EMERGING AND FUTURE TRENDS REPORT ON FEDERATED LEARNING

ZHAO NING

1904564@student.uwtsd.ac.uk

School of Applied Computing

University of Wales Trinity Saint David

2020.05

EMERGING AND FUTURE TRENDS REPORT ON FEDERATED LEARNING

ABSTRACT

The objective of this report is to identify the emerging Federated Learning approach as a promising future trend in distributed machine learning.

First of all, the research background and a concise introduction to this research topic are presented. Next, key algorithms, frameworks and practical applications with regard to real-life learning tasks in this field are briefly discussed. Then three major challenges that Federated Learning is facing are outlined: communication-efficiency statistical heterogeneity, and adversarial attacks. Relevant literature aimed at addressing these challenges are critically reviewed. Finally, a conclusion is drawn based on all the discussions, and potential directions of future work are provided.

Keywords: Distributed Machine Learning, Federated Learning, Communication-efficiency, Statistical Heterogeneity

TABLE OF CONTENTS

ABSTRACT **2**

TABLE OF CONTENTS **3**

LIST OF FIGURES **4**

CHAPTERS

Chapter 1. Introduction **5**

Chapter 2. Main Concepts and Techniques **6**

2.1 System Design 6

2.2 Key Algorithms 7

2.3 Key Frameworks and Applications 8

Chapter 3. Major Challenges **9**

3.1 Improving Communication-efficiency 9

3.2 Combating Statistical Heterogeneity 12

3.3 Dealing with Adversarial Attacks 15

Chapter 4. Conclusion and Future Trends **16**

REFERENCES **17**

LIST OF FIGURES

TITLE	PAGE
Figure 1. The Lifecycle of Federated Learning [3]	6
Figure 2. The Federated Averaging (FedAvg) Algorithm [2]	7
Figure 3. Next word predictions in GBoard [6]	8

Chapter 1. Introduction

The past decade has witnessed the prevalent successes in machine learning tasks such as image classification, face/speech recognition, and machine translation. These achievements in artificial intelligence can be attributed to three main aspects: 1) an explosive growth in data generated/stored in diverse locations such as data centers, mobile phones/tablets and IoT sensors; 2) impressive improvements in computation capabilities of modern processors; and 3) huge advances in machine learning techniques like convolutional neural networks (CNN), recurrent neural networks (RNN) and etc.

In recent years, the speed of increase in the amounts of data has risen above that in the processing power of modern computers, and therefore the demand for developing distributed machine learning approaches, in which workload is distributed in a de-centralized way across multiple devices, has also increased [1].

Among these approaches is **federated learning** (**FL**, also known as **federated optimization**) [2]which preserves user data privacy by training both private local models on user devices and a shared global model on a central server. FL is a learning paradigm that pulls training tasks to the data, instead of pushing data to the training code. Other than data privacy, communication efficiency is another important criterion for assessing learning performance in FL settings, due to the fact that network connections between mobile clients and servers are often costly, unreliable or have high latency.

Typical problems that FL tends to solve have the following properties [2]:

- training on real-world data from large amounts of mobile devices has a clear edge over training on auxiliary/proxy data available in data centers;
- compared to the size of models, the size of user data is much larger, and therefore in order to decrease communication cost, rather than sending data back and forth, it is preferable that only model updates (e.g. gradients or weights) are communicated between clients and servers.

Chapter 2. Main Concepts and Techniques

2.1 System Design

There are several phases in a typical lifecycle of federated learning, as is shown in *Figure* 1[3, 4], including:

- the **Client Selection** and **Broadcasting** stage during which the server selects a number of eligible devices (e.g. devices that are charging or on Wi-Fi), towards which the central server broadcasts the shared global model;
- the **Local Training** and **Reporting** stage during which clients perform local training without their sensitive data leaving the local devices, and then individual model updates are reported to the central parameter server;
- the **Global Aggregation** stage during which the central server collects all local updates uploaded by participating clients in stage 2.

A complete training task is carried out by iteratively going through the stages mentioned above till the shared global model converges with an acceptable test accuracy, and is ready for mass deployment to a large scale of distributed devices.

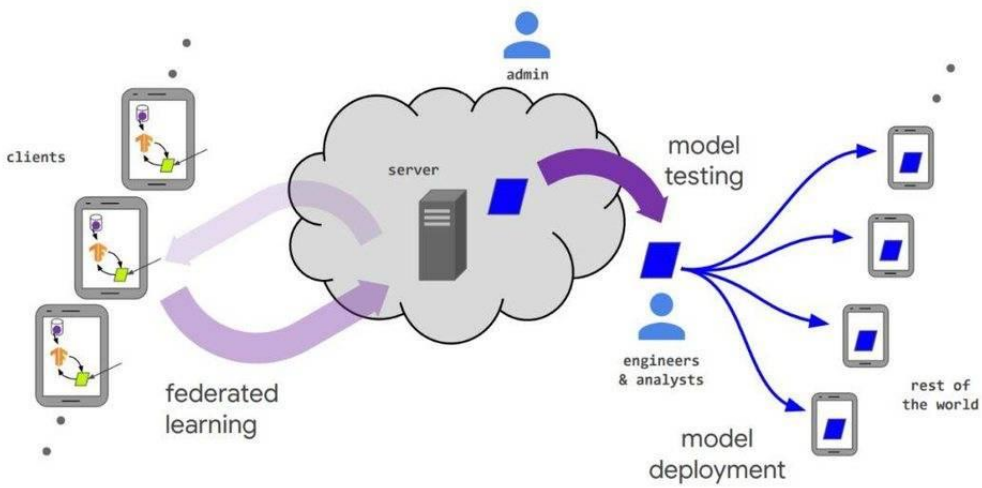

Figure 1. The Lifecycle of Federated Learning [3]

2.2 Key Algorithms

Among the primal algorithms that lay the foundation for production-level Federated Learning is **Federated Averaging** (FedAvg)[2]. It combines local training on client devices (through synchronous stochastic gradient descent (SGD)) with a central parameter server that incorporates local updates into the shared global model by averaging (as shown in *Figure 2*).

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

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

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

Figure 2. The Federated Averaging (FedAvg) Algorithm [2]

Another important algorithm is the **Secure Aggregation** protocol [5] which aims to enhance privacy and security guarantees by leveraging secure multi-party computation (MPC) when updating the shared global model by aggregating model updates) from local devices.

2.3 Key Frameworks and Applications

An open-source framework called **TensorFlow Federated** (**TFF**) [1] is developed by Google to facilitate open research in FL. Another FL infrastructure already used in production is **Federated AI Technology Enabler** (**FATE**)[2], which is also an open-source project, initiated and implemented by Webank's AI Department. It provides secure computation protocols based on MPC and homomorphic encryption in FL.

Making inferences such as next-word predictions on mobile keyboard like Gboard (as shown in *Figure 3*) is one of the main practical applications of federated learning. Researchers in [6] train a RNN language model using FegAvg which, through various experiments, is shown to achieve better results than traditional server-based training using SGD.

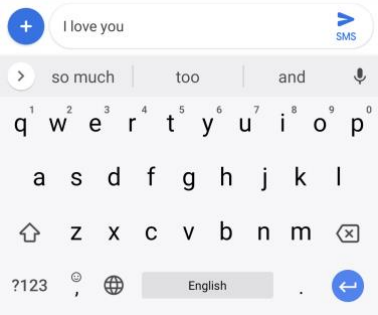

Figure 3. Next word predictions in Gboard [6]

Smart retail is also able to take full advantage of FL to provide customers with personalized services such as personal recommendations in sales and promotion. In this case, users might be reluctant to share their private information like purchase history and preferences with service providers [7].

Another practical domain to benefit from FL is smart healthcare. Medical data such as diagnosis and treatment records must be kept private. However, this isolation of information among medical centres and hospitals could lead to insufficiency of training data and thus an unsatisfactory performance of predictive models [7, 8].

[1] https://www.tensorflow.org/federated
[2] https://github.com/FederatedAI/FATE

Chapter 3. Major Challenges

3.1 Improving Communication-efficiency

It is often the case that mobile devices have limited and costly data plans, and network connections between these edge devices and the central server are unstable or slow. Therefore, for collaborative learning approaches like federated learning, communication cost has become a major system bottleneck. In this section, an overview of distributed learning techniques that address this problem is presented. These techniques can be roughly grouped into 3 categories:

1. reducing the number of communication rounds;
2. decreasing the size of updates communicated in each round;
3. knowledge distillation.

Within the first category, one type of method is **local updating** (also known as **communication delay**) by which the communication frequency is reduced. Under the assumption that data are distributed among different clients in an independently-and-identically-distributed (i.i.d.) way, using the original FedAvg algorithm, communications can be delayed for up to 100 SGD iterations without causing a significant delay in the convergence of the shared global model [9].

Another type of method that falls into the second category is **model compression**, including [10]:

- **compact models** (e.g. manipulating special correlations by stacking up two 3×3 convolution layers to form a receptive field (RF) comparable to one 5×5 convolution layer, with fewer CNN model parameters ([1 - (3*3*2) / (5*5)] * 100% = 28% less) and multiply-and-accumulate operations; In addition, Neural Architecture Search (NAS) has been developed to automate the search for optimal neural network architectures using machine learning techniques);
- **data quantization** and **network sparsification** (e.g. in [11], Bonawitz et al. follows FedAvg up with a combination of probabilistic quantization (e.g. the parameter N in N-bit quantization allows for a balance between accuracy and

communication costs) and random sparsification (random sparse weight updates – either structured updates or sketched updates). Although proven to be scalable across a varying number of clients (full or partial client participation), this method could slow down convergence significantly).

Some of these methods have such limitations that either they only compress uplinks (upstream communications from local clients to the central server), or that they make unrealistic assumptions that data are i.i.d. In [12], a novel compression framework called **sparse ternary compression** (**STC**) which is specially designed to suit the needs of FL settings is proposed. STC extends the existing top-N gradient sparsification with optimal Golomb encoding of weight updates to enable data compression and ternarization in downlinks (downstream connections from the central server to local clients). It has been demonstrated that STC achieves better results than FedAvg in various FL scenarios.

Given the non-IID characteristics of local datasets (see *Section 3.2 Combating Statistical Heterogeneity*), some local updates can be biased or outliers. Incorporating these updates into the global update could cause the global model to deviate from the correct direction of convergence. To tackle the issue, authors of [13] proposed **Communication-mitigated Federated Learning** (**CMFL**) to improve communication-efficiency while still guaranteeing model convergence.

The key idea of CMFL is to identify irrelevant local updates produced by private training on biased user data, and then exclude these updates from global aggregation. CMFL measures the relevance of local updates based on their conformity to the global update collected from all participating clients in the previous iteration, i.e. checking if they align with the tendency of global model updating. To be more specific, CMFL calculates the percentage of parameters in local updates that have signs different from their counterparts in the global update. Any update whose percentage (used as the relevance metric) is higher than a pre-defined threshold is rendered irrelevant or divergent.

Excluding these irrelevant updates from being uploaded not only has no adverse influence on model accuracy or convergence guarantees, but it also reduces communication overhead by avoiding transferring extraneous data.

Another type of method for boosting communication efficiency is known as **knowledge distillation**. Caruana et al. have shown that compressing knowledge in an ensemble of models into a single model is possible. G. Hinton et al. develop this method further by distilling the knowledge of a cumbersome teacher model and then transferring it to a much smaller distilled model by local training on a small transfer dataset using a soft target distribution (the layer of tensors/logits before applications of the Softmax activation function) [14]. Besides, by treating the global cumbersome model as the teacher model and the personalized local models as student models, the problem of overfitting (which has an obvious negative impact on model performance especially when the size of local datasets is small) during local training can be alleviated. The authors also introduce an ensemble of both the general models and the specialist models to help categorize specific classes that the general models might confuse.

Federated Distillation (**FD**) introduced in [9] is inspired by knowledge distillation. With FD, devices exchange with the server the average output logit vectors which only include one entry for each classification label. Using this distributed knowledge distillation method, the size of communication payload no longer depends on the size of the model but rather on the dimension of ground-truth labels. Although this approach generally reduces the communication cost to a large extent, the accuracy gain is likely to suffer due to the loss of information in updates. It is worth exploring whether FD and FL could be combined to strike a good balance between communication-efficiency and model accuracy. Besides, the performance of the global model obtained through FD may as well suffer from degradation in non-i.i.d. scenarios (which will be discussed in the next section).

3.2 Combating Statistical Heterogeneity

There are many cases in which it is preferable to have a global model, for example, to share the global model with clients who have no data stored locally. Unfortunately, in typical federated learning scenarios, data distribution varies greatly across client devices. For example, the amount of data that different end users hold might vary significantly, which is referred to as **quantity skew**.

In addition, for different users, **concept drift** is likely to happen, where the same input data (and thus same features) may not receive the same labels (e.g. due to personal preferences). To put it in a statistical perspective, the conditional distribution P(y|x) across clients may differ while P(x) remains the same.

These types of **statistical heterogeneity** (data not being non-i.i.d.) among participating users could severely degrade the performance of FedAvg.

We have mentioned federated distillation (FD) in the previous section and know its limitations in non-i.i.d. scenarios. To mitigate the negative impact of non-i.i.d. data on FD, researchers in [9] propose a data augmentation approach called **Federated Augmentation** (**FedAug**) where each client device is capable of generating data samples that it lacks locally using generative adversarial networks (GAN). Each device finds out its missing labels (referred to as the target labels), and uploads a few samples of these labels to the central server. Then the server searches for images over the Internet (e.g. through the Google search engine) for similar samples. Having obtained training data in this manner, a conditional GAN is trained at the server (with powerful processors and fast network connections). Next, client devices download the trained generator of GAN to complement the target labels in order to establish an i.i.d. dataset. However, operations in FedAug may expose client devices' bias during data generation (i.e., target/missing labels) and thus lead to privacy leakage.

Multiple **personalization** techniques have been put forward to mitigate statistical heterogeneity, including a hybrid neural network architecture called **FedPer** that

Arivazhagan et al. come up with in [15]. In FedPer, models are divided into two separate components: the base layers, and the personalized layers. Base layers are trained collaboratively using the original FedAvg, and personalization layers are trained privately using SGD. Personalization can also been leveraged in fully decentralized settings (independent learning).

Multi-task learning [16] can be relevant in combating statistical heterogeneity as well, in that it allows models to extract both common and distinct features across multiple tasks by learning collaboratively and solving related but different tasks simultaneously. Intuitively, it is a good choice for building personalized models.

Meta-learning is another useful approach herein. Finn et al. [17] bring forward the **Model-agnostic Meta-learning** (**MAML**) algorithm that involves training on multiple learning tasks to produce models so highly-adaptable that they can proceed to solve new tasks with only a small number of training examples. There are two connected phases in MAML, one being meta-training and the other being meta-testing. During the meta-training phase, a global model is generated by solving multiple tasks, and then during the meta-testing phase, the global model is adapted separately and accordingly to each individual task. When comparing vanilla FL to meta-training, and local personalization to meta-testing, it can be observed that FedAvg is similar to the popular Reptile algorithm in MAML.

Khodak et al. [18] propose the **Average Regret-Upper-Bound Analysis** (**ARUBA**) algorithm based on online convex optimization. An improvement in model performance is achieved by applying it to FedAvg, the main goal of which is to train models on heterogeneous user data to generate a global model without much upstream or downstream communication, with another auxiliary goal being model personalization. It has been shown that ARUBA performs better than non-tuned FedAvg and its performance is on a par with carefully fine-tuned FedAvg. This makes hyperparameter tuning less important, which in contrast could be essential in vanilla FL settings.

Recently, a group of Google researchers introduced an innovative approach called **Weight-agnostic Neural Networks** (**WANN**) in [19] to search for simple neural network architectures with strong inductive biases. Inspired by the Baldwin effect in evolutionary developmental biology (which describes trait or behaviour changes that occur in an organism as a result of its interaction with surrounding environments), neural networks architectures in this work evolve and increase their complexity by adding connections during the learning process, using a shared weight over a range of values. In this manner, networks are naturally capable of completing certain tasks using only randomly sampled weight parameters.

The tremendous success of CNNs during the last decade can be largely attributed to their novel building blocks: convolutional layers which have strong biases towards tasks in computer vision. Since WANN is still in its early stage, networks built using WANN do not match the performance of refined CNNs. However, this work can facilitate and automate the discovery of new architectures or building blocks that not only possess strong inductive biases for various practical domains, but also do not necessarily have to be trained with gradient-based algorithms.

There have been much past and ongoing work to provide **convergence guarantees** for FedAvg based on two assumptions: 1) data are i.i.d. and 2) all devices are active participants. The **FedProx** algorithm (as shown in ***Error! Reference source not found.***) designed by T. Li et al. [20] doesn't rely on these assumptions. FedProx generalizes and re-parametrizes FedAvg, and it has been validated theoretically, through numerous empirical evaluations, to have more robust convergence than FL in real-life heterogeneous networks across a range of practical datasets. In FedAvg and FedProx, each device uses its local objective function (instead of the global objective) based on its private data. The key to increase flexibility (in the amount of local computation and data transfer) in this case is to inexactly solve these local objectives, based on the number of local iterations conducted.

3.3 Dealing with Adversarial Attacks

Examples of adversarial attacks include (and are not limited to) **data poisoning** and **model updates poisoning**, which can be further categorized into training-time attacks and inference-time attacks. Compared to distributed datacentre learning and centralized learning, new attack surfaces may be present in FL setting at training-time. Model updates poisoning happens when the adversary directly tampers with local updates uploaded to the server. Data poisoning is relatively restrictive in that the adversary can only corrupt client data, e.g. by replacing correct labels with false ones or manipulating certain data features. One worthy line of study is quantitatively measuring the gap between these two types of attacks, which can be particularly important in FL due to differences in the capabilities of adversaries (the maximum number of attacks or the percentage of data corruption).

Data sanitization is a class of defence methods aiming to remove irregular/abnormal data from the training data. Since it requires access to user data which may not be viable in FL, how to utilise it in this case without much privacy loss remains an open question. Furthermore, many of these methods are vulnerable to adaptive adversaries, and detection of data poisoning alone (without identifying the source of it or recovering from it) can be very challenging. Such metrics as training accuracy of the shared global model or that of local models may become insufficient to detect the presence of backdoor attacks.

Similar to training with **differential privacy** (achieved by the addition of random noise of a chosen magnitude related to the largest value an individual local update could have), the system can defend itself against model update poisoning by bounding the contribution of any individual local model to the global model: 1) clipping local updates by enforcing a normalized constraint; 2) adding Gaussian noise to the aggregated updates. This approach prevents the global model from overfitting to any individual update or a set of malicious adversaries. Since data poisoning can be considered to be a failure during learning, differential privacy is also an intuitive way to defend against this type of attacks.

Differential privacy is strong in that it is a general approach that provides protection in worst possible cases (no matter what harm the adversary plans to cause), and it is weak in that the adversary must be confined, and at the same time random noise must be injected at learning-time.

Chapter 4. Conclusion and Future Trends

In this report, a brief description of the background and underlying technologies (including key algorithms and mainstream frameworks) of Federated Learning (FL) is provided. Three major challenges that FL is faced with are mentioned here, including 1) how to boost communication-efficiency without compromising much model accuracy or convergence speed; 2) how to deal with the situation where non-i.i.d. is a common feature of the overall data distribution in FL; and 3) how to handle adversarial attacks that surface in FL. Various approaches designed to counteract these challenges are covered in detail.

Pioneered by Google's first industrial-level open-source platform (TFF), FL is revolutionizing the future of AI in such a way that end users can train machine learning models locally, without trading their private data for better services, or relying on the computing facilities owned by tech giants. Increased availability of geographically distributed datasets has a huge potential to help improve the diversity of training data available for machine learning models, which as a result can facilitate better modelling practice. With an ever-increasing demand for secure and distributed on-device learning, FL will surely inspire and spawn many more novel techniques and applications.

REFERENCES

[1] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A Survey on Distributed Machine Learning," *ACM Comput. Surv.*, vol. 53, no. 2, p. Article 30, 2020.

[2] H. M. B. McMahan, E. Ramage, D. Hampson and etc., "Communication-Efficient Learning of Deep Networks from Decentralized Data," *arXiv:1602.05629*.

[3] P. Kairouz *et al.*, *Advances and Open Problems in Federated Learning*. 2019.

[4] H. E. K. Bonawitz, J. Konečný, B. McMahan, J. Roselander and etc., "Towards Federated Learning at Scale: System Design," in *Systems and Machine Learning*, 2019.

[5] V. I. K. Bonawitz, B. Kreuter, A. Marcedone, H. Brendan McMahan, and etc., "Practical Secure Aggregation for Privacy-Preserving Machine Learning," presented at the Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, Texas, USA, 2017.

[6] A. Hard *et al.*, *Federated Learning for Mobile Keyboard Prediction*. 2018.

[7] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated Machine Learning: Concept and Applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, pp. 1-19, 01/28 2019.

[8] N. Tobis, "Federated Machine Learning: A Distributed Approach to Pain Expression Recognition in Healthcare," MSc, Department of Computing, Imperial College London, 2019.

[9] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, *Communication-Efficient On-Device Machine Learning: Federated Distillation and Augmentation under Non-IID Private Data*. 2018.

[10] G. L. B. L. Deng, S. Han, L. Shi and Y. Xie, "Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485-532, 2020.

[11] H. B. M. Jakub Konečný, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, Dave Bacon, "Federated Learning: Strategies for Improving Communication Efficiency," presented at the NIPS Workshop on Private Multi-Party Machine Learning, 2016.

[12] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[13] W. W. L. WANG, B. LI, "CMFL: Mitigating Communication Overhead for Federated Learning," in *IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, Dallas, TX, USA, 2019, pp. 954-964.

[14] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," 03/09 2015.

[15] M. Arivazhagan, V. Aggarwal, A. Singh, and S. Choudhary, *Federated Learning with Personalization Layers*. 2019.

[16] C. C. V. Smith, M. Sanjabi,A. Talwalkar, "Federated Multi-Task Learning," in *NIPS*, Long Beach, 2017.

[17] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," presented at the Proceedings of the 34th

International Conference on Machine Learning - Volume 70, Sydney, NSW, Australia, 2017.

[18] M. B. M. Khodak, A. Talwalkar, "Adaptive Gradient-Based Meta-Learning Methods," presented at the NeurIPS, 2019.

[19] A. Gaier and D. Ha, *Weight Agnostic Neural Networks*. 2019.

[20] A. K. T. Li, M. Zaheer and etc., "Federated Optimization in Heterogeneous Networks," in *Proceedings of the 3rd MLSys Conference*, Austin, TX, USA, 2020.