On the effectiveness of partial variance reduction in federated learning with heterogeneous data

Bo Li*  Mikkel N. Schmidt  Tommy S. Alstrøm
Technical University of Denmark
{blia, mnc, tsal}@dtu.dk

Sebastian U. Stich
CISPA‡
stich@cispa.de

Abstract

Data heterogeneity across clients is a key challenge in federated learning. Prior works address this by either aligning client and server models or using control variates to correct client model drift. Although these methods achieve fast convergence in convex or simple non-convex problems, the performance in over-parameterized models such as deep neural networks is lacking. In this paper, we first revisit the widely used FedAvg algorithm in a deep neural network to understand how data heterogeneity influences the gradient updates across the neural network layers. We observe that while the feature extraction layers are learned efficiently by FedAvg, the substantial diversity of the final classification layers across clients impedes the performance. Motivated by this, we propose to correct model drift by variance reduction only on the final layers. We demonstrate that this significantly outperforms existing benchmarks at a similar or lower communication cost. We furthermore provide proof for the convergence rate of our algorithm.

1. Introduction

Federated learning (FL) is emerging as an essential distributed learning paradigm in large-scale machine learning. Unlike in traditional machine learning, where a model is trained on the collected centralized data, in federated learning, each client (e.g., phones and institutions) learns a model with its local data. A centralized model is then obtained by aggregating the updates from all participating clients without ever requesting the client data, thereby ensuring a certain level of user privacy [13, 17]. Such an algorithm is especially beneficial for tasks where the data is sensitive, e.g., chemical hazards detection and diseases diagnosis [33].

Two primary challenges in federated learning are i) handling data heterogeneity across clients [13] and ii) limiting the cost of communication between the server and clients [10]. In this setting, FedAvg [17] is one of the most widely used schemes: A server broadcasts its model to clients, which then update the model using their local data in a series of steps before sending their individual model to the server, where the models are aggregated by averaging the parameters. The process is repeated for multiple communication rounds. While it has shown great success in many applications, it tends to achieve subpar accuracy and convergence when the data are heterogeneous [14, 24, 31].

The slow and sometimes unstable convergence of FedAvg can be caused by client drift [14] brought on by data heterogeneity. Numerous efforts have been made to improve FedAvg’s performance in this setting. Prior works attempt to mitigate client drift by penalizing the distance between a client model and the server model [20, 31] or by performing variance reduction techniques while updating client models [1, 14, 32]. These works demonstrate fast convergence on convex problems or for simple neural networks; however, their performance on deep neural

* Work done while at CISPA
‡ CISPA Helmholtz Center for Information Security
networks, which are state-of-the-art for many centralized learning tasks [1, 34], has yet to be well explored. Adapting techniques that perform well on convex problems to neural networks is non-trivial [7] due to their “intriguing properties” [38] such as over-parametrization and permutation symmetries.

To overcome the above issues, we revisit the FedAvg algorithm with a deep neural network (VGG-11 [34]) under the assumption of data heterogeneity and full client participation. Specifically, we investigate which layers in a neural network are mostly influenced by data heterogeneity. We define drift diversity, which measures the diversity of the directions and scales of the averaged gradients across clients per communication round. We observe that in the non-IID scenario, the deeper layers, especially the final classification layer, have the highest diversity across clients compared to an IID setting. This indicates that FedAvg learns good feature representations even in the non-IID scenario [3] and that the significant variation of the deeper layers across clients is a primary cause of FedAvg’s subpar performance.

Based on the above observations, we propose to align the classification layers across clients using variance reduction. Specifically, we estimate the average updating direction of the classifiers (the last several fully connected layers) at the client $c_i$ and server level $e$ and use their difference as a control variate [14] to reduce the variance of the classifiers across clients. We analyze our proposed algorithm and derive a convergence rate bound.

We perform experiments on the popular federated learning benchmark datasets CIFAR10 [19] and CIFAR100 [19] using two types of neural networks, VGG-11 [34] and ResNet-8 [11], and different levels of data heterogeneity across clients. We experimentally show that we require fewer communication rounds compared to the existing methods [14, 17, 31] to achieve the same accuracy while transmitting a similar or slightly larger number of parameters between server and clients than FedAvg (see Fig. 1). With a (large) fixed number of communication rounds, our method achieves on-par or better top-1 accuracy, and in some settings it even outperforms centralized learning. Using conformal prediction [3], we show how performance can be improved further using adaptive prediction sets.

We show that applying variance reduction on the last layers increases the diversity of the feature extraction layers. This diversity in the feature extraction layers may give each client more freedom to learn richer feature representations, and the uniformity in the classifier then ensures a less biased decision. We summarize our contributions here:

- We present our algorithm for partial variance-reduced federated learning (FedPVR). We experimentally demonstrate that the key to the success of our algorithm is the diversity between the feature extraction layers and the alignment between the classifiers.
- We prove the convergence rate in the convex settings and non-convex settings, precisely characterize its weak dependence on data-heterogeneity measures and show that FedPVR provably converges as fast as the centralized SGD baseline in most practical relevant cases.
- We experimentally show that our algorithm is more communication efficient than previous works across various levels of data heterogeneity, datasets, and neural network architectures. In some cases where data heterogeneity exists, the proposed algorithm even performs slightly better than centralized learning.

2. Related work

2.1. Federated learning

Federated learning (FL) is a fast-growing field [13, 40]. We mainly describe FL methods in non-IID settings where the data is distributed heterogeneously across clients. Among the existing approaches, FedAvg [25] is the de facto optimization technique. Despite its solid empirical performances in IID settings [13, 25], it tends to achieve a subpar accuracy-communication trade-off in non-IID scenarios.

Many works attempt to tackle FL when data is heterogeneous across clients [1, 9, 14, 31, 39, 42]. FedProx [31] proposes a temperature parameter and proximal regularization term to control the divergence between client and server models. However, the proximal term does not bring the alignment between the global and local optimal points [1]. Similarly, some works control the update direction by introducing client-dependent control variate [1, 14, 17, 27, 32] that is also communicated between the server and clients. They have achieved a much faster convergence rate, but their performance in a non-convex setup, especially in deep neural networks, such as ResNet [11] and VGG [34], is not well explored. Besides, they suffer from a higher communication cost due to the transmission of the extra control variates, which may be a critical issue for resources-limited IoT mobile devices [10]. Among these methods, SCAF-FOLD [14] is the most closely related method to ours, and we give a more detailed comparison in section 3 and 5.

Another line of work develops FL algorithms based on the characteristics, such as expressive feature representations [28] of neural networks. Collins et al. [5] show that FedAvg is powerful in learning common data representations from clients’ data. FedBabu [29], TCT [41], and CCVR [24] propose to improve FL performance by fine-tuning the classifiers with a standalone dataset or features that are simulated based on the client models. However, preparing a standalone dataset/features that represents the data distribution across clients is challenging as this usually requires domain knowledge and may raise privacy concerns.
Deep layers in an over-parameterised neural network have higher disagreement and variance when the clients are heterogeneous using FedAvg.

Moon [20] encourages the similarity of the representations across different client models by using contrastive loss [4] but with the cost of three full-size models in memory on each client, which may limit its applicability in resource-limited devices.

Other works focus on reducing the communication cost by compressing the transmitted gradients [2, 8, 21, 26, 36]. They can reduce the communication bandwidth by adjusting the number of bits sent per iteration. These works are complementary to ours and can be easily integrated into our method to save communication costs.

Table 1. Notations used in this paper

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R, r$</td>
<td>Number of communication rounds and round index</td>
</tr>
<tr>
<td>$K, k$</td>
<td>Number of local steps, local step index</td>
</tr>
<tr>
<td>$N, i$</td>
<td>Number of clients, client index</td>
</tr>
<tr>
<td>$y_{i,k}^r$</td>
<td>client model $i$ at step $k$ and round $r$</td>
</tr>
<tr>
<td>$x^r$</td>
<td>server model at round $r$</td>
</tr>
<tr>
<td>$c^r_i, c^r$</td>
<td>client and server control variate</td>
</tr>
</tbody>
</table>

3. Method

3.1. Problem statement

Given $N$ clients with full participation, we formalise the problem as minimizing the average of the stochastic functions with access to stochastic samples in Eq. 1 where $x$ is the model parameters and $f_i$ represents the loss function at client $i$ with dataset $D_i$,

$$\min_{x \in \mathbb{R}^d} \left( f(x) := \frac{1}{N} \sum_{i=1}^{N} f_i(x) \right),$$

where $f_i(x) := E_{D_i}[f_i(x; D_i)]$.

3.2. Motivation

When the data $\{D_i\}$ are heterogeneous across clients, FedAvg suffers from client drift [14], where the average of the local optimal $x^* = \frac{1}{N} \sum_{i \in N} x_i^*$ is far from the global optimal $x^*$. To understand what causes client drift, specifically which layers in a neural network are influenced most by the data heterogeneity, we perform a simple experiment using FedAvg and CIFAR10 datasets on a VGG-11. The detailed experimental setup can be found in section 4.
In an over-parameterized model, it is difficult to directly calculate client drift $|\|x^r - x^{r-1}\|_2^2$ as it is challenging to obtain the global optimum $x^r$. We instead hypothesize that we can represent the influence of data heterogeneity on the model by measuring 1) drift diversity and 2) client model similarity. Drift diversity reflects the diversity in the amount each client model deviates from the server model after an update round.

**Definition 1 (Drift diversity).** We define the drift diversity across $N$ clients at round $r$ as:

$$
\xi^r := \frac{\sum_{i=1}^{N} ||m_i^r||^2}{\sum_{i=1}^{N} m_i^r} = y_{i,K} - x^{r-1}
$$

Drift diversity $\xi^r$ is high when all the clients update their models in different directions, i.e., when dot products between client updates $m_i$ are small. When each client performs $K$ steps of vanilla SGD updates, $\xi$ depends on the directions and amplitude of the gradients over $N$ clients and is equivalent to $\frac{\sum_{i=1}^{N} \sum_{k=1}^{m} g_i(y_{i,k})}{\sum_{i=1}^{N} \sum_{k=1}^{m} ||y_{i,k}||_2^2}$, where $g_i(y_{i,k})$ is the stochastic mini-batch gradient.

After updating client models, we quantify the client model similarity using centred kernel alignment (CKA) [18] computed on a test dataset. CKA is a widely used permutation invariant metric for measuring the similarity between feature representations in neural networks [18, 24, 28].

Fig. 2 shows the movement of $\xi$ and CKA across different levels of data heterogeneity using FedAvg. We observe that the similarity and diversity of the early layers (e.g., layer index 4 and 12) are with a higher agreement between the IID ($\alpha = 100.0$) and non-IID ($\alpha = 0.1$) experiments, which indicates that FedAvg can still learn and extract good feature representations even when it is trained with non-IID data. The lower similarity on the deeper layers, especially the classifiers, suggests that these layers are strongly biased towards their local data distribution. When we only look at the model that is trained with $\alpha = 0.1$, we see the highest diversity and variance on the classifiers across clients compared to the rest of the layers. Based on the above observations, we propose to align the classifiers across clients using variance reduction. We deploy client and server control variates to control the updating directions of the classifiers.

### 3.3. Classifier variance reduction

Our proposed algorithm (Alg. 1) consists of three parts: i) client updating (Eq. 5-6) ii) client control variate updating, (Eq. 7), and iii) server updating (Eq. 8-9).

We first define a vector $p \in \mathbb{R}^d$ that contains 0 or 1 with $v$ non-zero elements ($v \ll d$) in Eq. 3. We recover SCAFFOLD with $p = 1$ and recover FedAvg with $p = 0$. For the set of indices $j$ where $p_j = 1$ ($S_{svr}$ from Eq. 4), we update the corresponding weights $y_{j,Svr}$ with variance reduction such that we maintain a state for each client ($c_i \in \mathbb{R}^v$) and for the server ($c \in \mathbb{R}^v$) in Eq. 5. For the rest of the indices $S_{sgd}$ from Eq. 4, we update the corresponding weights $y_{j,svr}$ with SGD in Eq. 6. As the server variate $c$ is an average of $c_i$ across clients, we can safely initialise them as 0.

In each communication round, each client receives a copy of the server model $x$ and the server control variate $c$. They then perform $K$ model updating steps (see Eq. 5-6) for one step) using cross-entropy as the loss function. Once this is finished, we calculate the updated client control variate $c_i$ using Eq. 7. The server then receives the updated $c_i$ and $y_j$ from all the clients for aggregation (Eq. 8-9). This completes one communication round.

**Algorithm 1** Partial variance reduction (FedPVR)

**server:** initialise the server model $x$, the control variate $c$, and global step size $\eta_g$

**client:** initialise control variate $c_i$, and local step size $\eta_l$

**mask:** $p := \{0,1\}^d$, $S_{svr} := \{j : p_j = 0\}$, $S_{sgd} := \{j : p_j = 1\}$

1: procedure MODEL UPDATING
2: for $r = 1 \rightarrow R$ do
3: communicate $x$ and $c$ to all clients $i \in [N]$
4: for $On$ client $i \in [N]$ in parallel do
5: $y_i \leftarrow x$
6: for $k = 1 \rightarrow K$ do
7: compute minibatch gradient $g_i(y_i)$
8: $y_i,svr \leftarrow y_i,svr - \eta_l g_i(y_i)$
9: $y_i,svr \leftarrow y_i,svr - \eta_l g_i(y_i) - c_i + c$
10: end for
11: $c_i \leftarrow c_i - c + \frac{1}{\eta_l} (x_{svr} - y_i)_{svr}$
12: communicate $y_i, c_i$
13: end for
14: $x \leftarrow (1 - \eta_g) x + \frac{1}{\eta_l} \sum_{i \in N} y_i$
15: $c \leftarrow \frac{1}{n} \sum_{i \in N} c_i$
16: end for
17: end procedure

In terms of implementation, we can simply assume the control variate for the block of weights that is updated with SGD as 0 and implement line 8 and 9 in one step.

**Ours vs SCAFFOLD** [14] While our work is similar to SCAFFOLD in the use of variance reduction, there are some fundamental differences. We both communicate control variates between the clients and server, but our control...
variante (2v ≤ 0.1d) is significantly smaller than the one in SCAFFOLD (2d). This 2x decrease in bits can be critical for some low-power IoT devices as the communication may consume more energy [10]. From the application point of view, SCAFFOLD achieved great success in convex or simple two layers problems. However, adapting the techniques that work well from convex problems to over-parameterized models is non-trivial [38], and naively adapting variance reduction techniques on deep neural networks gives little or no convergence speedup [7]. Therefore, the significant improvement achieved by our method gives essential and non-trivial insight into what matters when tackling data heterogeneity in FL in over-parameterized models.

### 3.4. Convergence rate

We state the convergence rate in this section. We assume functions \( \{f_i\} \) are \( \beta \)-smooth following [16, 35]. We then assume \( g_i(\mathbf{x}) := \nabla f_i(\mathbf{x}; D_i) \) is an unbiased stochastic gradient of \( f_i \) with variance bounded by \( \sigma^2 \). We assume strongly convexity (\( \mu > 0 \)) and general convexity (\( \mu = 0 \)) for some of the results following [14]. Furthermore, we also make assumptions about the heterogeneity of the functions.

For convex functions, we assume the heterogeneity of the function \( \{f_i\} \) at the optimal point \( \mathbf{x}^* \) (such a point always exists for a strongly convex function) following [15, 16].

**Assumption 1** (\( \zeta \)-heterogeneity). We define a measure of variance at the optimum \( \mathbf{x}^* \) given \( N \) clients as:

\[
\zeta^2 := \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}[\|\nabla f_i(\mathbf{x})\|^2].
\] (10)

For the non-convex functions, such an unique optimal point \( \mathbf{x}^* \) does not necessarily exist, so we generalize Assumption 1 to Assumption 2.

**Assumption 2** (\( \tilde{\zeta} \)-heterogeneity). We assume there exists constant \( \zeta \) such that \( \forall \mathbf{x} \in \mathbb{R}^d \)

\[
\frac{1}{N} \sum_{i=1}^{N} \mathbb{E}[\|\nabla f_i(\mathbf{x})\|^2] \leq \tilde{\zeta}^2.
\] (11)

Given the mask \( p \) as defined in Eq. 3, we know \( \|p \odot \mathbf{x}\| \leq \|\mathbf{x}\| \). Therefore, we have the following propositions.

**Proposition 1** (Implication of Assumption 1). Given the mask \( p \), we define the heterogeneity of the block of weights that are not variance reduced at the optimum \( \mathbf{x}^* \) as:

\[
\zeta^2_{1-p} := \frac{1}{N} \sum_{i=1}^{N} \|(1-p) \odot \nabla f_i(\mathbf{x}^*)\|^2.
\] (12)

If Assumption 1 holds, then it also holds that:

\[
\zeta^2_{1-p} \leq \zeta^2.
\] (13)

In Proposition 1, \( \zeta^2_{1-p} = \zeta^2 \) if \( p = 0 \) and \( \zeta^2_{1-p} = 0 \) if \( p = 1 \). If \( p \neq 0 \) and \( p \neq 1 \), as the heterogeneity of the shallow weights is lower than the deeper weights [41], we have \( \zeta^2_{1-p} \leq \zeta^2 \). Similarly, we can validate Proposition 2.

**Proposition 2** (Implication of Assumption 2). Given the mask \( p \), we assume there exists constant \( \tilde{\zeta}_{1-p} \) such that \( \forall \mathbf{x} \in \mathbb{R}^d \), the heterogeneity of the block of weights that are not variance reduced:

\[
\frac{1}{N} \sum_{i=1}^{N} \|(1-p) \odot \nabla f_i(\mathbf{x})\|^2 \leq \tilde{\zeta}_{1-p}.
\] (14)

If Assumption 2 holds, then it also holds that:

\[
\tilde{\zeta}^2_{1-p} \leq \tilde{\zeta}^2.
\] (15)

**Theorem 1.** For any \( \beta \)-smooth function \( \{f_i\} \), the output of FedPVR has expected error smaller than \( \epsilon \) for \( \eta_g = \sqrt{N} \) and some values of \( \eta_g \). R satisfying:

- **Strongly convex:** \( \eta_g \leq \min \left( \frac{1}{80 \mu KN^2 \beta}, \frac{26}{20 \mu KN^2 \beta} \right) \).

\[
R = \tilde{O}\left( \frac{\sigma^2}{\mu KN \epsilon^2} + \frac{\tilde{\zeta}^2_{1-p} D}{\epsilon^2} + \frac{\beta D}{\epsilon} + F \right),
\] (16)

- **General convex:** \( \eta_g \leq \frac{1}{20 \mu N^2 \beta} \), and \( R \geq 1 \), then:

\[
R = O\left( \frac{\beta \sigma^2}{\mu KN c^2} + \frac{\tilde{\zeta}^2_{1-p} F}{\mu N c^2} + \frac{\beta F}{\epsilon} \right),
\] (18)

Where \( D := \|\mathbf{x}^0 - \mathbf{x}^*\|^2 \) and \( F := f(\mathbf{x}^0) - f^* \).

Given the above assumptions, the convergence rate is given in Theorem 1. When \( p = 1 \), we recover SCAFFOLD convergence guarantee as \( \zeta^2_{1-p} = 0, \tilde{\zeta}^2_{1-p} = 0 \). In the strongly convex case, the effect of the heterogeneity of the block of weights that are not variance reduced \( \zeta^2_{1-p} \) becomes negligible if \( \tilde{O}\left( \frac{\tilde{\zeta}^2_{1-p}}{\epsilon^2} \right) \) is sufficiently smaller than \( \tilde{O}\left( \frac{\sigma^2}{\mu KN} \right) \). In such case, our rate is \( \frac{\sigma^2}{\mu KN c^2} + \frac{1}{\mu} \), which recovers the SCAFFOLD in the strongly convex without sampling and further matches that of SGD (with mini-batch size \( K \) on each worker). We also recover the FedAvg rate at simple IID case. See Appendix B for the full proof.
4. Experimental setup

We demonstrate the effectiveness of our approach with CIFAR10 [19] and CIFAR100 [19] on image classification tasks. We simulate the data heterogeneity scenario following [22] by partitioning the data according to the Dirichlet distribution with the concentration parameter $\alpha$. The smaller the $\alpha$ is, the more imbalanced the data are distributed across clients. An example of the data distribution over multiple clients using the CIFAR10 dataset can be seen in Fig. 2. In our experiment, we use $\alpha \in \{0.1, 0.5, 1.0\}$ as these are commonly used concentration parameters [22]. Each client has its local data, and this data is kept to be the same during all the communication rounds. We hold out the test dataset at the server for evaluating the classification performance of the server model. Following [22], we perform the same data augmentation for all the experiments.

We use two models: VGG-11 and ResNet-8 following [22]. We perform variance reduction for the last three layers in VGG-11 and the last layer in ResNet-8. We use 10 clients with full participation following [31] (close to cross-silo setup) and a batch size of 256. Each client performs 10 local epochs of model updating. We set the server learning rate $\eta_y = 1$ for all the models [14]. We tune the clients learning rate from $[0.05, 0.1, 0.2, 0.3]$ for each individual experiment. The learning rate schedule is experimentally chosen from constant, cosine decay [23], and multiple step decay [22]. We compare our method with the representative federated learning algorithms FedAvg [25], FedProx [31], SCAFFOLD [14], and FedDyn [1]. All the results are averaged over three repeated experiments with different random initialization. We leave 1% of the training data from each client out as the validation data to tune the hyperparameters (learning rate and schedule) per client. See Appendix. C for more experimental setups. The code is at github.com/lyn1874/fedpvr.

5. Experimental results

We demonstrate the performance of our proposed approach in the FL setup with data heterogeneity in this section. We compare our method with the existing state-of-the-art algorithms on various datasets and deep neural networks. For the baseline approaches, we finetune the hyperparameters (learning rate and schedule) per client. See Appendix. C for more information about the baseline approaches, we require much fewer number of rounds for almost all types of data heterogeneity and models. We can achieve a speedup between 1.5 and 6.7 than FedAvg. We also observe that ResNet-8 tends to converge slower than VGG-11, which may be due to the aggregation of the Batch Normalization layers that are discrepant between the local data distribution [22].

We next compare the top-1 accuracy between centralized learning and federated learning algorithms. For the centralized learning experiment, we tune the learning rate from $\{0.01, 0.05, 0.1\}$ and report the best test accuracy based on the validation dataset. We train the model for 800 epochs which is as same as the total number of epochs in the federated learning algorithms (80 communication rounds x 10 local epochs). The results are shown in Table. 3. We also show the number of copies of the parameters that need to be transmitted between the server and clients (e.g. 2x means we communicate $x$ and $y$).

Table. 3 shows that our approach achieves a much better Top-1 accuracy compared to FedAvg with transmitting a similar or slightly bigger number of parameters between the server and client per round. Our method also achieves slightly better accuracy than centralized learning when the data is less heterogeneous (e.g. $\alpha = 0.5$ for CIFAR10 and $\alpha = 1.0$ for CIFAR100).

5.2. Conformal prediction

When the data heterogeneity is high across clients, it is difficult for a federated learning algorithm to match the centralized learning performance [24]. Therefore, we demonstrate the benefit of using simple post-processing conformal prediction to improve the model performance.

We examine the relationship between the empirical coverage and the average predictive set size for the server model after 80 communication rounds for each federated learning algorithm. The empirical coverage is the percentage of the data samples where the correct prediction is in the predictive set, and the average predictive size is the average of the length of the predictive sets over all the test images [3]. See Appendix. C for more information about conformal prediction setup and results.

The results for when $\alpha = 0.1$ for both datasets and architectures are shown in Fig. 3. We show that by slightly increasing the predictive set size, we can achieve a similar accuracy as the centralized performance. Besides, our approach tends to surpass the centralized top-1 performance similar to or faster than other approaches. In sensitive use cases such as chemical threat detection, conformal prediction is a valuable tool to achieve certified accuracy at the
Ours FedDyn SCAFFOLD FedProx FedAvg
with only lightweight modifications to FedAvg. We next speedup and performance against the existing approaches to surpass the centralised Top-1 accuracy.

5.3. Diversity and uniformity

We have shown that our algorithm achieves a better speedup and performance against the existing approaches with only lightweight modifications to FedAvg. We next investigate what factors lead to better accuracy. Specifically, we calculate the drift diversity $\xi$ across clients after each communication round using Eq. 2 and average $\xi$ across three runs. We show the result of using ResNet-8 and CIFAR10 with $\alpha = 1.0$ in Fig. 4.

Fig. 4 shows the drift diversity for different layers in ResNet-8 and the testing accuracy along the communication rounds. We observe that classifiers have the highest diver-
sity in FedAvg against other layers and methods. SCAFFOLD, which applies control variate on the entire model, can effectively reduce the disagreement of the directions and scales of the averaged gradient across clients. Our proposed algorithm that performs variance reduction only on the classifiers can reduce the diversity of the classifiers even further but increase the diversity of the feature extraction layers. This high diversity tends to boost the learning speed as the curvature of the diversity movement (Fig. 4 left) seems to match the learning curve (Fig. 4 right). Based on this observation, we hypothesize that this diversity along the feature extractor and the uniformity of the classifier is the main reason for our better speedup.

To test this hypothesis, we perform an experiment where we use variance reduction starting from different layers of a neural network. If the starting position of the use of variance reduction influences the learning speed, it indicates where in a neural network we need more diversity and where we need more uniformity. We here show the result of using VGG-11 on CIFAR100 with $\alpha = 1.0$ as there are more layers in VGG-11. The result is shown in Fig. 5 where $\text{SVR}:16 \rightarrow 20$ is corresponding to our approach and $\text{SVR}:0 \rightarrow 20$ is corresponding to SCAFFOLD that applies variance reduction for the entire model. Results for using ResNet-8 is shown in Appendix. C.

![Figure 5. Influence of using stochastic variance reduction (SVR) on layers that start from different positions in a neural network on the learning speed. $\text{SVR}:0 \rightarrow 20$ applies variance reduction on the entire model (SCAFFOLD). $\text{SVR}:16 \rightarrow 20$ applies variance reduction from the layer index 16 to 20 (ours). The later we apply variance reduction, the better performance speedup we obtain. However, no variance reduction (FedAvg) performs the worst here.](image)

We see from Fig. 5 that the deeper in a neural network we apply variance reduction, the better learning speedup we can obtain. There is no clear performance difference between where to activate the variance reduction when the layer index is over 10. However, applying no variance reduction (FedAvg) achieves by far the worst performance. We believe that these experimental results indicate that in a distributed optimization framework, to boost the learning speed of an over-parameterized model, we need some levels of diversity in the middle and early layers for learning richer feature representation and some degrees of uniformity in the classifiers for making a less biased decision.

6. Conclusion

In this work, we studied stochastic gradient descent learning for deep neural network classifiers in a federated learning setting, where each client updates its local model using stochastic gradient descent on local data. A central model is periodically updated (by averaging local model parameters) and broadcast to the clients under a communication bandwidth constraint. When data is homogeneous across clients, this procedure is comparable to centralized learning in terms of efficiency; however, when data is heterogeneous, learning is impeded. Our hypothesis for the primary reason for this is that when the local models are out of alignment, updating the central model by averaging is ineffective and sometimes even destructive.

Examining the diversity across clients of their local model updates and their learned feature representations, we found that the misalignment between models is much stronger in the last few neural network layers than in the rest of the network. This finding inspired us to experiment with aligning the local models using a partial variance reduction technique applied only on the last layers, which we named FedPVR. We found that this led to a substantial improvement in convergence speed compared to the competing federated learning methods. In some cases, our method even outperformed centralized learning. We derived a bound on the convergence rate of our proposed method, which matches the rates for SGD when the gradient diversity across clients is sufficiently low. Compared with FedAvg, the communication cost of our method is only marginally worse, as it requires transmitting control variates for the last layers.

We believe our FedPVR algorithm strikes a good balance between simplicity and efficiency, requiring only a minor modification to the established FedAvg method; however, in our further research, we plan to pursue more optimal methods for aligning and guiding the local learning algorithms, e.g. using adaptive procedures. Furthermore, the degree of over-parameterization in the neural network layers (e.g. feature extraction vs bottlenecks) may also play an important role, which we would like to understand better.

Acknowledgements

The first three authors thank for financial support from the European Union’s Horizon 2020 research and innovation programme under grant agreement no. 883390 (H2020-SU-SECU-2019 SERSing Project). BL thanks for the financial support from the Otto Mønsted Foundation.
References


[21] Zhize Li, Dmitry Kovalev, Xin Qian, and Peter Richtárik. Acceleration for compressed gradient descent in distributed and federated optimization, 2020. 3


3972
[27] Konstantin Mishchenko, Grigory Malinin,ky, Sebastian Stich, and Peter Richtárik. ProxSkip: Yes! local gradient steps provably lead to communication acceleration! finally! International Conference on Machine Learning (ICML), 2022. 2