Implicit Gradient Alignment in Distributed and Federated Learning

Yatin Dandi[*1 2], **Luis Barba**[*2], **Martin Jaggi** [2]

[1] IIT Kanpur, India
[2] EPFL, Switzerland
yatind@iitk.ac.in, luis.barbaflores@epfl.ch, martin.jaggi@epfl.ch

Abstract

A major obstacle to achieving global convergence in distributed and federated learning is the misalignment of gradients across clients, or mini-batches due to heterogeneity and stochasticity of the distributed data. In this work, we show that data heterogeneity can in fact be exploited to improve generalization performance through implicit regularization. One way to alleviate the effects of heterogeneity is to encourage the alignment of gradients across different clients throughout training. Our analysis reveals that this goal can be accomplished by utilizing the right optimization method that replicates the implicit regularization effect of SGD, leading to gradient alignment as well as improvements in test accuracies. Since the existence of this regularization in SGD completely relies on the sequential use of different mini-batches during training, it is inherently absent when training with large mini-batches. To obtain the generalization benefits of this regularization while increasing parallelism, we propose a novel GradAlign algorithm that induces the same implicit regularization while allowing the use of arbitrarily large batches in each update. We experimentally validate the benefits of our algorithm in different distributed and federated learning settings.

Introduction

In this paper we focus on sum structured optimization of the form $f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x})$, where each f_i is a different function representing the loss function of either distinct data points, mini-batches or clients. To prove convergence, many assumptions over the f_i's have been studied. For example, one may assume fixed bounds on the variance or dissimilarity of gradients across different f_i. However, in practice, for non-convex optimization problems such as deep neural networks, the dissimilarity across gradients is likely to vary across different values of $\mathbf{x}$. We instead argue that to obtain optimal generalization performance, it is desirable to not only converge to a solution that minimizes the mean loss $f(\mathbf{x})$, but also encourage convergence to regions with reduced gradient dissimilarity.

We propose to achieve convergence to such solutions by aligning the gradients across different f_i. To this end, we introduce a regularizer $r(\mathbf{x}) = \frac{1}{2n}\sum_{i=1}^{n}\|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\|^2$ measuring the variance of gradients across the mini-batches, whose minimization leads to the alignment of different gradients. As demonstrated recently by Smith et al. (2021), stochastic gradient descent (SGD) (Robbins and Monro 1951) already contains an *implicit* regularization effect over gradient descent (GD) corresponding to the minimization of $r(\mathbf{x})$, when comparing updates over an entire epoch. Our analysis applicable to arbitrary sequences of SGD steps further reveals that the optimization trajectory followed by SGD can be approximated through gradient descent on the *surrogate function* $\hat{f}(\mathbf{x}) := f(\mathbf{x}) + \lambda r(\mathbf{x})$ with the strength of the regularization being controlled by the step size. This motivates us to devise new algorithms tailored to implicitly minimize this surrogate function $\hat{f}(\mathbf{x})$.

While control variates-based variance reduction techniques can effectively reduce the variance across different updates (Johnson and Zhang 2013), they do not directly promote variance reduction through the alignment of different f_i's gradients for the current iterate, i.e., such methods do not encourage the decrease of $r(\mathbf{x})$ throughout training. A small variance of gradients across mini-batches, i.e., small $r(\mathbf{x})$, corresponds to the alignment of gradients for different data points. Such alignment can benefit generalization throughout training, since large gradient alignment across datapoints implies that gradient updates on f_i corresponding to empirical risk on a subset of the data may reduce the loss for a much larger number of data points, even outside the training set. A similar observation was recently utilized to improve transfer in error reduction across datapoints in meta-learning (Nichol, Achiam, and Schulman 2018). The gradient alignment in SGD arises due to its sequential nature and the use of small mini-batches, which together induce dependencies between successive updates contributing to the implicit minimization of $r(\mathbf{x})$. These effects, however, decrease as the mini-batch size is increased, since the variance across mini-batches diminishes. This imposes a trade-off between using large mini-batches per update and obtaining gradient alignment and hence better generalization. A similar trade-off has been observed empirically (Keskar et al. 2017; Ma, Bassily, and Belkin 2018; Yin et al. 2018), where using larger mini-batches has been shown to worsen the generalization performance.

We argue that the utilization of gradient alignment to improve generalization can be especially beneficial in distributed and federated learning. In datacenter distributed

[*]These authors contributed equally.

learning (Goyal et al. 2018; Dean et al. 2012), where the primary bottleneck is the computation of gradients instead of communication, (Kairouz and McMahan 2021), it is desirable to exploit the available parallelism to the maximum extent, without losing the benefits of sequential updates on small mini-batches provided by SGD. Our proposed algorithm, GradAlign, achieves this by aligning the gradients across clients through implicit regularization.

In a federated setting (Konečný et al. 2016), where multiple updates for each client are required to reduce the communication cost, data dissimilarity among clients plays an especially important role. One common approach to obtain the regularization benefits of SGD in federated learning is to run SGD on small mini-batches in parallel on separate clients, each with a different subset of the data, while periodically averaging the iterates to obtain global updates (FedAvg (McMahan et al. 2017a)). However, the local nature of optimization in each client, prevents gradient alignment across mini-batches corresponding to different clients. Such gradient alignment across clients is particularly desirable in the presence of data heterogeneity across clients where the convergence of Federated Averaging is hindered due to the phenomenon of "client drift." (Karimireddy et al. 2020), corresponding to the deviation of local updates for each client from the gradient of the global objective. Thus gradient alignment across clients in federated learning, analogous to the gradient alignment across mini-batches in SGD, would not only improve the test accuracy upon convergence, but also minimize the client drift in the presence of heterogeneity. To achieve this, we design a novel algorithm Federated Gradient Alignment (*FedGA*), that replicates the implicit regularization effect of SGD by promoting inter-client gradient alignment. We further derive the existence of a similar regularization effect in a recently proposed algorithm, SCAFFOLD (Karimireddy et al. 2020), albeit without the ability to fine-tune the regularization coefficient. Our main contributions are thus as follows:

1. We design a novel algorithm GradAlign that replicates the regularization effect of a sequence of SGD steps while allowing the use of the entire set of mini-batches for each update.
2. We extend GradAlign to the federated learning setting as FedGA, and derive the existence of the implicit inter-client gradient alignment regularizer $r(\mathbf{x})$ for FedGA as well as for SCAFFOLD.
3. We derive sufficient conditions under which GradAlign causes a decrease in the regularized objective $\hat{f}(\mathbf{x})$.
4. We empirically demonstrate that FedGA achieves better generalization than both FedAvg (McMahan et al. 2017a) and SCAFFOLD (Karimireddy et al. 2020).

Related Work

The relationship between the similarity of gradients and generalization has been explored in several recent works (Chatterjee 2020; Chatterjee and Zielinski 2020; Fort et al. 2020). Our work strengthens the empirical findings in these papers and provides a mechanism to extend the benefits of gradient alignment to distributed and federated learning settings.

The generalization benefits of gradient alignment can also be interpreted through the lens of Neural Tangent Kernel (Jacot, Gabriel, and Hongler 2018): the loss $l(\mathbf{x})$ at a test point $\mathbf{x}$ and the prediction y decreases as $\nabla l(y)^\top \frac{1}{n}\sum_{i=1}^{n} K(x, x_i)\nabla l_i(y_i)$, where x_i, y_i correspond to training points, $K(x, x_i)$ represents the inner product between the output's gradient at test point x and training point x_i and $\nabla l(y), \nabla l_i(y_i)$ denote the gradient of the loss w.r.t the outputs at the corresponding points. Thus, test points with high gradient similarity lead to a larger decrease in their loss. Our work corroborates the recent empirical findings in (Lin et al. 2020a), where the use of extrapolation for large batch SGD leads to significant gains in generalization performance. While Lin et al. (2020a) attributed the improved generalization to smoothening of the landscape due to extrapolation, our analysis and results provide a novel perspective to the benefits of displacement through implicit regularization.

The generalization benefits of SGD have been analyzed through several related perspectives such as Stochastic Differential Equations (SDEs) (Chaudhari and Soatto 2018; Jastrzębski et al. 2018), Bayesian analysis (Smith and Le 2018; M et al. 2017) and flatness of minima (Yao et al. 2018; Keskar et al. 2017), which has been challenged by Dinh et al. (2017). Unlike these works, the implicit regularization perspectives in Barrett and Dherin (2021) and our work directly describe a modified objective upon which gradient flow and gradient descent respectively approximate the updates of SGD. Moreover, our analysis incorporates the effects of finite step sizes, whereas the SDE-based analysis relies on infinitesimal learning rates.

The existence of shared optima in sum structured optimization has previously been analyzed in the context of a strongly convex objective, where the strong growth condition (Schmidt and Roux 2013) implies the existence of a shared optimum and linear convergence for both deterministic and stochastic gradient descent. However, for general non-convex objectives having multiple local minima, it is desirable to encourage convergence to the set of minima to the ones being nearly optimal for all the components f_i without sacrificing the ability to use large amounts of data for each update.

A large number of works have attempted to modify the FedAvg algorithm to improve convergence rates and minimize client drift. Our implicit regularization can easily be incorporated into the various modifications of FedAvg such as FedProx (Li et al. 2020), FedDyn (Acar et al. 2021), FedAvgM (Hsu, Qi, and Brown 2019), FedAdam (Reddi et al. 2021), etc. by introducing the displacements used in our algorithms into the local gradient updates used in these algorithms. Comparisons against FedProx in the Experiments section further verify the utility of our approach as a standalone modification in heterogeneous as well as i.i.d federated learning settings.

Setup

We consider the standard setting of empirical risk minimization with parameters $\mathbf{x}$, represented as a sum

$$\min_{\mathbf{x}\in\mathbb{R}^d}\Big\{f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x})\Big\},$$

where the function f_i denotes the empirical risks on the i_{th} subset of the training data. Here the subsets correspond to different mini-batches, clients, or clients depending on the application. We further define the regularizer

$$r(\mathbf{x}) = \frac{1}{2n}\sum_{i=1}^{n}\|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\|^2.$$

Here $r(\mathbf{x})$ represents $\frac{1}{2}$ times the trace of the covariance matrix for the mini-batch gradients. The gradient of $r(\mathbf{x})$ is then given by:

$$\nabla r(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\left(\nabla^2 f_i(\mathbf{x}) - \nabla^2 f(\mathbf{x})\right)\left(\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\right).$$

Analysis and Proposed Algorithms

A key component in all our subsequent analysis is the expression for the gradient of f_i at a point obtained after applying a displacement $\boldsymbol{v}_{\mathbf{x}}$ to a given point $\mathbf{x}$, i.e., $\nabla f_i(\mathbf{x} + \boldsymbol{v}_{\mathbf{x}})$. By applying Taylor's theorem to each component of ∇f_i, we obtain the following expression (see Appendix):

Lemma 1. *If f_i has Lipschitz Hessian, i.e., $\|\nabla^2 f_i(\mathbf{x}) - \nabla^2 f_i(\mathbf{y})\|_2 \leq \rho\|\mathbf{x} - \mathbf{y}\|$ for some $\rho > 0$, then*

$$\nabla f_i(\mathbf{x} + \boldsymbol{v}_{\mathbf{x}}) = \nabla f_i(\mathbf{x}) + \nabla^2 f_i(\mathbf{x})\boldsymbol{v}_{\mathbf{x}} + \mathcal{O}(\|\boldsymbol{v}_{\mathbf{x}}\|^2). \quad (1)$$

For instance, when $\boldsymbol{v}_{\mathbf{x}} = -\alpha\nabla f_i(\mathbf{x})$, we have:

$$\nabla f_i(\mathbf{x} - \alpha\nabla f_j(\mathbf{x})) = \nabla f_i(\mathbf{x}) - \alpha\nabla^2 f_i(\mathbf{x})\nabla f_j(\mathbf{x}) + \mathcal{O}(\alpha^2) \quad (2)$$

Implicit Gradient Alignment

Suppose that, given two minibatches corresponding to objectives f_i, f_j, we optimize the parameters $\mathbf{x}$ by first updating in the direction of the negative gradient of say f_i and then compute the gradient with respect to the new mini-batch, say f_j, i.e., we utilize $\nabla f_j(\mathbf{x} - \alpha\nabla f_i(\mathbf{x}))$ for the second update. From Equation 2, we observe that, when the order of gradient steps on f_i and f_j, is random, second-order term due to displacement (Lemma 1) in expectation equals $-\frac{\alpha}{2}\left(\nabla^2 f_i(\mathbf{x})\nabla f_j(\mathbf{x}) + \nabla^2 f_j(\mathbf{x})\nabla f_i(\mathbf{x})\right) = -\frac{\alpha}{2}\nabla\left(\nabla f_i(\mathbf{x})^\top \nabla f_j(\mathbf{x})\right)$. **Thus for two given mini-batches, i and j, sequential SGD steps in random order lead to implicit maximization of the inner product of the corresponding gradients. We refer to this phenomenon of alignment of gradients across mini-batches as "Implicit Gradient Alignment".** In the next section, we generalize this argument to arbitrary sequences of minibatches.

SGD over K Sequential Steps

Recall that SGD computes gradients with respect to randomly sampled mini-batches in each round. As explained above, such sequential updates on different mini-batches implicitly align the gradients corresponding to different minibatches. We make this precise by deriving the implicit regularization in SGD for a sequence of K steps under SGD. A similar regularization term was derived by Smith et al. (2021) in the context of backward error analysis for the case of a sequence corresponding to non-overlapping batches covering the entire dataset. They derived a surrogate loss function upon which gradient flow approximates the path followed by SGD when optimizing the original loss function f. Since continuous-time gradient flow is unusable in practice, we instead aim to derive a surrogate loss function $\hat{f}$ where a large batch gradient descent algorithm on this surrogate loss would approximate the path followed by SGD when optimizing f.

Moreover, our analysis applies to arbitrary K and any sampling procedure symmetric w.r.t time, i.e, we only assume that for any sequence of K mini-batches $A = \{a_i\}_{i=1}^K$, the corresponding reverse sequence $A_{-1} = \{a_{K+1-i}\}_{i=1}^K$ has the same probability. This allows us to conveniently evaluate the average effect of SGD for a particular sequence over all possible re-orderings of the sequence. Note that this assumption is valid both when sampling with and without replacement from any arbitrary distribution over mini-batches.

While each gradient update in SGD is an unbiased estimate of the full gradient, the cumulative effect of multiple updates on randomly sampled mini-batches can differ from the minimization of the original objective, as illustrated through Equation (2). To isolate the effect of sequential updates on particular sequences of sampled mini-batches, we compare the steps taken by SGD against the same number of steps using GD on the sample mean of the sequence's objective. We denote the gradient and Hessian for mini-batch a_i by $\nabla f_{a_i}(\mathbf{x})$ and $\nabla^2 f_{a_i}(\mathbf{x})$ respectively while $\nabla f_A(\mathbf{x}), \nabla^2 f_A(\mathbf{x})$ denote the mean gradient and Hessian for the entire sequence A. By applying Lemma 1 to each gradient step, we obtain the following result (proof in the Appendix):

Theorem 1. *Conditioned on the (multi)set of mini-batches in a randomly sampled sequence A of length K, the expected difference between the parameters reached after K steps of SGD using the corresponding mini-batches in A and K steps of GD with step size α on the mean objective $f_A(\mathbf{x}) = \frac{1}{K}\sum_{i=1}^K f_{a_i}(\mathbf{x})$, both starting from the same initial parameters $\mathbf{x}$ is given by:*

$$\mathbb{E}\left[\mathbf{x}_{SGD,A} - \mathbf{x}_{GD,A}\right]$$
$$= -\frac{\alpha^2}{2}\Big(\sum_{i=1}^{K}(\nabla^2 f_{a_i}(\mathbf{x})\left(\nabla f_{a_i}(\mathbf{x}) - \nabla f_A(\mathbf{x})\right)\Big) + \mathcal{O}(\alpha^3) \quad (3)$$

$$= -\frac{K\alpha^2}{2}\nabla r_A(\mathbf{x}) + \mathcal{O}(\alpha^3) \quad (4)$$

where, analogous to beforeSection , we define $r_A(\mathbf{x}) = \frac{1}{2K}\left(\sum_{i=1}^K \|\nabla f_{a_i}(\mathbf{x}) - \nabla f_A(\mathbf{x})\|^2\right)$. For the particular case of a sequence covering an entire epoch, i.e. $K = n$ and sampling without replacement, we recover the implicit regularization over gradient descent derived by Smith et al. (2021). The above results imply that K steps of SGD not only optimize the original objective function analogous to GD, but additionally move the parameters opposite to the gradient of $r_A(\mathbf{x})$ Thus, SGD implicitly minimizes $r_A(\mathbf{x})$ along with the original objective, which leads us to call the latter term an *implicit regularizer*. As we show in the Appendix , the net displacement of SGD in Equation (4) can be approximated by K gradient descent steps on the mean objective regularized by $\frac{\alpha}{2}r_A(\mathbf{x})$. Thus optimizing the regularized objective can

allow us to utilize K times more data for each update, while still approximating the trajectory followed by SGD. This is in contrast to the linear scaling rule discussed in Goyal et al. (2018), which aims to approximate the sequence of K SGD steps with a single GD step with a step size scaled by K. However, such linear scaling only approximates the first-order gradient terms in the sequence, ignoring the implicit gradient alignment. We discuss this further in Appendix , and analyze a linearly scaled approximation of SGD that incorporates implicit gradient alignment. An advantage of approximating SGD using the same number of gradient steps is that it allows the use of larger total batch sizes, whereas linear scaling is only effective for batch sizes much smaller than the total training set size (Shallue et al. 2019).

We observe that the term corresponding to the Hessian for the mini-batch a_i in Equation (3) can be obtained using as the product of the Hessian and the vector $\boldsymbol{v}_{\mathbf{x}} = -\frac{\alpha}{2}\left(\nabla f_A(\mathbf{x}) - \nabla f_{a_i}(\mathbf{x})\right)$. Thus utilizing the right vector for each mini-batch allows us to approximate the regularization effect of SGD. We further observe that Lemma 1 provides an efficient method for obtaining the Hessian-vector product by computing the gradient of f_{a_i} on the point $\mathbf{x}$ displayed by $\boldsymbol{v}_{\mathbf{x}}$, eliminating the time and memory overhead of explicit Hessian-gradient vector computation. Moreover, as we illustrate in section , the displacement-based formulation allows the utilization of gradient alignment in federated settings with multiple (K>1) local steps, without additional communication, computation time, or memory overhead for each of the local steps. In the subsequent sections, we utilize these observations to design algorithms for distributed and federated learning that replicate the regularization effect of SGD while allowing parallelism for the use of arbitrarily large batches, overcoming the generalization failure of traditional large-batch training (Shallue et al. 2019).

Gradient Alignment under Parallel Computations

The analysis in the previous section revealed that sequential updates on a randomly sampled set of mini-batches not only minimize the mean sampled objective but also the variance of gradients across the sampled mini-batches. We aim to replicate this effect while allowing the use of parallelism across mini-batches. Through Equation (3) and Lemma 1, we observed that the source of gradient alignment in the sequential updates for SGD is the evaluation of the gradient of a mini-batch i after an additional displacement in the direction of $-\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right)$. Thus we can replicate the gradient alignment of SGD by utilizing gradients for each mini-batch i computed after an initial displacement $\boldsymbol{v}_i(x) = -\beta\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right)$. This ensures that the vector multiplying $\nabla^2 f_i(\mathbf{x})$ due to displacement (Lemma 1) matches the corresponding vector in the negative gradient of $\beta r(\mathbf{x}) = \beta\frac{1}{2n}\sum_{i=1}^{n}\|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\|^2$. Moreover, unlike SGD, the step size for the displacement β can differ from $\frac{\alpha}{2}$, enabling the fine-tuning of the regularization coefficient. We refer to the resulting Algorithm 1 as GradAlign (GA).

Theorem 2. *The difference between the parameters reached by one step of GradAlign with step size α and displacement β and gradient descent objective starting from the initial parameters $\mathbf{x}$ is given by*

$$\mathbf{x}_{GA} - \mathbf{x}_{GD} = -\frac{\alpha\beta}{2n}\nabla_{\mathbf{x}}\Big(\sum_{i=1}^{n}\|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\|^2\Big) + \mathcal{O}(\alpha\beta^2).$$

Algorithm 1: GradAlign (GA)

Learning rate α, initial model parameters :$\mathbf{x}$
while not done **do**
  $\nabla f(\mathbf{x}) \leftarrow \frac{1}{n}\sum_{i=1}^{n}\nabla f_i(\mathbf{x})$ ▷ Obtain the full gradient by computing the mini-batch gradients in parallel
  for mini-batches i in $[1, \cdots, n]$ in parallel **do**
    Obtain the displacement for the i_{th} minibatch as $\boldsymbol{v}_i \leftarrow (\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x}))$
    $\boldsymbol{g}_i \leftarrow \alpha\nabla f_i(\mathbf{x} - \beta\boldsymbol{v}_i)$ ▷ Obtain gradient after displacement
  end for
  $\mathbf{x} \leftarrow \mathbf{x} - \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{g}_i$
end while

Descent Condition. Since the displacement step size β controls the strength of regularization as well as the error in approximating the gradient of the regularized objective, it is imperative to know if there exists a suitable range of β under which GradAlign causes a decrease in the surrogate objective $\hat{f}(\mathbf{x}) = f(\mathbf{x}) + \beta r(\mathbf{x})$. We prove that unless the algorithm is at a point that is simultaneously critical for $f(\mathbf{x})$ as well as $r(\mathbf{x})$, for sufficiently small step and displacement sizes, each step of FedGA causes a decrease in $\hat{f}(\mathbf{x})$. This lends credence to the use of GradAlign to ensure convergence to shared optima in distributed settings for general smooth non-convex objectives. The proof of the theorem, the justifications for the assumptions, and the consequences for convergence, are provided in the Appendix .

Theorem 3. *Assuming L_1-smoothness of $f(\mathbf{x})$, L_2-smoothness of $r(\mathbf{x})$, and Lipschitzness of Hessians, for $\mathbf{x}^{(t)}$ satisfying at least one of $\nabla f(\mathbf{x}^{(t)}) \neq \mathbf{0}$ or $\nabla r(\mathbf{x}^{(t)}) \neq \mathbf{0}$, $\exists\beta > 0$ such that updating $\mathbf{x}^{(t)}$ using GradAlign with step size $\alpha < \frac{1}{2L_1}$ and displacement β results in updated parameters $\mathbf{x}^{(t+1)}$ satisfying $\hat{f}(\mathbf{x}^{(t+1)}) - \hat{f}(\mathbf{x}^{(t)}) < 0$.*

While the above theorem suggests the possibility of requiring adaptation of the displacement step size with time, in practice, we found that a constant step size is sufficient to achieve significant gains in test accuracy. We hypothesize that this is due to the decrease in variance across mini-batch gradients over time, which balances the effect of the decrease in the gradient norm.

Federated Learning

In the presence of large communication costs across clients, it is desirable to allow multiple local updates for each client before each round of communication. Such an approach is known in the literature as Federated Averaging (FedAvg) (McMahan et al. 2017b) or local SGD, where each round involves $K > 1$ updates on local objectives corresponding to

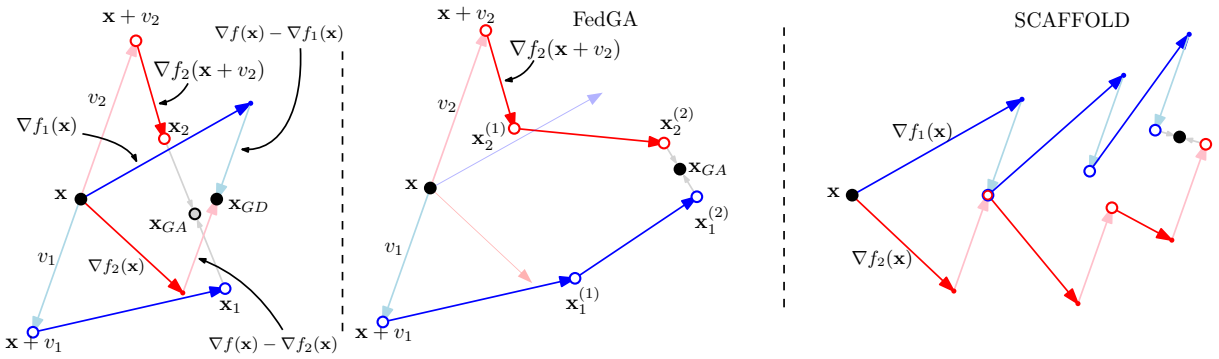

Figure 1: Left: Depiction of one round of GD against one round of GradAlign (equivalent to one round of FedGA with $K = 1$, see Appendix) along with the computation of the displacements $\boldsymbol{v}_i = -\beta(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x}))$. Middle: Schematic depiction of one round of FedGA consisting of $K = 2$ steps. After the initial displacement of $\mathbf{x}$, the algorithm follows K local updates. Right: Schematic depiction of one round of SCAFFOLD where the displacement is applied after each local update.

the loss of randomly sampled clients. In the case of identical data distributions across clients, parts of the generalization benefits of SGD readily appear in FedAvg due to the sequential local update steps within each client (Zinkevich et al. 2010), leading to significant gains in test accuracies over gradient descent on large batches (Lin et al. 2020b; Woodworth et al. 2020). However, as we prove in the appendix , local SGD steps lead to gradient alignment only across mini-batches within the same client. We argue that extending FedAvg to allow implicit gradient alignment *across* clients is desirable for two major reasons. First, similar to SGD and GradAlign, implicit regularization through the minimization of inter-client variance of the gradients is expected to improve generalization performance by encouraging convergence to shared optima across the different clients' objectives. Moreover, gradient alignment across clients crucially minimizes the effects of "client drift", where the presence of the heterogeneity in the data distributions across clients can cause each client's iterates to deviate from the optimization trajectory of the global objective significantly (Karimireddy et al. 2020).

We consider a federated learning setup corresponding to the minimization of the average loss over n clients w.r.t. parameters $\mathbf{x}$. For simplicity, we assume that all the n clients are sampled in each round. We extend the GradAlign algorithm to the federated setting by computing the local updates for each client i using the gradients obtained after an initial additive displacement $\boldsymbol{v}_i(x) = -\beta\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right)$ obtained at the beginning of each round. Since the displacement for each client remains constant throughout a round, the displacement step $\boldsymbol{v}_i$ needs to be applied only once for each client before obtaining the K local updates. Furthermore, since the displacements average to 0 i.e $\sum_{i=1}^n \boldsymbol{v}_i = \sum_{i=1}^n -\beta\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right) = 0$, they don't require being reverted in the end. This is illustrated through Figure 1 and further described in the Appendix . We refer to the resulting Algorithm 2as FedGA (Federated Gradient Alignment).

We assume that, for the k_{th} local update, client i obtains an unbiased stochastic gradient of f_i denoted by $\nabla f_i(.;\zeta_{i,k})$ where $\zeta_{i,k}$ for $k \in [1, \cdots, K]$ are sampled i.i.d such that $f_i(\mathbf{x}) := \mathbb{E}_{\zeta_i}[f_i(\mathbf{x};\zeta_i)]$. The stochasticity in the local updates

Algorithm 2: Federated Gradient Alignment

Input: Learning rate α, initial model parameters $\mathbf{x}$
while not done **do**
  $\nabla f(\mathbf{x}) \leftarrow \frac{1}{n}\sum_{i=1}^n \nabla f_i(\mathbf{x})$ ▷ Update the mean gradient computing $\nabla f_i(\mathbf{x})$ in parallel
  for Client i in $[1, \cdots, n]$ **do**
    Obtain the displacement of the mean gradient as $\boldsymbol{v}_i \leftarrow (\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x}))$
    $\mathbf{x}_i^{(0)} \leftarrow \mathbf{x} - \beta \boldsymbol{v}_i$ ▷ Apply displacement
    for k in $[1, \cdots, K]$ **do**
      $\mathbf{x}_i^{(k)} \leftarrow \mathbf{x}_i^{(k-1)} - \alpha \nabla f_i(\mathbf{x}_i^{(k-1)}; \zeta_{i,k})$
    end for
  end for
  $\mathbf{x} \leftarrow \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i^{(K)}$
end while

allows our algorithm to retain the generalization benefits of local SGD, while additionally aligning the gradients across clients through the use of suitable displacements. Through a derivation similar to Theorem 2 (Appendix), we obtain the following result:

Theorem 4. *The expected difference between the parameters reached by FedGA step size α and displacement β and FedAvg after one round with K local updates per client starting from the initial parameters $\mathbf{x}$ is given by*

$$\mathbb{E}\left[\mathbf{x}_{FedGA} - \mathbf{x}_{FedAvg}\right] = -\frac{\alpha\beta K}{2n}\nabla_{\mathbf{x}}\Big(\sum_{i=1}^n \|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\|^2\Big) + \mathcal{O}(\alpha\beta^2).$$

Scaffold. As noted above, unlike distributed gradient descent with communication at each round, multiple local updates for each client in federated learning can cause the global updates to deviate from the objective's gradient significantly. This motivated Karimireddy et al. (2020) to use control variate based corrections for each client's local updates. Surprisingly, our analysis reveals that the resulting algorithm, SCAFFOLD, not only minimizes the variance of the updates,

	EMNIST IID 10 out of 47	EMNIST heterogeneous 10 out of 47	CIFAR10 IID 10 out of 10
FedGA	$\mathbf{88.66 \pm 0.13}$	$\mathbf{85.95 \pm 0.56}$	$\mathbf{74.34 \pm 0.48}$
SCAFFOLD	88.56 ± 0.12	84.67 ± 0.78	73.89 ± 0.65
FedAvg	88.32 ± 0.06	82.9 ± 0.58	73.1 ± 0.17
FedProx	88.176 ± 0.12	83.197 ± 0.19	73.93 ± 0.38

Table 1: Test Accuracy achieved by FedGA, SCAFFOLD, and FedAvg on EMNIST and CIFAR10. For EMNIST we sample roughly 20% of the clients in each round, while for CIFAR10 100% of the clients are used. For EMNIST we distinguish between the IID and the heterogeneous distributions described in Section .

but also leads to the alignment of the gradients across clients through implicit regularization. This is because, as illustrated in the Appendix , Scaffold and FedGA differ only in that Scaffold directly adds the control variates into the local update while FedGA utilizes them for displacement. This corroborates the empirical improvements in convergence rates and explains the improvements in test accuracies due to SCAFFOLD. The implicit gradient alignment in SCAFFOLD is described through the following result, proved in Appendix :

Theorem 5. *The expected difference between the parameters reached by SCAFFOLD and FedAvg with step size α after one round with K local updates per client starting from the initial parameters $\mathbf{x}$ is given by:*

$$\mathbf{x}_{SCAFFOLD} - \mathbf{x}_{FedAVG} = \\ -\frac{\alpha^2 K(K-1)}{4n}\nabla_{\mathbf{x}}\Big(\sum_{i=1}^{n}\|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\|^2\Big) + \mathcal{O}(\alpha^3). \tag{5}$$

A crucial difference between FedGA and Scaffold is that FedGA allows the ability to utilize a displacement step size β, different from α, enabling finer control over the effect of the regularization term. Moreover, unlike SCAFFOLD, FedGA does not require applying the displacement at each local step, which improves the consistency between consecutive updates as well as the overall efficiency. We describe this in more detail in Appendix .

Experiments

Motivated by the analysis presented in previous sections, we aim to confirm the effectiveness of implicit regularization through a series of experiments on image classification tasks. To this end, we evaluate the effectiveness of GradAlign in achieving improved generalization in the following settings: (1) Federated Learning: Data is distributed on a large number of clients (with different distributions), and only a subset of the clients is sampled to be used in each round. (2) Datacenter distributed learning: Data is distributed (i.i.d.) among the clients, and all clients are used on each round.

Since our primary focus is the quantitative evaluation of generalization performance through test accuracy and test losses, we do not constrain the algorithms to use the same number of local epochs (a local epoch is completed when the entire data of a client has been used, typically in Federated Learning a client can pass more than once trough its data before communicating). Indeed, while increasing the number of local epochs may decrease the number of rounds needed to train, it has no noticeable effect on the maximum test accuracy reached by the algorithm (see Appendix). To further verify the regularization effects of our approach, we provide comparisons of training accuracies in Appendix , showing that the improvements due to gradient alignment are largely in the test rather than training loss. We use a constant learning rate throughout all our experiments to illustrate, as has been done in several federated learning papers (McMahan et al. 2017b; Hsu, Qi, and Brown 2019; Khaled, Mishchenko, and Richtárik 2020; Liu et al. 2020). We also do not use batch normalization or momentum (neither server nor local momentum) in our experiments. Throughout, we report the best results with the hyperparameters obtained through grid search for each of the studied algorithms. For more details, see Appendix . Moreover, each of the reported curves and results is averaged over at least 3 different runs with different random seeds. All experiments were performed using PyTorch on Tesla V100-SXM2 with 32GB of memory.

Recall that both FedGA and SCAFFOLD require one extra round of communication to compute the displacement. **This extra round is included in all our plots and results**, i.e., even with this $2\times$ overhead, FedGA still outperforms the competition. To ensure a fair comparison, for both the settings, we use the following definition of rounds:

Definition of Rounds : In our experimental plots (Figures 2, 9, 10), the "rounds" label in the x-axis denotes the total communication rounds, including the extra communication round for computing the displacements. improve generalization.

Federated Learning

For Federated learning, we use the (balanced) EMNIST dataset (Cohen et al. 2017) consisting of 47 classes distributed among 47 clients, each receiving 2400 training examples. We split the data using two distinct distributions: In the *IID* setting, data is shuffled using a random permutation and then distributed (without overlap) among the 47 clients. In the *heterogeneous* setting, each of the 47 clients is assigned all the data corresponding to a unique label from the 47 classes. This setting has been extensively studied following the work of Hsu, Qi, and Brown (2019). We further include additional results on Natural Language Processing tasks in Appendix and CIFAR-100 in Appendix along with plots of the variance of gradients and test accuracy for EMNIST in Appendix . For EMNIST, we use a (simple) CNN neural network architecture for our experiments with 2 convolutional layers followed by a fully connected layer. The exact description of the network can be found in the Appendix . In each round, we sample 10 out of 47 clients uniformly at random. We compare the performance of four algorithms: FedAvg, Scaffold, FedProx (Li et al. 2020), and FedGA. With approximately 20% of the clients sampled on each round, FedGA achieves the highest Test accuracy and the lowest Test Loss in both settings (see

Figure 10).

IID data Since the data in each client is i.i.d. sampled, using smaller mini-batches for local steps achieves an implicit regularization that promotes gradient alignment within the clients' data (see Section). Scaffold, FedGA, and FedAvg all benefit from this regularization when using smaller mini-batches. On top of that, FedGA and Scaffold promote inter-client gradient alignment as seen in Theorems 4 and 5. Therefore, these algorithms with smaller mini-batches benefit from both inter and intra client gradient alignment. We believe this is the reason why they clearly outperform FedAvg; see Figure 10. Furthermore, FedGA has an additional parameter β that can be used to tune the constant in front of the regularizer (see Theorem 4). Thus, while the implicit regularization term might be present in both Scaffold and FedGA, the fine-tuning of this parameter is crucial for its improvements over Scaffold. Indeed, as seen in Appendix , modifying the constant β has a significant impact on the performance of FedGA. This is a double-edged sword, where on the one hand, β improves generalization, but on the other hand, it can be quite difficult to tune. In fact, β used for the IID and the heterogeneous settings are different, as they depend on the magnitude of the displacement.

Heterogeneous data Federated learning is more challenging if each client has their own data distribution (Hsu, Qi, and Brown 2019), as the gradients become less transferable between clients. Achieving gradient alignment thus has a strong promise to mitigate this problem and to better align the updates on clients with the common objective. Indeed, FedGA achieves a significantly better generalization than FedAvg and SCAFFOLD, the latter ranking in the middle but closer to FedAvg. We also found that increasing the batch size had only a minor impact on training with FedAvg, while it significantly impacts FedGA and SCAFFOLD.

Datacenter distributed learning

We use the CIFAR10 dataset (Krizhevsky, Hinton et al. 2009) consisting of 50000 training examples split among 10 classes, which are then distributed among 10 clients, each receiving 5000 training examples. We split the data using the same IID setting used in Federated Learning. We use a (simple) CNN neural network architecture consisting of 2 convolutional layers followed by 2 fully connected layers. The exact description of the network can be found in the supplementary materials. We study two different settings: In the first, we are interested in maximizing parallelism, i.e., we assume that communication is not the bottleneck, and hence we aim to minimize the total number of updates to reach top accuracy, while communicating once per local gradient computation. In this setting we compare GradAlign (FedGA with $K = 1$) against large-batch SGD and SCAFFOLD (large-batch). The second setting is equivalent to the IID federated learning setting, but with every client sampled in each round.

Sampling all clients Similar to the IID federated learning setting, FedGA obtains the highest accuracy followed by SCAFFOLD and then by FedAvg; see Figure 2. In this setting, even with the overhead of $2\times$ in the number of rounds used by both FedGA and Scaffold, they outperform FedAvg. As in

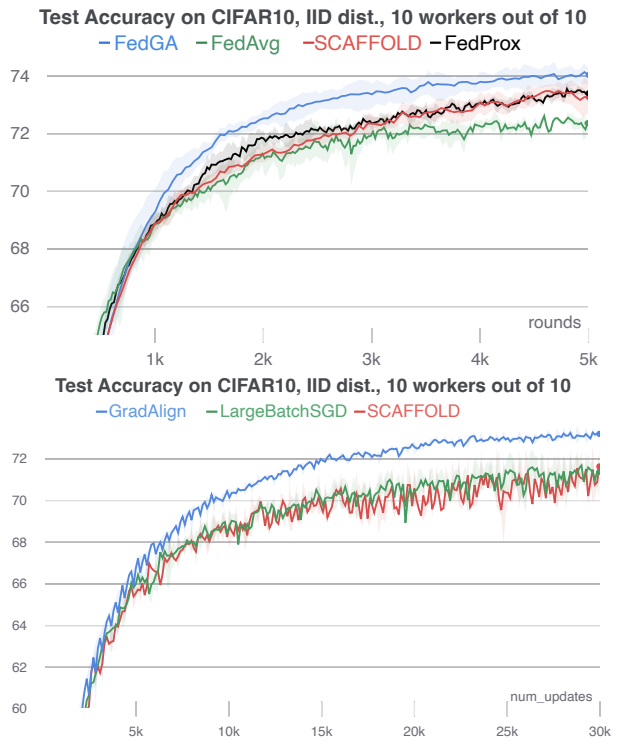

Figure 2: Test accuracy on CIFAR10 for the distributed setting with 100% client sampling per round. Top: In Federated Learning FedGA is not only faster in terms of the number of rounds, it also achieves higher test accuracy. Bottom: The x-axis depicts the number of updates. GradAlign profits from the available parallelism better than Large-Batch SGD and SCAFFOLD.

the federated IID setting, a smaller mini-batch size benefits all algorithms. This is explained by the gradient alignment coming from the use of different mini-batches sequentially during the local updates. Thus both FedGA and SCAFFOLD benefit from inter- and intra-client gradient alignment.

Minimizing number of updates. In this setting, the algorithm to beat is Large-Batch SGD. If communication is fast enough, the main bottleneck is the sequential dependencies between consecutive gradient updates. To increase parallelism, the standard solution is to increase the batch size, but it is known to have an impact on generalization (Keskar et al. 2017; Ma, Bassily, and Belkin 2018; Yin et al. 2018). Our algorithm GradAlign (see Section) allows us to use large mini-batches while retaining the generalization properties of using smaller mini-batches. Indeed, our experiments show that GradAlign noticeably achieves higher Test Accuracy than Large-Batch SGD. Moreover, it converges faster in terms of the number of updates (see Figure 2).

References

Acar, D. A. E.; Zhao, Y.; Matas, R.; Mattina, M.; Whatmough, P.; and Saligrama, V. 2021. Federated Learning Based on

Dynamic Regularization. In *International Conference on Learning Representations*.

Barrett, D.; and Dherin, B. 2021. Implicit Gradient Regularization. In *International Conference on Learning Representations*.

Caldas, S.; Duddu, S. M. K.; Wu, P.; Li, T.; Konečnỳ, J.; McMahan, H. B.; Smith, V.; and Talwalkar, A. 2018. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*.

Chatterjee, S. 2020. Coherent Gradients: An Approach to Understanding Generalization in Gradient Descent-based Optimization. In *International Conference on Learning Representations*.

Chatterjee, S.; and Zielinski, P. 2020. Making Coherence Out of Nothing At All: Measuring the Evolution of Gradient Alignment. arXiv:2008.01217.

Chaudhari, P.; and Soatto, S. 2018. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *International Conference on Learning Representations*.

Cohen, G.; Afshar, S.; Tapson, J.; and Van Schaik, A. 2017. EMNIST: Extending MNIST to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, 2921–2926. IEEE.

Dean, J.; Corrado, G.; Monga, R.; Chen, K.; Devin, M.; Mao, M.; Ranzato, M. a.; Senior, A.; Tucker, P.; Yang, K.; Le, Q.; and Ng, A. 2012. Large Scale Distributed Deep Networks. In Pereira, F.; Burges, C. J. C.; Bottou, L.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Dinh, L.; Pascanu, R.; Bengio, S.; and Bengio, Y. 2017. Sharp Minima Can Generalize For Deep Nets. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 1019–1028. PMLR.

Fort, S.; Nowak, P. K.; Jastrzebski, S.; and Narayanan, S. 2020. Stiffness: A New Perspective on Generalization in Neural Networks. arXiv:1901.09491.

Goyal, P.; Dollár, P.; Girshick, R.; Noordhuis, P.; Wesolowski, L.; Kyrola, A.; Tulloch, A.; Jia, Y.; and He, K. 2018. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. arXiv:1706.02677.

Hsu, T.-M. H.; Qi, H.; and Brown, M. 2019. Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification. arXiv:1909.06335.

Jacot, A.; Gabriel, F.; and Hongler, C. 2018. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Jastrzębski, S.; Kenton, Z.; Arpit, D.; Ballas, N.; Fischer, A.; Bengio, Y.; and Storkey, A. 2018. Three Factors Influencing Minima in SGD. arXiv:1711.04623.

Johnson, R.; and Zhang, T. 2013. Accelerating Stochastic Gradient Descent using Predictive Variance Reduction. In *NIPS*, 315–323.

Kairouz, P.; and McMahan, H. B. 2021. Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning*, 14(1): –.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*.

Karimireddy, S. P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; and Suresh, A. T. 2020. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 5132–5143. PMLR.

Keskar, N. S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; and Tang, P. T. P. 2017. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Khaled, A.; Mishchenko, K.; and Richtárik, P. 2020. Tighter theory for local SGD on identical and heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, 4519–4529. PMLR.

Konečný, J.; McMahan, H. B.; Ramage, D.; and Richtárik, P. 2016. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. arXiv:1610.02527.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. *University of Toronto*.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020. Federated Optimization in Heterogeneous Networks. In Dhillon, I. S.; Papailiopoulos, D. S.; and Sze, V., eds., *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org.

Lin, T.; Kong, L.; Stich, S.; and Jaggi, M. 2020a. Extrapolation for Large-batch Training in Deep Learning. In *ICML - Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 6094–6104. PMLR.

Lin, T.; Stich, S. U.; Patel, K. K.; and Jaggi, M. 2020b. Don't Use Large Mini-batches, Use Local SGD. In *International Conference on Learning Representations*.

Liu, W.; Chen, L.; Chen, Y.; and Zhang, W. 2020. Accelerating federated learning via momentum gradient descent. *IEEE Transactions on Parallel and Distributed Systems*, 31(8): 1754–1766.

M, S.; t; Hoffman, M. D.; and Blei, D. M. 2017. Stochastic Gradient Descent as Approximate Bayesian Inference. *Journal of Machine Learning Research*, 18(134): 1–35.

Ma, S.; Bassily, R.; and Belkin, M. 2018. The Power of Interpolation: Understanding the Effectiveness of SGD in Modern Over-parametrized Learning. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 3325–3334. PMLR.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017a. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A.; and Zhu, J., eds., *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, 1273–1282. Fort Lauderdale, FL, USA: PMLR.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017b. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of AISTATS*, 1273–1282.

Nedic, A. 2020. Distributed Gradient Methods for Convex Machine Learning Problems in Networks: Distributed Optimization. *IEEE Signal Processing Magazine*, 37(3): 92–101.

Nichol, A.; Achiam, J.; and Schulman, J. 2018. On First-Order Meta-Learning Algorithms. arXiv:1803.02999.

Reddi, S. J.; Charles, Z.; Zaheer, M.; Garrett, Z.; Rush, K.; Konečný, J.; Kumar, S.; and McMahan, H. B. 2021. Adaptive Federated Optimization. In *International Conference on Learning Representations*.

Robbins, H.; and Monro, S. 1951. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3): 400–407.

Schmidt, M.; and Roux, N. L. 2013. Fast Convergence of Stochastic Gradient Descent under a Strong Growth Condition. arXiv:1308.6370.

Shallue, C. J.; Lee, J.; Antognini, J.; Sohl-Dickstein, J.; Frostig, R.; and Dahl, G. E. 2019. Measuring the Effects of Data Parallelism on Neural Network Training. *Journal of Machine Learning Research*, 20(112): 1–49.

Smith, S. L.; Dherin, B.; Barrett, D.; and De, S. 2021. On the Origin of Implicit Regularization in Stochastic Gradient Descent. In *International Conference on Learning Representations*.

Smith, S. L.; and Le, Q. V. 2018. A Bayesian Perspective on Generalization and Stochastic Gradient Descent. In *International Conference on Learning Representations*.

Woodworth, B.; Patel, K. K.; Stich, S.; Dai, Z.; Bullins, B.; Mcmahan, B.; Shamir, O.; and Srebro, N. 2020. Is Local SGD Better than Minibatch SGD? In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 10334–10343. PMLR.

Yao, Z.; Gholami, A.; Keutzer, K.; and Mahoney, M. W. 2018. Hessian-Based Analysis of Large Batch Training and Robustness to Adversaries. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, 4954–4964. Red Hook, NY, USA: Curran Associates Inc.

Yin, D.; Pananjady, A.; Lam, M.; Papailiopoulos, D.; Ramchandran, K.; and Bartlett, P. 2018. Gradient Diversity: a Key Ingredient for Scalable Distributed Learning. In Storkey, A.; and Perez-Cruz, F., eds., *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, 1998–2007. PMLR.

Zinkevich, M.; Weimer, M.; Li, L.; and Smola, A. 2010. Parallelized Stochastic Gradient Descent. In Lafferty, J.; Williams, C.; Shawe-Taylor, J.; Zemel, R.; and Culotta, A., eds., *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc.