Clustered Federated Learning: Model-Agnostic Distributed Multi-Task Optimization under Privacy Constraints

Felix Sattler, Klaus-Robert Müller*, *Member, IEEE*, and Wojciech Samek*, *Member, IEEE*

***Abstract*—Federated Learning (FL) is currently the most widely adopted framework for collaborative training of (deep) machine learning models under privacy constraints. Albeit it's popularity, it has been observed that Federated Learning yields suboptimal results if the local clients' data distributions diverge. To address this issue, we present Clustered Federated Learning (CFL), a novel Federated Multi-Task Learning (FMTL) framework, which exploits geometric properties of the FL loss surface, to group the client population into clusters with jointly trainable data distributions. In contrast to existing FMTL approaches, CFL does not require any modifications to the FL communication protocol to be made, is applicable to general non-convex objectives (in particular deep neural networks) and comes with strong mathematical guarantees on the clustering quality. CFL is flexible enough to handle client populations that vary over time and can be implemented in a privacy preserving way. As clustering is only performed after Federated Learning has converged to a stationary point, CFL can be viewed as a post-processing method that will always achieve greater or equal performance than conventional FL by allowing clients to arrive at more specialized models. We verify our theoretical analysis in experiments with deep convolutional and recurrent neural networks on commonly used Federated Learning datasets.**

I. Introduction

Federated Learning [3][4][5][6][7] is a distributed training framework, which allows multiple clients (typically mobile or IoT devices) to jointly train a single deep learning model on their combined data in a communication-efficient way, without requiring any of the participants to reveal their private training data to a centralized entity or to each other. Federated Learning realizes this goal via an iterative three-step protocol where in every communication round t, the clients first synchronize with the server by downloading the latest master model θ_t. Every client then proceeds to improve the downloaded model, by performing multiple iterations of stochastic gradient descent with mini-batches sampled from it's local data D_i, resulting in a weight-update vector

$$\Delta\theta_i^{t+1} = \text{SGD}_k(\theta^t, D_i) - \theta^t,\ i = 1, .., m \tag{1}$$

Finally, all clients upload their computed weight-updates to the server, where they are aggregated by weighted averaging according to

$$\theta^{t+1} = \theta^t + \sum_{i=1}^{m} \frac{|D_i|}{|D|} \Delta\theta_i^{t+1} \tag{2}$$

to create the next master model. The procedure is summarized in Algorithm 2.

Federated Learning implicitly makes the assumption that it is possible for one single model to fit all client's data generating distributions φ_i at the same time. Given a model $f_\theta : \mathcal{X} \to \mathcal{Y}$ parametrized by $\theta \in \Theta$ and a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ we can formally state this assumption as follows:

Assumption 1. ***("Federated Learning"):*** *There exists a parameter configuration $\theta^* \in \Theta$, that (locally) minimizes the risk on all clients' data generating distributions at the same time:*

$$R_i(\theta^*) \leq R_i(\theta)\ \forall \theta \in B_\varepsilon(\theta^*), i = 1, .., m \tag{3}$$

for some $\varepsilon > 0$. Hereby

$$R_i(\theta) = \int l(f_\theta(x), y) d\varphi_i(x, y) \tag{4}$$

is the risk function associated with distribution φ_i.

It is easy to see that this assumption is not always satisfied. Concretely it is violated if either (a) clients have disagreeing conditional distributions $\varphi_i(y|x) \neq \varphi_j(y|x)$ or (b) the model f_θ is not expressive enough to fit all distributions at the same time. Simple counter examples for both cases are presented in Figure 1.

In the following we will call a set of clients and their data generating distributions φ *congruent* (with respect to f and l) if they satisfy Assumption 1 and *incongruent* if they don't.

In this work, we argue that Assumption 1 is frequently violated in real Federated Learning applications, especially given the fact that in Federated Learning clients (a) can hold arbitrary non-iid data, which can not be audited by the centralized server due to privacy constraints and (b) typically run on limited hardware which puts restrictions on the model

This work was supported by the German Ministry for Education and Research as Berlin Big Data Center (01IS14013A) and the Berlin Center for Machine Learning (01IS18037I). Partial funding by DFG is acknowledged (EXC 2046/1, project-ID: 390685689). This work was also supported by the Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (No. 2017-0-00451).

F. Sattler and W. Samek are with Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany (e-mail: wojciech.samek@hhi.fraunhofer.de).

K.-R. Müller is with the Technische Universität Berlin, 10587 Berlin, Germany, with the Max Planck Institute for Informatics, 66123 Saarbrücken, Germany, and also with the Department of Brain and Cognitive Engineering, Korea University, Seoul 136-713, South Korea (e-mail: klaus-robert.mueller@tu-berlin.de).

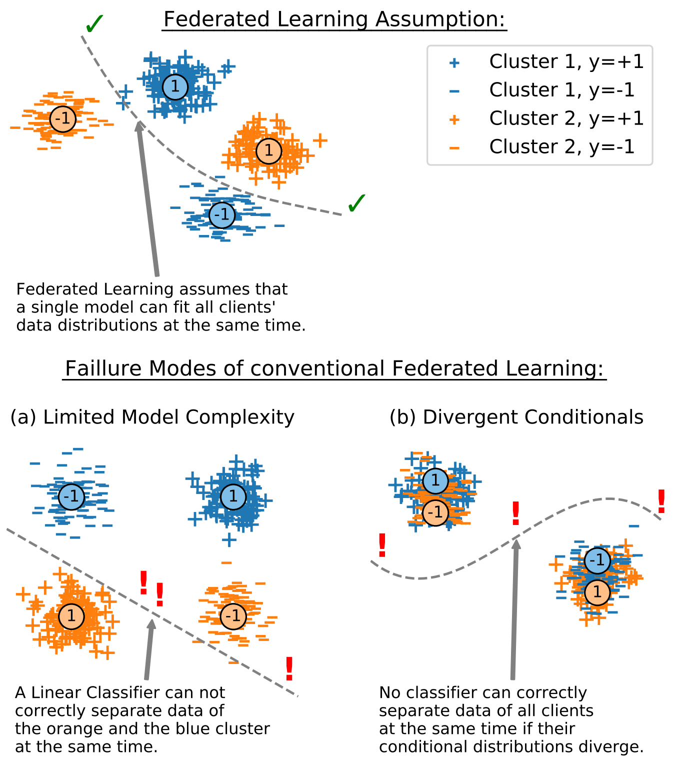
Federated Learning Assumption:
Cluster 1, y=+1
Cluster 1, y=-1
Cluster 2, y=+1
Cluster 2, y=-1
Federated Learning assumes that a single model can fit all clients' data distributions at the same time.
Faillure Modes of conventional Federated Learning:
(a) Limited Model Complexity
(b) Divergent Conditionals
A Linear Classifier can not correctly separate data of the orange and the blue cluster at the same time.
No classifier can correctly separate data of all clients at the same time if their conditional distributions diverge.

Fig. 1: Two toy cases in which the Federated Learning Assumption is violated. Blue points belong to clients from the first cluster while orange points belong to clients from the second cluster. Left: Federated XOR-problem. An insufficiently complex model is not capable of fitting all clients' data distributions at the same time. Right: If different clients' conditional distributions diverge, no single model can fit all distributions at the same time. In both cases the data on clients belonging to the same cluster can be easily separated.

complexity. For illustration consider the following practical scenarios:

Varying Preferences: Assume a scenario where every client holds a local dataset of images of human faces and the goal is to train an 'attractiveness' classifier on the joint data of all clients. Naturally, different clients will have varying opinions about the attractiveness of certain individuals, which corresponds to disagreeing conditional distributions on all clients' data. Assume for instance that one half of the client population thinks that people wearing glasses are attractive, while the other half thinks that those people are unattractive. In this situation one single model will *never* be able to accurately predict attractiveness of glasses-wearing people for all clients at the same time (confer also Figure 1 right).

Limited Model Complexity: Assume a number of clients are trying to jointly train a language model for next-word prediction on private text messages. In this scenario the statistics of a client's text messages will likely vary a lot based on demographic factors, interests, etc. For instance, text messages composed by teenagers will typically exhibit different statistics than than those composed by elderly people. An insufficiently expressive model will not be able to fit the data of all clients at the same time.

Presence of Adversaries: A special case of incongruence is given, if a subset of the client population behaves in an adversarial manner. In this scenario the adversaries could deliberately alter their local data distribution in order to encode arbitrary behavior into the jointly trained model, thus affecting the model decisions on all other clients and causing potential harm.

The goal in Federated Multi-Task Learning is to provide every client with a model that optimally fits it's local data distribution. In all of the above described situations the ordinary Federated Learning framework, in which all clients are treated equally and only one single global model is learned, is not capable of achieving this goal.

In order to incorporate the above presented problems with incongruent data generating distributions, we suggest to generalize the conventional Federated Learning Assumption:

Assumption 2. ***("Clustered Federated Learning"):*** *There exists a partitioning $\mathcal{C} = \{c_1, .., c_k\}$, $\bigcup_{i=1}^{k} c_k = \{1, .., m\}$ of the client population, such that every subset of clients $c \in \mathcal{C}$ satisfies the conventional Federated Learning Assumption.*

The remainder of this manuscript is organized as follows: In the next section (II) we will derive a computationally efficient tool based on the cosine similarity between the clients' gradient updates that *provably* allows us to infer whether two members of the client population have the same data generating distribution, thus making it possible for us to infer the clustering structure $\mathcal{C}$. Based on the theoretical insights in section II we present the Clustered Federated Learning Algorithm in section III. After reviewing related literature in section IV, we address implementation details in section V and demonstrate that our novel method can be implemented without making modifications to the Federated Learning communication protocol (section V-A). We furthermore show that our method can be implemented in a privacy preserving way (section V-B) and is flexible enough to handle client populations that vary over time (section V-C). Finally, in section VI, we perform extensive experiments on a variety of convolutional and recurrent neural networks applied to common Federated Learning datasets.

II. Cosine Similarity based Clustering

In this paper, we address the question of how to solve distributed learning problems that satisfy Assumption 2 (which generalizes the Federated Learning Assumption 1). This will require us to first identify the correct partitioning $\mathcal{C}$, which at first glance seems like a daunting task, as under the Federated Learning paradigm the server has no access to the clients' data, their data generating distributions or any meta information thereof.

An easier task than trying to directly infer the entire clustering structure $\mathcal{C}$, is to find a correct bi-partitioning in the sense of the following definition:

Definition 1. Let $m \geq k \geq 2$ and

$$I : \{1, .., m\} \rightarrow \{1, .., k\}, i \mapsto I(i) \tag{5}$$

be the mapping that assigns a client i to it's data generating distribution $\varphi_{I(i)}$. Then we call a bi-partitioning $c_1 \dot\cup c_2 = \{1, .., m\}$ with $c_1 \neq \emptyset$ and $c_2 \neq \emptyset$ *correct* if and only if

$$I(i) \neq I(j) \quad \forall i \in c_1, j \in c_2. \tag{6}$$

In other words, a bi-partitioning is called *correct*, if clients with the same data generating distribution end up in the same cluster. It is easy to see, that the clustering $\mathcal{C} = \{c_1, .., c_k\}$ can be obtained after exactly $k - 1$ correct bi-partitions.

In the following we will demonstrate that there exists an explicit criterion based on which a correct bi-partitioning can be inferred. To see this, let us first look at the following simplified Federated Learning setting with m clients, in which the data on every client was sampled from one of two data generating distributions φ_1, φ_2 such that

$$D_i \sim \varphi_{I(i)}(x, y). \tag{7}$$

Every Client is associated with an empirical risk function

$$r_i(\theta) = \sum_{x \in D_i} l_\theta(f(x_i), y_i) \tag{8}$$

which approximates the true risk arbitrarily well if the number of data points on every client is sufficiently large

$$r_i(\theta) \approx R_{I(i)}(\theta) := \int_{x,y} l_\theta(f(x), y) d\varphi_{I(i)}(x, y). \tag{9}$$

For demonstration purposes let us first assume equality in (9). Then the Federated Learning objective becomes

$$F(\theta) := \sum_{i=1}^{m} \frac{|D_i|}{|D|} r_i(\theta) = a_1 R_1(\theta) + a_2 R_2(\theta) \tag{10}$$

with $a_1 = \sum_{i, I(i)=1} |D_i|/|D|$ and $a_2 = \sum_{i, I(i)=2} |D_i|/|D|$. Under standard assumptions it has been shown [8][9] that the Federated Learning optimization protocol described in equations (1) and (2) converges to a stationary point θ^* of the Federated Learning objective. In this point it holds that

$$0 = \nabla F(\theta^*) = a_1 \nabla R_1(\theta^*) + a_2 \nabla R_2(\theta^*) \tag{11}$$

Now we are in one of two situations. Either it holds that $\nabla R_1(\theta^*) = \nabla R_2(\theta^*) = 0$, in which case we have simultaneously minimized the risk of all clients. This means φ_1 and φ_2 are congruent and we have solved the distributed learning problem. Or, otherwise, it has to hold

$$\nabla R_1(\theta^*) = -\frac{a_2}{a_1} \nabla R_2(\theta^*) \neq 0 \tag{12}$$

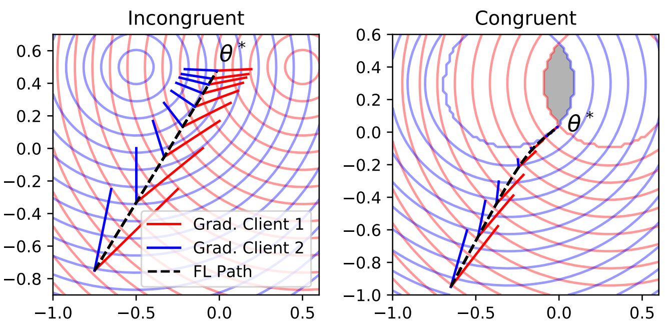

Fig. 2: Displayed are the optimization paths of Federated Learning with two clients, applied to two different toy problems with incongruent (left) and congruent (right) risk functions. In the incongruent case Federated Learning converges to a stationary point of the FL objective where the gradients of the two clients are of positive norm and point into opposite directions. In the congruent case there exists an area (marked grey in the plot) where both risk functions are minimized. If Federated Learning converges to this area the norm of both client's gradient updates goes to zero. By inspecting the gradient norms the two cases can be distinguished.

and φ_1 and φ_2 are incongruent. In this situation the *cosine similarity* between the gradient updates of any two clients is given by

$$\begin{aligned}\alpha_{i,j} := \alpha(\nabla r_i(\theta^*), \nabla r_j(\theta^*)) &:= \frac{\langle \nabla r_i(\theta^*), \nabla r_j(\theta^*) \rangle}{\|\nabla r_i(\theta^*)\| \|\nabla r_j(\theta^*)\|} \\ &= \frac{\langle \nabla R_{I(i)}(\theta^*), \nabla R_{I(j)}(\theta^*) \rangle}{\|\nabla R_{I(i)}(\theta^*)\| \|\nabla R_{I(j)}(\theta^*)\|} \\ &= \begin{cases} 1 & \text{if } I(i) = I(j) \\ -1 & \text{if } I(i) \neq I(j) \end{cases}\end{aligned} \tag{13}$$

Consequently, a correct bi-partitioning is given by

$$c_1 = \{i | \alpha_{i,0} = 1\}, \; c_2 = \{i | \alpha_{i,0} = -1\}. \tag{14}$$

This consideration provides us with the insight that, in a stationary solution of the Federated Learning objective θ^*, we can *distinguish clients based on their hidden data generating distribution by inspecting the cosine similarity between their gradient updates*. For a visual illustration of the result we refer to Figure 2.

If we drop the equality assumption in (9) and allow for an arbitrary number of data generating distributions, we obtain the following generalized version of result (13):

Theorem 1 (Separation Theorem). *Let $D_1, .., D_m$ be the local training data of m different clients, each dataset sampled from one of k different data generating distributions $\varphi_1, .., \varphi_k$, such that $D_i \sim \varphi_{I(i)}(x, y)$. Let the empirical risk on every client approximate the true risk at every stationary solution of the*

Federated Learning objective θ^* s.t.

$$\|\nabla R_{I(i)}(\theta^*)\| > \|\nabla R_{I(i)}(\theta^*) - \nabla r_i(\theta^*)\| \tag{15}$$

and define

$$\gamma_i := \frac{\|\nabla R_{I(i)}(\theta^*) - \nabla r_i(\theta^*)\|}{\|\nabla R_{I(i)}(\theta^*)\|} \in [0,1) \tag{16}$$

Then there exists a bi-partitioning $c_1^* \cup c_2^* = \{1,..,m\}$ of the client population such that that the maximum similarity between the updates from any two clients from different clusters can be bounded from above according to

$$\alpha_{cross}^{max} := \min_{c_1 \cup c_2 = \{1,..,m\}} \max_{i\in c_1, j\in c_2} \alpha(\nabla r_i(\theta^*), \nabla r_j(\theta^*)) \tag{17}$$

$$= \max_{i\in c_1^*, j\in c_2^*} \alpha(\nabla r_i(\theta^*), \nabla r_j(\theta^*)) \tag{18}$$

$$\leq \begin{cases} \cos(\frac{\pi}{k-1})H_{i,j} + \sin(\frac{\pi}{k-1})\sqrt{1-H_{i,j}^2} & \textit{if } H \geq \cos(\frac{\pi}{k-1}) \\ 1 & \textit{else} \end{cases} \tag{19}$$

with

$$H_{i,j} = -\gamma_i\gamma_j + \sqrt{1-\gamma_i^2}\sqrt{1-\gamma_j^2} \in (-1,1]. \tag{20}$$

At the same time the similarity between updates from clients which share the same data generating distribution can be bounded from below by

$$\alpha_{intra}^{min} := \min_{\substack{i,j \\ I(i)=I(j)}} \alpha(\nabla_\theta r_i(\theta^*), \nabla_\theta r_j(\theta^*)) \geq \min_{\substack{i,j \\ I(i)=I(j)}} H_{i,j}. \tag{21}$$

The proof of Theorem 1 can be found in the Appendix.

Remark 1. *In the case with two data generating distributions ($k = 2$) the result simplifies to*

$$\alpha_{cross}^{max} = \max_{i\in c_1^*, j\in c_2^*} \alpha(\nabla_\theta r_i(\theta^*), \nabla_\theta r_j(\theta^*)) \leq \max_{i\in c_1^*, j\in c_2^*} -H_{i,j} \tag{22}$$

for a certain partitioning, respective

$$\alpha_{intra}^{min} = \min_{\substack{i,j \\ I(i)=I(j)}} \alpha(\nabla_\theta r_i(\theta^*), \nabla_\theta r_j(\theta^*)) \geq \min_{\substack{i,j \\ I(i)=I(j)}} H_{i,j} \tag{23}$$

for two clients from the same cluster. If additionally $\gamma_i = 0$ for all $i = 1,..,m$ then $H_{i,j} = 1$ and we retain result (13).

From Theorem 1 we can directly deduce a correct separation rule:

Corollary 1. *If in Theorem 1 k and γ_i, $i = 1,..,m$ are in such a way that*

$$\alpha_{intra}^{min} > \alpha_{cross}^{max} \tag{24}$$

then the partitioning

$$c_1, c_2 \leftarrow \arg\min_{c_1\cup c_2 = c} (\max_{i\in c_1, j\in c_2} \alpha_{i,j}). \tag{25}$$

is always *correct in the sense of Definition 1.*

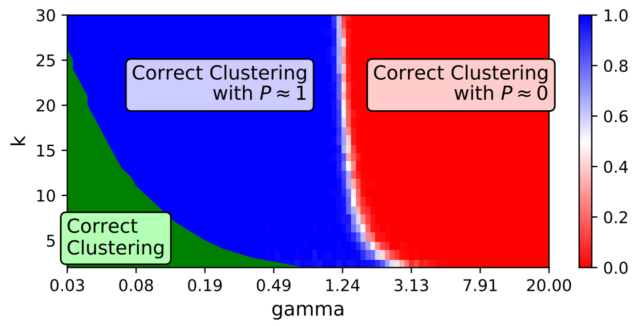

Fig. 3: Clustering quality as a function of the number of data generating distributions k and the relative approximation noise γ. For all values of k and γ in the green area, CFL will *always* correctly separate the clients (by Theorem 1). For all values of k and γ in the blue area we find empirically that CFL will correctly separate the clients with probability close to 1.

Proof: Set

$$c_1, c_2 \leftarrow \arg\min_{c_1\cup c_2 = c} (\max_{i\in c_1, j\in c_2} \alpha_{i,j}) \tag{26}$$

and let $i \in c_1$, $j \in c_2$ then

$$\alpha_{i,j} \leq \alpha_{cross}^{max} < \alpha_{intra}^{min} = \min_{\substack{i,j \\ I(i)=I(j)}} \alpha_{i,j} \tag{27}$$

and hence i and j can not have the same data generating distribution. ■

This consideration leads us to the notion of the separation gap:

Definition 2 (Separation Gap)**.** Given a cosine-similarity matrix α and a mapping from client to data generating distribution I we define the notion of a separation gap

$$g(\alpha) := \alpha_{intra}^{min} - \alpha_{cross}^{max} \tag{28}$$

$$= \min_{\substack{i,j \\ I(i)=I(j)}} \alpha_{i,j} - \min_{c_1\cup c_2 = c} (\max_{i\in c_1, j\in c_2} \alpha_{i,j}) \tag{29}$$

Remark 2. *By Corollary 1 the bi-partitioning* (25) *will be correct in the sense of Definition 1 if and only if the separation gap is greater than zero.*

Theorem 1 gives an estimate for the similarities in the *absolute worst-case*. In practice α_{intra}^{min} typically will be much larger and α_{cross}^{max} typically will be much smaller, especially if the parameter dimension d is large. For instance, if we set $d = 10^2$ (which is still many orders of magnitude smaller than typical modern neural networks), $m = 3k$, and assume $\nabla R_{I(i)}(\theta^*)$ and $\nabla R_{I(i)}(\theta^*) - \nabla r_i(\theta^*)$ to be normally distributed for all $i = 1,..,m$ then experimentally we find (Figure 3) that

$$P[\text{"Correct Clustering"}] = P[g(\alpha) > 0] \approx 1 \tag{30}$$

even for large values of $k > 10$ and

$$\gamma_{max} := \max_{i=1,..,m} \gamma_i > 1. \tag{31}$$

This means that using the cosine similarity criterion (25) we can readily find a correct bi-partitioning c_1, c_2 even if the number of data generating distributions is high and the empirical risk on every client's data is only a very loose approximation of the true risk.

A. Distinguishing Congruent and Incongruent Clients

In order to appropriately generalize the classical Federated Learning setting, we need to make sure to only split up clients with *incongruent* data distributions. In the classical congruent non-iid Federated Learning setting described in [3] where one single model can be learned, performance will typically degrade if clients with varying distributions are separated into different clusters due to the restricted knowledge transfer between clients in different clusters. Luckily we have a criterion at hand to distinguish the two cases. To realize this we have inspect the gradients computed by the clients at a stationary point θ^*. When client distributions are incongruent, the stationary solution of the Federated Learning objective by definition can not be stationary for the individual clients. Hence the norm of the clients' gradients has to be strictly greater than zero. If conversely the client distributions are congruent, Federated optimization will be able to jointly optimize all clients' local risk functions and hence the norm of the clients' gradients will tend towards zero as we are approaching the stationary point. Based on this observation we can formulate the following criteria which allow us make the decision whether to split or not: Splitting should only take place if it holds that both (a) we are close to a stationary point of the FL objective

$$0 \leq \| \sum_{i=1,..,m} \frac{D_i}{|D_c|} \nabla_\theta r_i(\theta^*) \| < \varepsilon_1 \tag{32}$$

and (b) the individual clients are far from a stationary point of their local empirical risk

$$\max_{i=1,..,m} \|\nabla_\theta r_i(\theta^*)\| > \varepsilon_2 > 0 \tag{33}$$

Figure 2 gives a visual illustration of this idea for a simple two dimensional problem. We will also experimentally verify the clustering criteria (32) and (33) in section VI-B.

In practice we have another viable option to distinguish the congruent from the incongruent case. As splitting will only be performed after Federated Learning has converged to a stationary point, we always have computed the conventional Federated Learning solution as part of Clustered Federated Learning. This means that if after splitting up the clients a degradation in model performance is observed, it is always possible to fall back to the Federated Learning solution. In this sense Clustered Federated Learning will always improve the Federated Learning performance (or perform equally well at worst).

III. CLUSTERED FEDERATED LEARNING

Clustered Federated Learning recursively bi-partitions the client population in a top-down way: Starting from an initial set of clients $c = \{1, .., m\}$ and a parameter initialization θ_0, CFL performs Federated Learning according to Algorithm 2, in order to obtain a stationary solution θ^* of the FL objective. After Federated Learning has converged, the stopping criterion

$$0 \leq \max_{i \in c} \|\nabla_\theta r_i(\theta^*)\| < \varepsilon_2 \tag{34}$$

is evaluated. If criterion (34) is satisfied, we know that all clients are sufficiently close to a stationary solution of their local risk and consequently CFL terminates, returning the FL solution θ^*. If on the other hand, criterion (34) is violated, this means that the clients are incongruent and the server computes the pairwise cosine similarities α between the clients' latest transmitted updates according to equation (13). Next, the server separates the clients into two clusters in such a way that the maximum similarity between clients from different clusters is minimized

$$c_1, c_2 \leftarrow \arg \min_{c_1 \cup c_2 = c} \big(\max_{i \in c_1, j \in c_2} \alpha_{i,j} \big). \tag{35}$$

This optimal bi-partitioning problem at the core of CFL can be solved in $\mathcal{O}(m^3)$ using Algorithm 1. Since in Federated Learning it is assumed that the server has far greater computational power than the clients, the overhead of clustering will typically be negligible.

As derived in section II, a correct bi-partitioning can always be ensured if it holds that

$$\alpha_{intra}^{min} > \alpha_{cross}^{max}. \tag{36}$$

While the optimal cross-cluster similarity α_{cross}^{max} can be easily computed in practice, computation of the intra cluster similarity requires knowledge of the clustering structure and hence α_{intra}^{min} can only be estimated using Theorem 1 according to

$$\begin{aligned} \alpha_{intra}^{min} &\geq \min_{\substack{i,j \\ I(i)=I(j)}} -\gamma_i\gamma_j + \sqrt{1-\gamma_i^2}\sqrt{1-\gamma_j^2} \qquad (37) \\ &\geq 1 - 2\gamma_{max}^2. \qquad (38) \end{aligned}$$

Consequently we know that the bi-partitioning will be correct if

$$\gamma_{max} < \sqrt{\frac{1-\alpha_{cross}^{max}}{2}}. \tag{39}$$

independent of the number of data generating distributions k! This criterion allows us to reject bi-partitionings, based on our assumptions on the approximation noise γ_{max} (which is an interpretable hyperparameter).

If criterion (39) is satisfied, CFL is recursively re-applied to each of the two separate groups starting from the stationary solution θ^*. Splitting recursively continues on until (after at most $k-1$ recursions) none of the sub-clusters violate the stopping criterion anymore, at which point all groups of mutually congruent clients $\mathcal{C} = \{c_1, .., c_k\}$ have been identified, and the Clustered Federated Learning problem characterized by Assumption 2 is solved. The entire recursive procedure is presented in Algorithm 3.

Algorithm 1: Optimal Bipartition

```
1  input: Similarity Matrix $\alpha \in [-1,1]^{m,m}$
2  outout: bi-partitioning $c_1, c_2$ satisfying (25)
3  • $s \leftarrow$ argsort$(-\alpha[:]) \in \mathbb{N}^{m^2}$
4  • $\mathcal{C} \leftarrow \{\{i\} | i = 1,..,m\}$
5  for $i = 1,..,m^2$ do
6      • $i_1 \leftarrow s_i$ div $m$; $i_2 \leftarrow s_i$ mod $m$
7      • $c_{tmp} \leftarrow \{\}$
8      for $c \in \mathcal{C}$ do
9          if $i_1 \in c$ or $i_2 \in c$ then
10             • $c_{tmp} \leftarrow c_{tmp} \cup c$
11             • $\mathcal{C} \leftarrow \mathcal{C} \setminus c$
12         end
13     end
14     • $\mathcal{C} \leftarrow \mathcal{C} \cup \{c_{tmp}\}$
15     if $|\mathcal{C}| = 2$ then
16         return $\mathcal{C}$
17     end
18 end
```

Algorithm 2: Federated Learning (FL)

```
1  Input: initial parameters $\theta$, set of clients $c$, $\varepsilon_1 > 0$
2  repeat
3      for $i \in c$ in parallel do
4          • $\theta_i \leftarrow \theta$
5          • $\Delta\theta_i \leftarrow \text{SGD}_n(\theta_i, D_i) - \theta_i$
6      end
7      • $\theta \leftarrow \theta + \sum_{i \in c} \frac{|D_i|}{|D_c|} \Delta\theta_i$
8  until $\| \sum_{i \in c} \frac{|D_i|}{|D_c|} \Delta\theta_i \| < \varepsilon_1$
9  return $\theta$
```

Algorithm 3: Clustered Federated Learning (CFL)

```
1  Input: initial parameters $\theta$, set of clients $c$, $\gamma_{max} \in [0,1]$,
       $\varepsilon_2 > 0$
2  • $\theta^* \leftarrow$ FederatedLearning$(\theta, c)$
3  • $\alpha_{i,j} \leftarrow \frac{\langle \nabla r_i(\theta^*), \nabla r_j(\theta^*) \rangle}{\|\nabla r_i(\theta^*)\| \|\nabla r_j(\theta^*)\|}, i, j \in c$
4  • $c_1, c_2 \leftarrow \arg\min_{c_1 \cup c_2 = c} (\max_{i \in c_1, j \in c_2} \alpha_{i,j})$
5  • $\alpha_{cross}^{max} \leftarrow \max_{i \in c_1, j \in c_2} \alpha_{i,j}$
6  if $\max_{i \in c} \|\nabla r_i(\theta^*)\| \geq \varepsilon_2$ and $\sqrt{\frac{1-\alpha_{cross}^{max}}{2}} > \gamma_{max}$ then
7      • $\theta_i^*, i \in c_1 \leftarrow$ ClusteredFederatedLearning$(\theta^*, c_1)$
8      • $\theta_i^*, i \in c_2 \leftarrow$ ClusteredFederatedLearning$(\theta^*, c_2)$
9  else
10     • $\theta_i^* \leftarrow \theta^*$, $i \in c$
11 end
12 return $\theta_i^*, i \in c$
```

IV. Related Work

Federated Learning [3][4][10][7][11][6] is currently the dominant framework for distributed training of machine learning models under communication- and privacy constraints. Federated Learning assumes the clients to be congruent, i.e. that one central model can fit all client's distributions at the same time. Different authors have investigated the convergence properties of Federated Learning in congruent iid and non-iid scenarios: [12],[13],[14] and [15] perform an empirical investigation, [8], [16], [17] and [9] prove convergence guarantees. As argued in section I conventional Federated Learning is not able to deal with the challenges of incongruent data distributions. Other distributed training frameworks [18][19][20][21] are facing the same issues.

The natural framework for dealing with incongruent data is Multi-Task Learning [22][23][24]. An overview over recent techniques for multi-task learning in deep neural networks can be found in [25]. However all of these techniques are applied in a centralized setting in which all data resides at one location and the server has full control over and knowledge about the optimization process. Smith et al. [1] present MOCHA, which extends the multi-task learning approach to the Federated Learning setting, by explicitly modeling client similarity via a correlation matrix. However their method relies on alternating bi-convex optimization and is thus only applicable to convex objective functions and limited in it's ability to scale to massive client populations. Corinzia et al. [26] model the connectivity structure between clients and server as a Bayesian network and perform variational inference during learning. Although their method can handle non-convex models, it is expensive to generalize to large federated networks as the client models are refined sequentially.

Finally, Ghosh et al. [2] propose a clustering approach, similar to the one presented in this paper. However their method differs from ours in the following key aspects: Most significantly they use $l2$-distance instead of cosine similarity to determine the distribution similarity of the clients. This approach has the severe limitation that it only works if the client's risk functions are convex and the minima of different clusters are well separated. The $l2$-distance also is not able to distinguish congruent from incongruent settings. This means that the method will incorrectly split up clients in the conventional congruent non-iid setting described in [3]. Furthermore, their approach is not adaptive in the sense that the decision whether to cluster or not has to be made after the first communication round. In contrast, our method can be applied to arbitrary Federated Learning problems with non-convex objective functions. We also note that we have provided theoretical considerations that allow a systematic understanding of the novel CFL framework.

V. Implementation Considerations

A. Weight-Updates as generalized Gradients

Theorem 1 makes a statement about the cosine similarity between *gradients* of the empirical risk function. In Federated Learning however, due to constraints on both the memory of the client devices and their communication budged, instead commonly weight-updates as defined in (1) are computed and communicated. In order to deviate as little as possible from the classical Federated Learning algorithm it would hence be desirable to generalize result 1 to weight-updates. It is commonly conjectured (see e.g. [27]) that accumulated mini-batch gradients approximate the full-batch gradient of the objective function. Indeed, for a sufficiently smooth loss function and low learning rate, a weight update computed over one epoch approximates the direction of the true gradient since by Taylor approximation we have

$$\nabla_\theta r(\theta_\tau + \eta\Delta\theta_{\tau-1}, D_\tau) \tag{40}$$
$$= \nabla_\theta r(\theta_\tau, D_\tau) + \eta\Delta\theta_{\tau-1}\nabla^2_\theta r(\theta_\tau, D_\tau) + \mathcal{O}(\|\eta\Delta\theta_{\tau-1}\|^2) \tag{41}$$
$$= \nabla_\theta r(\theta_\tau, D_\tau) + R \tag{42}$$

where R can be bounded in norm. Hence, by recursive application of the above result it follows

$$\Delta\theta = \sum_{\tau=1}^{T} \nabla_\theta r(\theta_\tau, D_\tau) \approx \sum_{\tau=1}^{T} \nabla_\theta r(\theta_1, D_\tau) = \nabla_\theta r(\theta_1, D). \tag{43}$$

In the remainder of this work we will compute cosine similarities between weight-updates instead of gradients according to

$$\alpha_{i,j} := \frac{\langle \Delta\theta_i, \Delta\theta_j \rangle}{\|\Delta\theta_i\|\|\Delta\theta_j\|}, i, j \in c \tag{44}$$

Our experiments in section VI will demonstrate that computing cosine similarities based on weight-updates in practice surprisingly achieves *even better* separations than computing cosine similarities based on gradients.

B. Preserving Privacy

Every machine learning model carries information about the data it has been trained on. For example the bias term in the last layer of a neural network will typically carry information about the label distribution of the training data. Different authors have demonstrated that information about a client's input data ("x") can be inferred from the weight-updates it sends to the server via model inversion attacks [28][29][30][31][32]. In privacy sensitive situations it might be necessary to prevent this type of information leakage from clients to server with mechanisms like the ones presented in [5]. Luckily, Clustered Federated Learning can be easily augmented with an encryption mechanism that achieves this end. As both the cosine similarity between two clients' weight-updates and the norms of these updates are invariant to orthonormal transformations P (such as permutation of the indices),

$$\frac{\langle \Delta\theta_i, \Delta\theta_j \rangle}{\|\Delta\theta_i\|\|\Delta\theta_j\|} = \frac{\langle P\Delta\theta_i, P\Delta\theta_j \rangle}{\|P\Delta\theta_i\|\|P\Delta\theta_j\|} \tag{45}$$

a simple remedy is for all clients to apply such a transformation operator to their updates before communicating them to the server. After the server has averaged the updates from all clients and broadcasted the average back to the clients they simply apply the inverse operation

$$\Delta\theta = \frac{1}{n}\sum_{i=1}^{n} \Delta\theta_i = P^{-1}(\frac{1}{n}\sum_{i=1}^{n} P\Delta\theta_i) \tag{46}$$

and the Federated Learning protocol can resume unchanged. Other multi-task learning approaches require direct access to the client's data and hence can not be used together with encryption, which represents a distinct advantage for CFL in privacy sensitive situations.

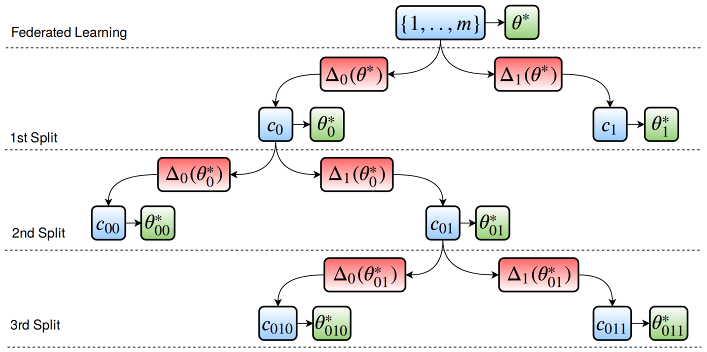

Fig. 4: An exemplary parameter tree created by Clustered Federated Learning. At the root node resides the conventional Federated Learning model, obtained by converging to a stationary point θ^* of the FL objective over all clients $\{1,..,m\}$. In the next layer, the client population has been split up into two groups, according to their cosine similarities and every subgroup has again converged to a stationary point θ^*_0 respective θ^*_1. Branching continues recursively until no stationary solution satisfies the splitting criteria. In order to quickly assign new clients to a leaf model, at each edge e of the tree the server caches the pre-split weight-updates Δ_e of all clients belonging to the two different sub-branches. This way the new client can be moved down the tree along the path of highest similarity.

C. Varying Client Populations and Parameter Trees

Up until now we always made the assumption that all clients participate from the beginning of training. Clustered Federated Learning however is flexible enough to handle client populations that vary over time.

In order to incorporate this functionality, the server, while running CFL, needs to build a parameter tree $T = (V, E)$ with the following properties:

- The tree contains a node $v \in V$ for every (intermediate) cluster c_v computed by CFL
- Both c_v and the corresponding stationary solution θ^*_v obtained by running the Federated Learning Algorithm 2 on cluster c_v are cached at node v
- At the root of the tree v_{root} resides the Federated Learning solution over the entire client population with $c_{v_{root}} = \{1,..,m\}$.

- If the cluster $c_{v_{child}}$ was created by bi-partitioning the cluster $c_{v_{parent}}$ in CFL then the nodes v_{parent} and v_{child} are connected via a directed edge $e \in E$
- At every edge $e(v_{parent} \to v_{child})$ the pre-split weight-updates of the children clients

$$\Delta_e = \{\text{SGD}_n(\theta^*_{v_{parent}}, D_i) - \theta_{v^*_{parent}} | i \in c_{v_{child}}\} \tag{47}$$

are cached

An exemplary parameter tree is shown in Figure 4. When a new client joins the training it can get assigned to a leaf cluster by iteratively traversing the parameter tree from the root to a leaf, always moving to the branch which contains the more similar client updates according to Algorithm 4.

Another feature of building a parameter tree is that it allows the server to provide every client with *multiple* models at varying specificity. On the path from root to leaf, the models get more specialized with the most general model being the FL model at the root. Depending on application and context, a CFL client could switch between models of different generality. Furthermore a parameter tree allows us to ensemble multiple models of different specificity together. We believe that investigations along those lines are a promising direction of future research.

Putting all pieces from the previous sections together, we arrive at a protocol for general privacy-preserving CFL which is described in Algorithm 5

Algorithm 4: Assigning new Clients to a Cluster

```
1  Input: new client with data $D_{new}$, parameter tree $T = (V, E)$
2  • $v \leftarrow v_{root}$
3  while $|\text{Children}(v)| > 0$ do
4      • $v_0, v_1 \leftarrow \text{Children}(v)$
5      • $\Delta\theta_{new} \leftarrow \text{SGD}_n(\theta^*_v, D_{new}) - \theta^*_v$
6      • $\alpha_0 \leftarrow \max_{\Delta\theta \in \Delta_{(v \to v_1)}} \alpha(\Delta\theta_{new}, \Delta\theta)$
7      • $\alpha_1 \leftarrow \max_{\Delta\theta \in \Delta_{(v \to v_2)}} \alpha(\Delta\theta_{new}, \Delta\theta)$
8      if $\alpha_0 > \alpha_1$ then
9          • $v \leftarrow v_0$
10     else
11         • $v \leftarrow v_1$
12     end
13 end
14 return $c_v$, $\theta^*_v$
```

Algorithm 5: Clustered Federated Learning with Privacy Preservation and Weight-Updates

```
1  input: initial parameters $\theta_0$, branching parameters $\varepsilon_1, \varepsilon_2 > 0$, empirical risk approximation error bound $\gamma_{max} \in [0, 1)$, number of local iterations/ epochs $n$
2  outout: improved parameters on every client $\theta_i$
3  init: set initial clusters $\mathcal{C} = \{\{1, .., m\}\}$, set initial models $\theta_i \leftarrow \theta_0 \ \forall i = 1, .., m$, set initial update $\Delta\theta_c \leftarrow 0 \ \forall c \in \mathcal{C}$, clients exchange random seed to create permutation operator $P$ (optional, otherwise set $P$ to be the identity mapping)
4  while not converged do
5      for $i = 1, .., m$ in parallel do
6          Client $i$ does:
7          • $\theta_i \leftarrow \theta_i + P^{-1}\Delta\theta_{c(i)}$
8          • $\Delta\theta_i \leftarrow P(\text{SGD}_n(\theta_i, D_i) - \theta_i)$
9      end
10     Server does:
11     • $\mathcal{C}_{tmp} \leftarrow \mathcal{C}$
12     for $c \in \mathcal{C}$ do
13         • $\Delta\theta_c \leftarrow \frac{1}{|c|}\sum_{i \in c} \Delta\theta_i$
14         if $\|\Delta\theta_c\| < \varepsilon_1$ and $\max_{i \in c} \|\Delta\theta_i\| > \varepsilon_2$ then
15             • $\alpha_{i,j} \leftarrow \frac{\langle \Delta\theta_i, \Delta\theta_j \rangle}{\|\Delta\theta_i\| \|\Delta\theta_j\|}$
16             • $c_1, c_2 \leftarrow \arg\min_{c_1 \cup c_2 = c}(\max_{i \in c_1, j \in c_2} \alpha_{i,j})$
17             • $\alpha^{max}_{cross} \leftarrow \max_{i \in c_1, j \in c_2} \alpha_{i,j}$
18             if $\gamma_{max} < \sqrt{\frac{1 - \alpha^{max}_{cross}}{2}}$ then
19                 • $\mathcal{C}_{tmp} \leftarrow (\mathcal{C}_{tmp} \setminus c) \cup c_1 \cup c_2$
20             end
21         end
22     end
23     • $\mathcal{C} \leftarrow \mathcal{C}_{tmp}$
24 end
25 return $\theta$
```

VI. Experiments

A. Practical Considerations

In section II we showed that the cosine similarity criterion does distinguish different incongruent clients under three conditions: (a) Federated Learning has converged to a stationary point θ^*, (b) Every client holds enough data s.t. the empirical risk approximates the true risk, (c) cosine similarity is computed between the full gradients of the empirical risk. In this section we will demonstrate that in practical problems none of these conditions have to be fully satisfied. Instead, we will find that CFL is able to correctly infer the clustering structure even if clients only hold small datasets and are trained to an approximately stationary solution of the Federated Learning objective. Furthermore we will see that cosine similarity can be computed between weight-updates instead of full gradients, which even improves performance.

In the experiments of this section we consider the following Federated Learning setup: All experiments are performed on either the MNIST [33] or CIFAR-10 [34] dataset using $m = 20$ clients, each of which belonging to one of $k = 4$ clusters. Every client is assigned an equally sized random subset of the total training data. To simulate an incongruent clustering structure, every clients' data is then modified by randomly swapping out two labels, depending on which cluster a client belongs to. For example, in all clients belonging to the first cluster, data points labeled as "1" could be relabeled as "7" and vice versa, in all clients belonging to the second cluster "3" and "5" could be switched out in the same way, and so on. This

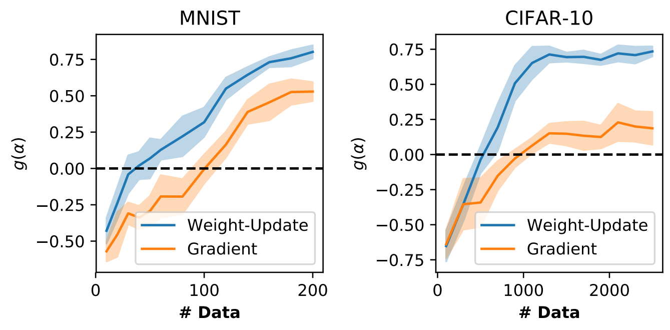

Fig. 5: Separation gap $g(\alpha)$ as a function of the number of data points on every client for the label-swap problem on MNIST and CIFAR. From Corollary 1 we know that CFL will always find a correct bi-partitioning if $g(\alpha) > 0$. On MNIST this is already satisfied if clients hold as little as 20 data points if weight-updates are used for the computation of the similarity α.

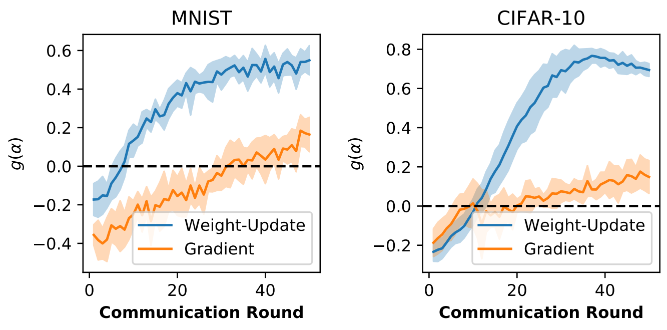

Fig. 6: Separation gap $g(\alpha)$ as a function of the number of communication rounds for the label-swap problem on MNIST and CIFAR. The separation quality monotonically increases with the number of communication rounds of Federated Learning. Correct separation in both cases is already achieved after around 10 communication rounds if α is computed using weight-updates.

relabeling ensures that both $\varphi(x)$ and $\varphi(y)$ are approximately the same across all clients, but the conditionals $\varphi(y|x)$ diverge between different clusters. We will refer to this as "label-swap augmentation" in the following. In all experiments we train multi-layer convolutional neural networks and adopt a standard Federated Learning strategy with 3 local epochs of training. We report the separation gap (Definition 2)

$$g(\alpha) := \alpha_{intra}^{min} - \alpha_{cross}^{max} \tag{48}$$

which according to Corollary 1 tells us whether CFL will correctly bi-partition the clients:

$$g(\alpha) > 0 \Leftrightarrow \text{"Correct Clustering"} \tag{49}$$

Number of Data points: We start out by investigating the effects of data set size on the cosine similarity. We randomly subsample from each client's training data to vary the number of data points on every client between 10 and 200 for MNIST and 100 and 2400 for CIFAR. For every different local data set size we run Federated Learning for 50 communication rounds, after which training progress has come mostly to halt and we can expect to be close to a stationary point. After round 50, we compute the pairwise cosine similarities between the weight-updates and the separation gap $g(\alpha)$. The results are shown in Figure 5. As expected, $g(\alpha)$ grows monotonically with increasing data set size. On the MNIST problem as little as 20 data points on every client are sufficient to achieve correct bi-partitioning in the sense of Definition 1. On the more difficult CIFAR problem a higher number of around 500 data points is necessary to achieve correct bi-partitioning.

Proximity to Stationary Solution: Next, we investigate the importance of proximity to a stationary point θ^* for the clustering. Under the same setting as in the previous experiment we reduce the number of data points on every client to 100 for MNIST and to 1500 for CIFAR and compute the pairwise cosine similarities and the separation gap after each of the first 50 communication rounds. The results are shown in Figure 6. Again, we see that the separation quality monotonically increases with the number of communication rounds. On MNIST and CIFAR as little as 10 communication rounds are necessary to obtain a correct clustering.

Weight-Updates instead of Gradients: In both the above experiments we computed the cosine similarities α based on either the full gradients

$$\alpha_{i,j} = \frac{\langle \nabla_\theta r_i(\theta), \nabla_\theta r_j(\theta)\rangle}{\|\nabla_\theta r_i(\theta)\|\|\nabla_\theta r_j(\theta)\|} \qquad \text{("Gradient")} \tag{50}$$

or Federated weight-updates

$$\alpha_{i,j} = \frac{\langle \Delta\theta_i, \Delta\theta_j\rangle}{\|\Delta\theta_i\|\|\Delta\theta_j\|} \qquad \text{("Weight-Update")} \tag{51}$$

over 3 epochs. Interestingly, weight-updates seem to provide even better separation $g(\alpha)$ with fewer data points and at a greater distance to a stationary solution. This comes in very handy as it allows us to leave the Federated Learning communication protocol unchanged. In all following experiments we will compute cosine similarities based on weight-updates instead of gradients.

B. *Distinguishing Congruent and Incongruent Clients*

In this subsection, we experimentally verify the validity of the clustering criteria (32) and (33) in a Federated Learning experiment on MNIST with two clients holding data from incongruent and congruent distributions. In the congruent case client one holds all training digits "0" to "4" and client two holds all training digits "5" to "9". In the incongruent case, both clients hold a random subset of the training data, but the distributions are modified according to the "label swap" rule described above. Figure 7 shows the development of the average update norm (equation (32)) and the maximum client norm (equation (33)) over the course of 1000 communication

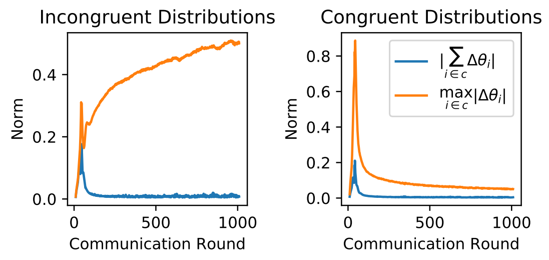

Fig. 7: Experimental verification of the norm criteria (33) and (32). Displayed is the development of gradient norms over the course of 1000 communication rounds of Federated Learning with two clients holding data from incongruent (left) and congruent distributions (right). In both cases Federated Learning converges to a stationary point of $F(\theta)$ and the average update norm (32) goes to zero. In the congruent case the maximum norm of the client updates (33) decreases along with the server update norm, while in contrast in the incongruent case it stagnates and even increases.

rounds. As predicted by the theory, in the congruent case the average client norm converges to zero, while in the incongruent case it stagnates and even increases over time. In both cases the server norm tends to zero, indicating convergence to a stationary point (see Figure 7).

C. Clustered Federated Learning

In this section, we apply CFL as described in Algorithm 5 to different Federated Learning setups, which are inspired by our motivating examples in the introduction. In all experiments, the clients perform 3 epochs of local training at a batch-size of 100 in every communication round.

Label permutation on Cifar-10: We split the CIFAR-10 training data randomly and evenly among $m = 20$ clients, which we group into $k = 4$ different clusters. All clients belonging to the same cluster apply the same random permutation $P_{c(i)}$ to their labels such that their modified training and test data is given by

$$\hat{D}_i = \{(x, P_{c(i)}(y)) | (x, y) \in D_i\} \tag{52}$$

respective

$$\hat{D_i^{test}} = \{(x, P_{c(i)}(y)) | (x, y) \in D^{test}\}. \tag{53}$$

The clients then jointly train a 5-layer convolutional neural network on the modified data using CFL with 3 epochs of local training at a batch-size of 100. Figure 8 (top) shows the joint training progression: In the first 50 communication rounds, all clients train one single model together, following the conventional Federated Learning protocol. After these initial 50 rounds, training has converged to a stationary point of the Federated Learning objective and the client test accuracies stagnate at around 20%. Conventional Federated Learning would be finalized at this point. At the same time, we observe (Figure 8, bottom) that a distinct gap $g(\alpha) = \alpha_{intra}^{min} - \alpha_{cross}^{max}$

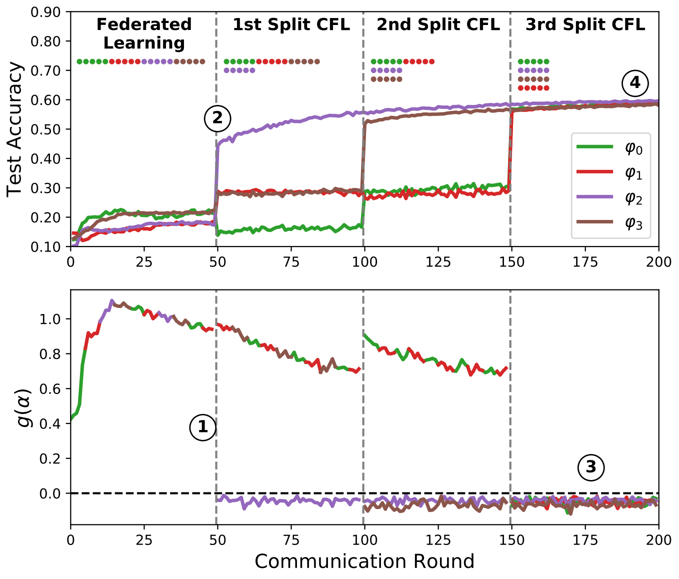

Fig. 8: CFL applied to the "permuted labels problem" on CIFAR with 20 clients and 4 different permutations. The top plot shows the accuracy of the trained model(s) on their corresponding validation sets. The bottom plot shows the separation gaps $g(\alpha)$ for all different clusters. After an initial 50 communication rounds a large separation gap has developed and a first split separates out the purple group of clients, which leads to an immediate drastic increase of accuracy for these clients. In communication rounds 100 and 150 this step is repeated until all clients with incongruent distributions have been separated. After the third split, the model accuracy for all clients has more than doubled and the separation gaps in all clusters have dropped to below zero which indicates that the clustering is finalized.

has developed (①), indicating an underlying clustering structure. In communication round 50 the client population is therefore split up for the first time, which leads to an immediate 25% increase in validation accuracy for all clients belonging to the "purple" cluster which was separated out ②. Splitting is repeated in communication rounds 100 and 150 until all clusters have been separated and $g(\alpha)$ has dropped to below zero in all clusters (③), which indicates that clustering is finalized. At this point the accuracy of all clients has more than doubled the one achieved by the Federated Learning solution and is now at close to 60% ④. This underlines, that after standard FL, our novel CFL can detect, the necessity for subsequent splitting and clustering which enables arriving at significantly higher performance. In addition, the cluster structure found can potentially be illuminating as it provides interesting insight about the composition of the complex underlying data distribution.

Language Modeling on Ag-News: The Ag-News corpus is a collection of 120000 news articles belonging to one of the four topics 'World', 'Sports', 'Business' and 'Sci/Tech'.

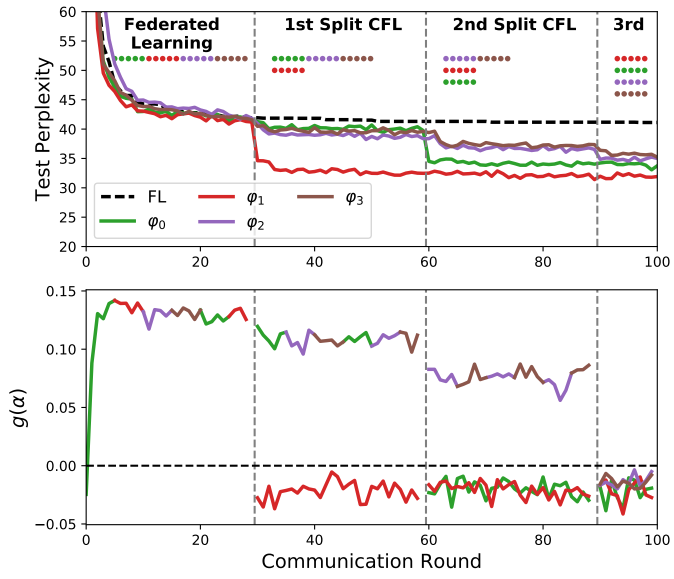

Fig. 9: CFL applied to the Ag-News problem. The top plot shows the perplexity achieved by the different clients on their local test set (lower is better). The clients are separated in communication rounds 30, 60 and 90. After the final separation the perplexity of all clients on their local test set has dropped to less than 36, while the Federated Learning solution (black, dotted) still stagnates at a perplexity of 42.

We split the corpus into 20 different sub-corpora of the same size, with every sub-corpus containing only articles from one topic and assign every corpus to one client. Consequently the clients form four clusters based on what type of articles they hold. Every Client trains a two-layer LSTM network to predict the next word on it's local corpus of articles. Figure 9 shows 100 communication rounds of multi-stage CFL applied to this distributed learning problem. As we can see, Federated Learning again converges to a stationary solution after around 30 communication rounds. At this solution all clients achieve a perplexity of around 43 on their local test set. After the client population has been split up in communication rounds 30, 60 and 90, the four true underlying clusters are discovered. After the 100th communication round the perplexity of all clients has dropped to less than 36. The Federated Learning solution, trained over the same amount of communication rounds, still stagnates at an average perplexity of 42.

VII. Conclusion

In this paper we presented Clustered Federated Learning, a framework for Federated Multi-Task Learning that can improve any existing Federated Learning Framework by enabling the participating clients to learn more specialized models. Clustered Federated Learning makes use of our theoretical finding, that (at any stationary solution of the Federated Learning objective) the cosine similarity between the weight-updates of different clients is highly indicative of the similarity of their data distributions. This crucial insight allows us to provide strong mathematic guarantees on the clustering quality under mild assumptions on the clients and their data, even for arbitrary non-convex objectives.

We demonstrated that CFL can be implemented in a privacy preserving way and without having to modify the FL communication protocol. Moreover, CFL is able to distinguish situations in which a single model can be learned from the clients' data from those in which this is not possible and only separates clients in the latter case.

Our experiments on convolutional and recurrent deep neural networks show that CFL can achieve drastic improvements over the Federated Learning baseline in terms of classification accuracy / perplexity in situations where the clients' data exhibits a clustering structure.

References

[1] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.

[2] A. Ghosh, J. Hong, D. Yin, and K. Ramchandran, "Robust federated learning in a heterogeneous environment," *arXiv preprint arXiv:1906.06629*, 2019.

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

[4] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[5] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy preserving machine learning." *IACR Cryptology ePrint Archive*, vol. 2017, p. 281, 2017.

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

[7] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *arXiv preprint arXiv:1908.07873*, 2019.

[8] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[9] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, 2018.

[10] S. Caldas, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," *arXiv preprint arXiv:1812.01097*, 2018.

[11] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, p. 12, 2019.

[12] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi, "Don't use large mini-batches, use local sgd," *arXiv preprint arXiv:1808.07217*, 2018.

[13] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Sparse binary compression: Towards distributed deep learning with minimal communication," *arXiv preprint arXiv:1805.08768*, 2018.

[14] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *arXiv preprint arXiv:1903.02891*, 2019.

[15] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[16] P. Jiang and G. Agrawal, "A linear speedup analysis of distributed deep learning with sparse and quantized communication," in *Advances in Neural Information Processing Systems*, 2018, pp. 2525–2536.

[17] H. Yu, S. Yang, and S. Zhu, "Parallel restarted sgd for non-convex optimization with faster convergence and less communication," *arXiv preprint arXiv:1807.06629*, 2018.

[18] A. Koloskova, T. Lin, S. U. Stich, and M. Jaggi, "Decentralized deep learning with arbitrary communication compression," *arXiv preprint arXiv:1907.09356*, 2019.

[19] A. Koloskova, S. U. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," *arXiv preprint arXiv:1902.00340*, 2019.

[20] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," in *Advances in Neural Information Processing Systems*, 2018, pp. 4447–4458.

[21] V. Smith, S. Forte, C. Ma, M. Takac, M. I. Jordan, and M. Jaggi, "Cocoa: A general framework for communication-efficient distributed optimization," *arXiv preprint arXiv:1611.02189*, 2016.

[22] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[23] L. Jacob, J.-p. Vert, and F. R. Bach, "Clustered multi-task learning: A convex formulation," in *Advances in neural information processing systems*, 2009, pp. 745–752.

[24] A. Kumar and H. Daume III, "Learning task grouping and overlap in multi-task learning," *arXiv preprint arXiv:1206.6417*, 2012.

[25] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.

[26] L. Corinzia and J. M. Buhmann, "Variational federated multi-task learning," *arXiv preprint arXiv:1906.06268*, 2019.

[27] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.

[28] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," *arXiv preprint arXiv:1812.00984*, 2018.

[29] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the gan: information leakage from collaborative deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 603–618.

[30] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1322–1333.

[31] N. Carlini, C. Liu, J. Kos, Ú. Erlingsson, and D. Song, "The secret sharer: Measuring unintended neural network memorization & extracting secrets," *arXiv preprint arXiv:1802.08232*, 2018.

[32] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," *arXiv preprint arXiv:1805.04049*, 2018.

[33] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[34] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset," *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.