning and automated evaluation of cervical images for cancer screening', *JNCI: Journal of the National Cancer Institute* .

Jaggi, M., Smith, V., Takác, M., Terhorst, J., Krishnan, S., Hofmann, T. & Jordan, M. I. (2014), Communication-efficient distributed dual coordinate ascent, *in* 'Advances in neural information processing systems', pp. 3068–3076.

Konečnỳ, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T. & Bacon, D. (2016), 'Federated learning: Strategies for improving communication efficiency', *arXiv preprint arXiv:1610.05492* .

Kotsiantis, S. B., Zaharakis, I. & Pintelas, P. (2007), 'Supervised machine learning: A review of classification techniques', *Emerging artificial intelligence applications in computer engineering* **160**, 3–24.

LeCun, Y. (1998), 'The mnist database of handwritten digits', *http://yann. lecun. com/exdb/mnist/* .

McMahan, H. B., Moore, E., Ramage, D., Hampson, S. et al. (2016), 'Communication-efficient learning of deep networks from decentralized data', *arXiv preprint arXiv:1602.05629* .

Nilsson, A., Smith, S., Ulm, G., Gustavsson, E. & Jirstrand, M. (2018), A performance evaluation of federated learning algorithms, *in* 'Proceedings of the Second Workshop on Distributed Infrastructures for Deep Learning', ACM, pp. 1–8.

Nishio, T. & Yonetani, R. (2018), 'Client selection for federated learning with heterogeneous resources in mobile edge', *arXiv preprint arXiv:1804.08333* .

Provost, F. (2000), Machine learning from imbalanced data sets 101, *in* 'Proceedings of the AAAI'2000 workshop on imbalanced data sets', Vol. 68, AAAI Press, pp. 1–3.

Reddi, S. J., Konečnỳ, J., Richtárik, P., Póczós, B. & Smola, A. (2016), 'Aide: Fast and communication efficient distributed optimization', *arXiv preprint arXiv:1608.06879* .

Ryffel, T., Trask, A., Dahl, M., Wagner, B., Mancuso, J., Rueckert, D. & Passerat-Palmbach, J. (2018), 'A generic framework for privacy preserving deep learning', *arXiv preprint arXiv:1811.04017* .

Shakespeare, W. (2007), *The complete works of William Shakespeare*, Wordsworth Editions.

Smith, V., Chiang, C.-K., Sanjabi, M. & Talwalkar, A. S. (2017), Federated multi-task learning, *in* 'Advances in Neural Information Processing Systems', pp. 4424–4434.

Smith, V., Forte, S., Ma, C., Takac, M., Jordan, M. I. & Jaggi, M. (2016), 'Cocoa: A general framework for communication-efficient distributed optimization', *arXiv preprint arXiv:1611.02189* .

Yang, Q., Liu, Y., Chen, T. & Tong, Y. (2019), 'Federated machine learning: Concept and applications', *ACM Transactions on Intelligent Systems and Technology (TIST)* **10**(2), 12.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D. & Chandra, V. (2018), 'Federated learning with non-iid data', *arXiv preprint arXiv:1806.00582* .