Implicit Gradient Alignment in Distributed and Federated Learning

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

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. One way to alleviate this problem 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 benefit of our algorithm in different distributed and federated learning settings.

1. 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. In order to achieve 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. 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 restrict the space of solutions by encouraging it to be nearly optimal for the individual components $f_i(\mathbf{x})$. While the existence of such a solution is in itself a strong assumption, modern machine learning involves highly over parametrized models, such as deep neural networks, where a solution nearly optimal for all components f_i is likely to exist (Zhang et al., 2017). 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, and whose minimization leads to the alignment of different gradients. While optimizing such a regularizer through gradient descent requires expensive Hessian-gradient vector computation, 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 datapoints. 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 et al., 2018). The gradient alignment in

[*] Equal contribution [1] IIT Kanpur, India [2] EPFL, Switzerland. Correspondence to: Yatin Dandi <yatin.dandi@epfl.ch>.

This work was presented at the International Workshop on Federated Learning for User Privacy and Data Confidentiality in Conjunction with ICML 2021 (FL-ICML'21). This workshop does not have official proceedings and this paper is non-archival.

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 et al., 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 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, 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., 2017b)). 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 explicitly regularized objective $\hat{f}(\mathbf{x})$.

4. We empirically demonstrate that FedGA achieves better generalization than both FedAvg (McMahan et al., 2017b) and SCAFFOLD (Karimireddy et al., 2020).

2. Related Work

Our work corroborates the recent empirical findings in (Lin et al., 2020a), where the use of extrapolation for large batch SGD lead 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 a number of related perspectives such as Stochastic Differential Equations (SDEs) (Chaudhari and Soatto, 2018; Jastrzębski et al., 2018), Bayesian analysis (Smith and Le, 2018; Mandt 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 the ones being nearly optimal for all the components f_i without sacrificing the ability to use large amounts of data for each update.

3. Setup

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

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

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:

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

4. 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 A.4):

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*

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

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

$$\begin{aligned}\nabla f_i(\mathbf{x} - \alpha\nabla f_j(\mathbf{x})) =& \nabla f_i(\mathbf{x}) - \alpha\nabla^2 f_i(\mathbf{x})\nabla f_j(\mathbf{x}) \\ &+ \mathcal{O}(\alpha^2)\end{aligned} \tag{2}$$

4.1. SGD over K Sequential Steps

Recall that SGD computes gradients with respect to randomly sampled mini-batches in each round. After updating in the direction of the negative gradient of say f_i, we are effectively using the displacement of $-\nabla f_i(\mathbf{x})$ to compute the gradient with respect to the new min-batch, say f_j, i.e., we compute $\nabla f_j(\mathbf{x} - \alpha\nabla f_i(\mathbf{x}))$ for our next 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 3) 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 sequential updates on different functions implicitly maximize the inner product of the corresponding gradients (Nichol et al., 2018). We refer to this phenomenon of alignment of gradients across mini-batches as "implicit gradient alignment". 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 A.4):

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 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:*

$$\begin{aligned}
&\mathbb{E}\left[\mathbf{x}_{SGD,A} - \mathbf{x}_{GD,A}\right] = \\
&-\frac{\alpha^2}{2}\Big(\sum_{i=1}^{K}(\nabla^2 f_{a_i}(\mathbf{x})\left(\nabla f_{a_i}(\mathbf{x}) - \nabla f_A(\mathbf{x})\right)\Big) + \mathcal{O}(\alpha^3) \qquad (3)\\
&= -\frac{\alpha^2}{4}\nabla_{\mathbf{x}}\Big(\sum_{i=1}^{K}\|\nabla f_{a_i}(\mathbf{x}) - \nabla f_A(\mathbf{x})\|^2\Big) + \mathcal{O}(\alpha^3) \qquad (4)\\
&\quad -\frac{K\alpha^2}{2}\nabla r_A(\mathbf{x}) + \mathcal{O}(\alpha^3) \qquad (5)
\end{aligned}$$

where, analogous to Section 3, we define $r_A(\mathbf{x}) = \frac{1}{2K}\big(\sum_{i=1}^{K}\|\nabla f_{a_i}(\mathbf{x}) - \nabla f_A(\mathbf{x})\|^2\big)$. 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 A.4, the net displacement of SGD in Equation (5) 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 A.2, and analyze a linearly scaled approximation of SGD that incorporates implicit gradient alignment. A crucial advantage of approximating SGD using the same number of gradient steps and step size 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).

However, explicit gradient computation of the regularized objective $r_A(\mathbf{x})$ is, however, practically infeasible due to the prohibitively expensive Hessian-gradient vector computations involved. To remedy this, we observe that the term corresponding to the Hessian for the mini-batch a_i in Equation (3) can be obtained using Lemma 1 after computing the gradient of f_{a_i} on the point $\mathbf{x}$ displayed by $\boldsymbol{v}_{\mathbf{x}} = -\frac{\alpha}{2}\left(\nabla f_A(\mathbf{x}) - \nabla f_{a_i}(\mathbf{x})\right)$. Thus utilizing the right displacement for each mini-batch allows us to approximate the regularization effect of SGD without the explicit computation of the regularization term's gradients. In the subsequent sections, we utilize this observation 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).

4.2. 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).

Algorithm 1 GradAlign (GA)

```
Learning rate $\alpha$, 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 $\boldsymbol{v}_i \leftarrow -\beta\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right)$
        $\mathbf{x}_i \leftarrow \mathbf{x} - \alpha\nabla f_i(\mathbf{x} + \boldsymbol{v}_i)$ {Obtain gradient after displacement}
    end for
    $\mathbf{x} \leftarrow \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$
end while
```

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

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

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

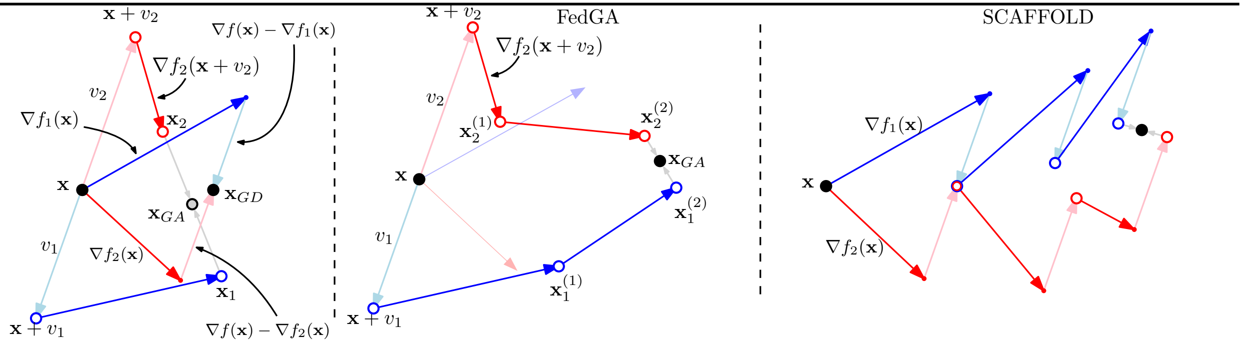

Figure 1. Left: Depiction of one round of GD against one round of GradAlign (equivalent to one round of FedGA with $K = 1$, see Appendix A.5) along with the computation of the displacements $\boldsymbol{v}_i = -\beta(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x}))$. Middle: Schematic depiction of one 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.

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 and the justifications for the assumptions are provided in the Appendix A.1.

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.

4.3. 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., 2017a) or local SGD, where each round involves $K > 1$ updates on local objectives corresponding to 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 A.4, 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 A.5. We refer to the resulting Algorithm 2 as 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 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 A.4), we obtain the following result:

Algorithm 2 Federated Gradient Alignment

```
Input: Learning rate $\alpha$, 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 $\boldsymbol{v}_i \leftarrow -\beta\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right)$
        $\mathbf{x}_i^{(0)} \leftarrow \mathbf{x} + \boldsymbol{v}_i$ {Displacement applied at the beginning}
        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
```

Theorem 4. *The expected difference between the parameters reached by FedGA 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, but also leads to the alignment of the gradients across clients through implicit regularization. This is because, as illustrated in the Appendix A.6, 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 provides an explanation for the improvements in test accuracies due to SCAFFOLD. The implicit gradient alignment in SCAFFOLD is described through the following result, proved in Appendix A.4:

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

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

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.

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 5.1.

	EMNIST	EMNIST	CIFAR10
	IID	heterogeneous	IID
	10 out of 47	10 out of 47	10 out of 10
FedGA	$\mathbf{88.66 \pm 0.1}$	$\mathbf{85.95 \pm 0.56}$	$\mathbf{74.34 \pm 0.48}$
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

5. 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 B.4). We use a constant learning rate throughout all our experiments to illustrate, as has been done in several federated learning papers (McMahan et al., 2017a; Hsu et al., 2019;

Khaled et al., 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 B.2. 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.

5.1. 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 et al. (2019).

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 B.1. In each round, we sample 10 out of 47 clients uniformly at random. We compare the performance of three algorithms: FedAvg, Scaffold, 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 3).

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 4.1). 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 3. 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 B.3, 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.

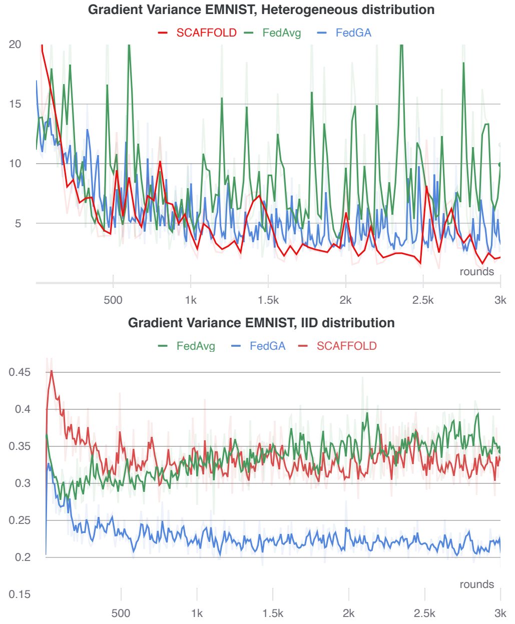

Figure 2. Magnitude of the difference between the global objective gradient and the gradient of the first client's objective, i.e., $\|\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\|$, over 3000 rounds of training. The magnitude of this difference is much smaller in the IID- than in the heterogeneous setting. However, in both cases, FedGA and SCAFFOLD tend to have a smaller difference than FedAvg. Moreover, not only is this quantity lower, but it has a smaller variability.

Heterogeneous data Federated learning is more challenging if each client has their own data distribution (Hsu et al., 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.

5.2. Datacenter distributed learning

We use the CIFAR10 dataset (Krizhevsky 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 number of updates to reach

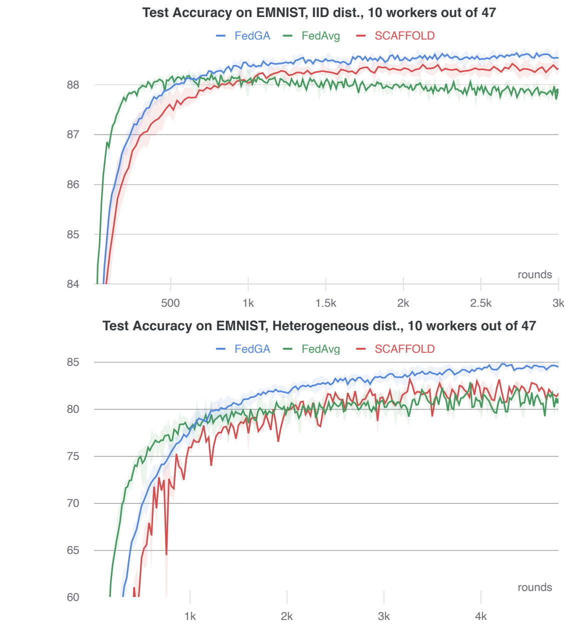

Figure 3. Experiments on the EMNIST dataset using a CNN architecture for the federated learning setting with 47 clients, out of which 10 are uniformly sampled in every round. While FedAvg is faster and can efficiently use more local epochs, both FedGA and SCAFFOLD generalize better. Left: Data is distributed using the IID setting, where data for each client is drawn uniformly at random. Right: Data is distributed heterogeneously, each client having examples of only a single class. This is the most challenging setting for federated algorithms.

top accuracy. The second setting is equivalent to the IID federated learning setting, with the main difference being that every client is sampled in each round. Recall that both FedGA and SCAFFOLD have an overhead of 2× on the number of rounds of communication as they require one extra round to compute the displacement. However, even with this handicap, they outperform FedAvg.

Sampling all clients Similar to the IID federated learning setting, FedGA obtains the highest accuracy followed by SCAFFOLD and then by FedAvg; see Figure 4. In this setting, even with the overhead of 2× in the number of rounds used by both FedGA and Scaffold, they outperform FedAvg. As in the federated IID setting, a smaller mini-batch size benefits all algorithms. We believe this is explained by the gradient alignment coming from the use of different mini-batches sequentially during the local updates. In this way, 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 et al., 2018; Yin et al., 2018). Our algorithm GradAlign (see Section 4.2) 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 number of updates (see Figure 4).

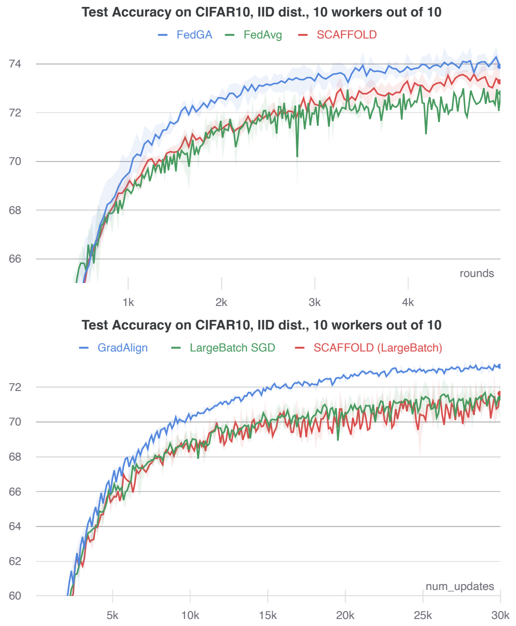

Figure 4. Test accuracy on the CIFAR10 dataset using a CNN architecture for the distributed setting where 100% of the clients are sampled in each round. Left: FedGA is not only faster in terms of the number of rounds, but it also achieves higher test accuracy than its counterparts. Right: The x-axis depicts the number of updates, i.e., the number of times the parameters of the model are modified. With this metric, GradAlign profits from the available parallelism better than Large-Batch SGD and SCAFFOLD.

6. Future Work

Promising directions for future work include designing algorithms with implicit gradient alignment for decentralized and asynchronous learning settings, incorporating optimization schemes such as momentum into gradient alignment, and developing techniques to reduce the communication overhead in FedGA.

References

David Barrett and Benoit Dherin. Implicit gradient regularization. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=3q5IqUrkcF`.

Pratik Chaudhari and Stefano Soatto. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HyWrIgW0W.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc' aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc Le, and Andrew Ng. Large scale distributed deep networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/6aca97005c68f1206823815f66102863-Paper.pdf.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1019–1028. PMLR, 06–11 Aug 2017. URL http://proceedings.mlr.press/v70/dinh17b.html.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour, 2018.

Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd, 2018.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, pages 315–323, 2013. URL http://papers.nips.cc/paper/4937.

Peter Kairouz and H. Brendan McMahan. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1):–, 2021. ISSN 1935-8237. doi: 10.1561/2200000083. URL http://dx.doi.org/10.1561/2200000083.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5132–5143. PMLR, 13–18 Jul 2020.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 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, 2017. URL https://openreview.net/forum?id=H1oyRlYgg.

Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. Tighter theory for local sgd on identical and heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*, pages 4519–4529. PMLR, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Tao Lin, Lingjing Kong, Sebastian Stich, and Martin Jaggi. 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*, pages 6094–6104. PMLR, 13–18 Jul 2020a.

Tao Lin, Sebastian U. Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't use large mini-batches, use local sgd. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=B1eyO1BFPr.

Wei Liu, Li Chen, Yunfei Chen, and Wenyi Zhang. Accelerating federated learning via momentum gradient descent. *IEEE Transactions on Parallel and Distributed Systems*, 31(8):1754–1766, 2020.

Siyuan Ma, Raef Bassily, and Mikhail Belkin. The power of interpolation: Understanding the effectiveness of SGD in modern over-parametrized learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3325–3334. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/ma18a.html.

Stephan Mandt, Matthew D. Hoffman, and David M. Blei. Stochastic gradient descent as approximate bayesian inference. 18(1): 4873–4907, January 2017. ISSN 1532-4435.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of AISTATS*, pages 1273–1282, 2017a.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282, Fort Lauderdale, FL, USA, 20–22 Apr 2017b. PMLR. URL http://proceedings.mlr.press/v54/mcmahan17a.html.

Angelia Nedic. Distributed gradient methods for convex machine learning problems in networks: Distributed optimization. *IEEE Signal Processing Magazine*, 37(3):92–101, 2020. URL https://ieeexplore.ieee.org/abstract/document/9084356.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms, 2018.

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

Mark Schmidt and Nicolas Le Roux. Fast convergence of stochastic gradient descent under a strong growth condition, 2013.

Christopher J. Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E. Dahl. Measuring the effects of data parallelism on neural network training. *Journal of Machine Learning Research*, 20(112):1–49, 2019. URL http://jmlr.org/papers/v20/18-789.html.

Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BJij4yg0Z.

Samuel L Smith, Benoit Dherin, David Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=rq_Qr0c1Hyo.

Blake Woodworth, Kumar Kshitij Patel, Sebastian Stich, Zhen Dai, Brian Bullins, Brendan Mcmahan, Ohad Shamir, and Nathan Srebro. Is local SGD better than minibatch SGD? In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10334–10343. PMLR, 13–18 Jul 2020.

Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W. Mahoney. 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, page 4954–4964, Red Hook, NY, USA, 2018. Curran Associates Inc.

Dong Yin, Ashwin Pananjady, Max Lam, Dimitris Papailiopoulos, Kannan Ramchandran, and Peter Bartlett. Gradient diversity: a key ingredient for scalable distributed learning. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, pages 1998–2007. PMLR, 09–11 Apr 2018. URL http://proceedings.mlr.press/v84/yin18a.html.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Sy8gdB9xx.

Martin A. Zinkevich, Markus Weimer, Alex Smola, and Lihong Li. Parallelized stochastic gradient descent. NIPS'10, page 2595–2603, Red Hook, NY, USA, 2010. Curran Associates Inc.

A. Appendix

A.1. Descent condition

In this section, we provide sufficient conditions for the smoothness of the regularizer $r(\mathbf{x})$ and subsequently prove Theorem 3.

A.1.1. Smoothness of Variance

While the smoothness of the objective $f(\mathbf{x})$ is commonly used to prove the sufficient conditions for descent (decrease of the objective value) in general non-convex settings, the smoothness of the variance regularization term $r(\mathbf{x})$ requires a few additional assumptions as illustrated through the subsequent analysis. The term $\|\nabla r(\mathbf{x}) - \nabla r(\mathbf{y})\|$ can be bounded as follows:

$$
\begin{aligned}
&\|\nabla r(\mathbf{x}) - \nabla r(\mathbf{y})\| \\
&= \|\frac{1}{n}\sum_{i=1}^{n}(\nabla^2 f_i(\mathbf{x}) - \nabla^2 f(\mathbf{x}))(\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x}))) - \frac{1}{n}\sum_{i=1}^{n}(\nabla^2 f_i(\mathbf{y}) - \nabla^2 f(\mathbf{y}))(\nabla f_i(\mathbf{y}) - \nabla f(\mathbf{y})))\| \\
&\leq \|\frac{1}{n}\sum_{i=1}^{n}(\nabla^2 f_i(\mathbf{x}) - \nabla^2 f(\mathbf{x}))\left((\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})) - (\nabla f_i(\mathbf{y}) - \nabla f(\mathbf{y}))\right))\| \\
&+ \|\frac{1}{n}\sum_{i=1}^{n}\left(\left(\nabla^2 f_i(\mathbf{x}) - \nabla^2 f(\mathbf{x})\right) - \left(\nabla^2 f_i(\mathbf{y}) - \nabla^2 f(\mathbf{y})\right)\right)(\nabla f_i(\mathbf{y}) - \nabla f(\mathbf{y})))\|.
\end{aligned}
$$

Thus boundedness and Lipschitzness of $\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})$ and, $\nabla^2 f_i(\mathbf{x})$ are sufficient conditions for the smoothness of $r(\mathbf{x})$. Moreover, since the positivity of $\|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\|$ and the Cauchy–Schwarz inequality further imply that

$$
\|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\| \leq \sum_{j=1}^{n}\|\nabla f_j(\mathbf{x}) - \nabla f(\mathbf{x})\| \leq \sqrt{n}\left(\sum_{j=1}^{n}\|\nabla f_j(\mathbf{x}) - \nabla f(\mathbf{x})\|^2\right)^{\frac{1}{2}},
$$

we note that boundedness of $\|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\|$ also follows from the boundedness of variance.

A.1.2. Theorem 3

Proof. Using the L_1, L_2 smoothness of $f(\mathbf{x}), r(\mathbf{x})$ respectively and $\nabla \hat{f}(\mathbf{x}) = \nabla f(\mathbf{x}) + \beta \nabla r(\mathbf{x})$, we have:

$$
\hat{f}(\mathbf{x}^{(t+1)}) - \hat{f}(\mathbf{x}^{(t)}) \leq \left\langle \mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}, \nabla \hat{f}(\mathbf{x}^{(t)}) \right\rangle + \frac{L_1}{2}\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|^2 + \frac{\beta L_2}{2}\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|^2, \tag{6}
$$

where $\langle \cdot, \cdot \rangle$ denotes the standard inner product in Euclidean space. Following the notation in Section 4, we denote by $\boldsymbol{v}_i$, the displacement $-\beta\big(\nabla f(\mathbf{x}^{(t)}) - \nabla f_i(\mathbf{x}^{(t)})\big)$ corresponding to the i_{th} minibatch. Using the fundamental theorem of calculus applied to each component of ∇f_i, we can express $\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}$ as follows:

$$
\begin{aligned}
\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)} &= -\alpha\left(\frac{1}{n}\sum_{i=1}^{n}\nabla f_i(\mathbf{x}^{(t)} + \boldsymbol{v}_i)\right) \\
= -\alpha\frac{1}{n}\sum_{i=1}^{n}&\left(\nabla f_i(\mathbf{x}^{(t)}) + \nabla^2 f_i(\mathbf{x}^{(t)})\,(\boldsymbol{v}_i) + \int_{z=0}^{1}\left(\nabla^2 f_i(\mathbf{x}^{(t)} + z\boldsymbol{v}_i) - \nabla^2 f_i(\mathbf{x}^{(t)})\right)\boldsymbol{v}_i dz\right) \\
&= -\alpha\nabla \hat{f}(\mathbf{x}^{(t)}) - \alpha\frac{1}{n}\sum_{i=1}^{n}\int_{z=0}^{1}\left(\nabla^2 f_i(\mathbf{x}^{(t)} + z\boldsymbol{v}_i) - \nabla^2 f_i(\mathbf{x}^{(t)})\right)\boldsymbol{v}_i dz. \qquad (7)
\end{aligned}
$$

We now utilize the above expression to bound the terms in equation (6) as follows:

$$
\begin{aligned}
&\left\langle \left(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\right), \nabla \hat{f}(\mathbf{x}^{(t)}) \right\rangle = \frac{1}{\alpha}\left\langle \mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}, -\left(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\right) + \alpha \nabla \hat{f}(\mathbf{x}^{(t)}) + \left(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\right) \right\rangle \\
&= -\frac{1}{\alpha}\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|^2 - \frac{1}{n}\sum_{i=1}^{n}\left\langle \int_{z=0}^{1} \left(\nabla^2 f_i(\mathbf{x}^{(t)} + z\boldsymbol{v}_i) - \nabla^2 f_i(\mathbf{x}^{(t)})\right)\boldsymbol{v}_i dz, \left(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\right)\right\rangle \\
&= -\frac{1}{\alpha}\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|^2 - \frac{1}{n}\sum_{i=1}^{n}\left\langle \int_{z=0}^{1} \left(\nabla^2 f_i(\mathbf{x}^{(t)} + z\boldsymbol{v}_i) - \nabla^2 f_i(\mathbf{x}^{(t)})\right)\boldsymbol{v}_i dz, \left(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\right)\right\rangle \\
&= -\frac{1}{\alpha}\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|^2 - \frac{1}{n}\sum_{i=1}^{n}\int_{z=0}^{1}\left\langle \left(\nabla^2 f_i(\mathbf{x}^{(t)} + z\boldsymbol{v}_i) - \nabla^2 f_i(\mathbf{x}^{(t)})\right)\boldsymbol{v}_i, \left(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\right)\right\rangle dz \\
&\leq -\frac{1}{\alpha}\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|^2 + \frac{1}{n}\sum_{i=1}^{n}\int_{z=0}^{1}\|\left(\nabla^2 f_i(\mathbf{x}^{(t)} + z\boldsymbol{v}_i) - \nabla^2 f_i(\mathbf{x}^{(t)})\right)\boldsymbol{v}_i\|\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\| dz \\
&\leq -\frac{1}{\alpha}\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|^2 + \frac{1}{n}\sum_{i=1}^{n}\int_{z=0}^{1}\rho z\|\boldsymbol{v}_i\|^2\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\| dz \\
&= -\frac{1}{\alpha}\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|^2 + \rho\frac{\beta^2}{2}\left(\sum_{i=1}^{n}\frac{1}{n}\|\nabla f_i(\mathbf{x}^{(t)}) - \nabla f(\mathbf{x}^{(t)})\|^2\right)\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\| \\
&= -\frac{1}{\alpha}\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|^2 + \rho\beta^2 r(\mathbf{x}^{(t)})\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|.
\end{aligned}
$$

Where the last two inequalities follow from Cauchy-Schwartz and ρ-Lipschitzness of $\nabla^2 f_i$ respectively.

We can further use Equation (7) to lower bound $\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|$ as follows:

$$
\begin{aligned}
\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\| &= \|-\alpha\nabla\hat{f}(\mathbf{x}^{(t)}) - \alpha\frac{1}{n}\sum_{i=1}^{n}\int_{z=0}^{1}\left(\nabla^2 f_i(\mathbf{x}^{(t)} + z\boldsymbol{v}_i) - \nabla^2 f_i(\mathbf{x}^{(t)})\right)\boldsymbol{v}_i dz\| \\
&\geq \|\alpha\nabla\hat{f}(\mathbf{x}^{(t)})\| - \frac{1}{n}\sum_{i=1}^{n}\|\alpha\int_{z=0}^{1}\left(\nabla^2 f_i(\mathbf{x}^{(t)} + z\boldsymbol{v}_i) - \nabla^2 f_i(\mathbf{x}^{(t)})\right)\boldsymbol{v}_i dz\| \\
&\geq \|\alpha\nabla\hat{f}(\mathbf{x}^{(t)})\| - \frac{1}{n}\sum_{i=1}^{n}\alpha\int_{z=0}^{1}\|\left(\nabla^2 f_i(\mathbf{x}^{(t)} + z\boldsymbol{v}_i) - \nabla^2 f_i(\mathbf{x}^{(t)})\right)\boldsymbol{v}_i\| dz \\
&\geq \|\alpha\nabla\hat{f}(\mathbf{x}^{(t)})\| - \frac{1}{n}\sum_{i=1}^{n}\alpha\int_{z=0}^{1}\rho\|z\boldsymbol{v}_i\|\|\boldsymbol{v}_i\| dz \\
&= \|\alpha\nabla\hat{f}(\mathbf{x}^{(t)})\| - \alpha\frac{\rho}{2}\left(\sum_{i=1}^{n}\frac{1}{n}\|\boldsymbol{v}_i\|^2\right) \\
&= \|\alpha\nabla\hat{f}(\mathbf{x}^{(t)})\| - \alpha\rho\frac{\beta^2}{2}\left(\sum_{i=1}^{n}\frac{1}{n}\|\nabla f_i(\mathbf{x}^{(t)}) - \nabla f(\mathbf{x}^{(t)})\|^2\right) \\
&= \|\alpha\nabla\hat{f}(\mathbf{x}^{(t)})\| - \alpha\rho\beta^2 r(\mathbf{x}^{(t)}).
\end{aligned}
$$

Substituting in (6), we obtain:

$$
\begin{aligned}
\hat{f}(\mathbf{x}^{(t+1)}) - \hat{f}(\mathbf{x}^{(t)}) \leq& -\frac{1}{\alpha}\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|^2 + \rho\beta^2 r(\mathbf{x}^{(t)})\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\| \\
&+ \frac{L_1}{2}\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|^2 + \frac{\beta L_2}{2}\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|^2.
\end{aligned}
$$

To ensure the negativity of the coefficient for $\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|^2$, we choose $\beta < \frac{L_1}{L_2}$. Then, for $\alpha \leq \frac{1}{2L_1}$, we have:

$$\hat{f}(\mathbf{x}^{(t+1)}) - \hat{f}(\mathbf{x}^{(t)}) \leq -L_1\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|^2 + \rho\beta^2 r(\mathbf{x}^{(t)})\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|$$

Thus a sufficient condition for $\hat{f}(\mathbf{x}^{(t+1)}) - \hat{f}(\mathbf{x}^{(t)}) < 0$ is:

$$-L_1\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|^2 + \rho\beta^2 r(\mathbf{x}^{(t)})\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\| < 0,$$

or equivalently,

$$\beta^2 < \frac{L_1}{\rho}\frac{\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|}{r(\mathbf{x}^{(t)})}.$$

We now consider the following cases:

1. $\|\nabla f(\mathbf{x}^{(t)})\| > 0$: Since $\lim_{\beta\to 0}\frac{L_1}{\rho}\frac{\|\mathbf{x}^{(t+1)}-\mathbf{x}^{(t)}\|}{r(\mathbf{x}^{(t)})} = \frac{L_1\alpha}{\rho}\frac{\|\nabla f(\mathbf{x}^{(t)})\|}{r(\mathbf{x}^{(t)})} > 0$, $\exists\beta'$ such that $-L_1\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|^2 + \rho\beta'^2 r(\mathbf{x}^{(t)})\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\| < 0$.
2. $\|\nabla f(\mathbf{x}^{(t)})\| = 0$ and $\|\nabla r(\mathbf{x}^{(t)})\| > 0$. Then

$$\begin{aligned}\frac{L_1}{\rho}\frac{\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|}{r(\mathbf{x}^{(t)})} &= \frac{L_1}{\rho}\frac{\|-\alpha\beta\nabla r(\mathbf{x}^{(t)}) - \alpha\frac{1}{n}\sum_{i=1}^n\int_{z=0}^1\left(\nabla^2 f_i(\mathbf{x}^{(t)} + z\boldsymbol{v}_i) - \nabla^2 f_i(\mathbf{x}^{(t)})\right)\boldsymbol{v}_i dz\|}{r(\mathbf{x}^{(t)})}\\ &\geq \frac{L_1}{\rho}\frac{\|\alpha\beta\nabla r(\mathbf{x}^{(t)})\| - \|\alpha\frac{1}{n}\sum_{i=1}^n\int_{z=0}^1\left(\nabla^2 f_i(\mathbf{x}^{(t)} + z\boldsymbol{v}_i) - \nabla^2 f_i(\mathbf{x}^{(t)})\right)\boldsymbol{v}_i dz\|}{r(\mathbf{x}^{(t)})}\\ &\geq \frac{\alpha L_1}{\rho}\left(\frac{\beta\|\nabla r(\mathbf{x}^{(t)})\| - \rho\beta^2 r(\mathbf{x}^{(t)})}{r(\mathbf{x}^{(t)})}\right)\end{aligned}$$

   Thus it is sufficient to use β' satisfying:

$$\beta'^2 \leq \frac{\alpha L_1}{\rho}\left(\frac{\beta'\|\nabla r(\mathbf{x}^{(t)})\| - \rho\beta'^2 r(\mathbf{x}^{(t)})}{r(\mathbf{x}^{(t)})}\right),$$

   or equivalently,

$$\beta' \leq \frac{\alpha L_1}{\rho(1+\alpha L_1)}\frac{\|\nabla r(\mathbf{x}^{(t)})\|}{r(\mathbf{x}^{(t)})}.$$

Combining with the assumption, $\beta < \frac{L_1}{L_2}$, we observe that in both cases, to ensure $\hat{f}(\mathbf{x}^{(t+1)}) - \hat{f}(\mathbf{x}^{(t)}) < 0$, it is sufficient to use β satisfying:

$$\beta < \min\{\beta', \frac{L_1}{L_2}\}$$

□

A.2. Linear Scaling

The linear scaling rule (Goyal et al., 2018), when applied to a given (multi)set of K minibatches A, proposes scaling the step size by K, while taking a gradient step on the combined objective $f_A(\mathbf{x}) = \frac{1}{K}\sum_{i=1}^K \nabla f_{a_i}(\mathbf{x})$. As explained by Goyal et al. (2018), a single scaled gradient step approximates K SGD steps on the sequence of minibatches, since $-K\alpha\nabla f_A(\mathbf{x}) = -\sum_{i=1}^K \alpha\nabla f_{a_i}(\mathbf{x})$. Using Lemma 1, we observe that $-K\alpha\nabla f_A(\mathbf{x})$ only incorporates the first order terms in $-\sum_{i=1}^{K} \alpha \nabla f_{a_i}(\mathbf{x})$. To incorporate the second order terms within a single update using the scaled step-size $K\alpha$, we require utilizing the displacement for each minibatch a_i equal to the expectation of the displacement prior to the gradient step on a_i conditioned on the given (multi)set A. Using the symmetry w.r.t time reversal, the expected displacement, upto the first order terms in α, $\mathbb{E}\left[\boldsymbol{v}_{a_i}\right]$ can be expressed as follows:

$$\mathbb{E}\left[\boldsymbol{v}_{a_i}\right] = -\frac{\alpha}{2}\left(\sum_{j\neq i, j=1}^{K} \nabla f_{a_j}(\mathbf{x})\right) + \mathcal{O}(\alpha^2).$$

Thus the single step approximation of SGD, with a linearly scaled step size $K\alpha$ is given by:

$$\mathbf{x} \leftarrow \mathbf{x} - \alpha \sum_{i=1}^{K} \nabla f_{a_i}(\mathbf{x} - \frac{\alpha}{2}\left(\sum_{j\neq i, j=1}^{K} \nabla f_{a_j}(\mathbf{x})\right)).$$

However, a major drawback of the above approximation is that for large K, the increase in step size amplifies the errors in the Taylor's theorem-based approximation for each gradient step. Therefore, to accurately assess the validity and effectiveness of the Taylor's theorem-based implicit regularization, we design algorithms GradAlign and FedGA compatible with small step sizes and arbitrarily large batches.

A.3. Main Assumptions

For Theorems 1,2,4,5, and the starting parameters $\mathbf{x}$ under consideration, we assume that within a neighbourhood of $\mathbf{x}$, the following conditions are satisfied: differentiability of $f_i(\cdot)\ \forall i$, differentiability of $r(\cdot)$ and ρ-Lipschitzness of $\nabla^2 f_i$ for some $\rho > 0$. We use the big-O notation $p(\beta) = \mathcal{O}(q(\beta))$ for a positive scalar β to represent the boundedness of $p(\beta)$ by $q(\beta)$ as $\beta \to 0$ i.e $p(\beta) = \mathcal{O}(q(\beta))$ implies that $\exists \beta' > 0$ such that $|p(\beta)| \leq C|q(\beta)|$ for all $0 \leq \beta \leq \beta'$ for some positive constant C.

A.4. Main Proofs

A.4.1. Lemma 1

Proof. By applying the fundamental theorem of calculus to each component of f_i, we obtain:

$$\nabla f_i(\mathbf{x} + \boldsymbol{v}_{\mathbf{x}}) = \nabla f_i(\mathbf{x}) + \nabla^2 f_i(\mathbf{x})\boldsymbol{v}_{\mathbf{x}} + \int_{z=0}^{1} \left(\nabla^2 f_i(\mathbf{x} + z\boldsymbol{v}_i) - \nabla^2 f_i(\mathbf{x})\right)\boldsymbol{v}_i dz.$$

We bound the norm of the error term as follows:

$$\begin{aligned}\|\nabla f_i(\mathbf{x} + \boldsymbol{v}_{\mathbf{x}}) - \left(\nabla f_i(\mathbf{x}) + \nabla^2 f_i(\mathbf{x})\boldsymbol{v}_{\mathbf{x}}\right)\| &= \|\int_{z=0}^{1} \left(\nabla^2 f_i(\mathbf{x} + z\boldsymbol{v}_i) - \nabla^2 f_i(\mathbf{x})\right)\boldsymbol{v}_i dz\| \\ &\leq \int_{z=0}^{1} \|\left(\nabla^2 f_i(\mathbf{x} + z\boldsymbol{v}_i) - \nabla^2 f_i(\mathbf{x})\right)\boldsymbol{v}_i\| dz \\ &\leq \int_{z=0}^{1} \|\rho\|z\boldsymbol{v}_i\|\boldsymbol{v}_i\| dz \\ &= \frac{\rho}{2}\|\boldsymbol{v}_i\|^2.\end{aligned}$$

Where the last inequality follows from the ρ-Lipschitzness of $\nabla^2 f_i$. □

A.4.2. Theorem 1: SGD over K Sequential Steps

Proof. The distribution over the sequences of K steps, conditioned on the (multi)set $A = \{a_i\}_{i=1}^{K}$ of the sampled minibatches can be described through the corresponding distribution over re-orderings of $\{a_i\}_{i=1}^{K}$. We denote a randomly sampled re-ordering of A as $A' = \{a'_i\}_{i=1}^{K}$, and the corresponding reverse ordering by A'_{-1}. The symmetry w.r.t time-reversal implies that the probability distribution P over A' satisfies $P(A') = P(A'_{-1})$. For a sequence of SGD steps under a given ordering A', we denote by $g_{A',i}(\mathbf{x})$, the i_{th} gradient step corresponding to A' and the starting parameters $\mathbf{x}$ and the displacement from the starting point $\mathbf{x}$ prior to the i_{th} gradient step by $\boldsymbol{v}_{A'}^{(i)}(\mathbf{x})$. Similarly, we denote the i_{th} gradient step and the corresponding displacement for K sequential gradient steps on the mean objective by $g_{GD}^{(i)}(\mathbf{x})$ and $\boldsymbol{v}_{GD}^{(i)}(\mathbf{x})$. Using Lemma 1, we have:

$$\begin{aligned} g_{A'}^{(i)}(\mathbf{x}) &= -\alpha \nabla f_{a'_i}(\mathbf{x} + \boldsymbol{v}_{A'}^{(i)}(\mathbf{x})) = -\alpha \left(\nabla f_{a'_i}(\mathbf{x}) + \nabla^2 f_{a'_i}(\mathbf{x}) \boldsymbol{v}_{A'}^{(i)}(\mathbf{x}) + \mathcal{O}(\|\boldsymbol{v}_{A'}^{(i)}(\mathbf{x})\|^2) \right) \\ &= -\alpha \nabla f_{a'_i}(\mathbf{x}) - \alpha \nabla^2 f_{a'_i}(\mathbf{x}) \boldsymbol{v}_{A'}^{(i)}(\mathbf{x}) + \alpha \mathcal{O}(\|\boldsymbol{v}_{A'}^{(i)}(\mathbf{x})\|^2). \end{aligned} \tag{8}$$

Where $\boldsymbol{v}_{A'}^{(i)}(\mathbf{x}) = \sum_{j=1}^{i-1} g_{A'}^{(j)}(\mathbf{x})$. For $i = 2$, we obtain:

$$\begin{aligned} g_{A'}^{(2)}(\mathbf{x}) &= -\alpha \left(\nabla f_{a'_2}(\mathbf{x}) + \nabla^2 f_{a'_1}(\mathbf{x}) \nabla f_{a'_1}(\mathbf{x}) + \mathcal{O}(\|\alpha \nabla f_{a'_1}(\mathbf{x})\|^2) \right) \\ &= -\alpha \nabla f_{a'_2}(\mathbf{x}) + \alpha^2 \nabla f_{a'_1}(\mathbf{x}) \nabla f_{a'_1}(\mathbf{x}) + \mathcal{O}(\alpha^3). \end{aligned}$$

By applying Equation (8) inductively for $i = 3, \ldots, K$, we obtain:

$$\begin{aligned} \boldsymbol{v}_{A'}^{(i)}(\mathbf{x}) &= \sum_{j=1}^{i-1} g_{A'}^{(j)}(\mathbf{x}) \\ &= \sum_{j=1}^{i-1} -\alpha \nabla f_{a'_j}(\mathbf{x}) - \alpha \nabla^2 f_{a'_j}(\mathbf{x}) \boldsymbol{v}_{A'}^{(j)}(\mathbf{x}) + \mathcal{O}(\alpha^3) \\ &= \sum_{j=1}^{i-1} -\alpha \nabla f_{a'_j}(\mathbf{x}) - \alpha \nabla^2 f_{a'_j}(\mathbf{x}) \left(-\alpha \left(\sum_{l=1}^{j-1} g_{A'}^{(l)}(\mathbf{x}) \right) \right) + \mathcal{O}(\alpha^3) \\ &= \sum_{j=1}^{i-1} -\alpha \nabla f_{a'_j}(\mathbf{x}) + \mathcal{O}(\alpha^2), \end{aligned} \tag{9}$$

and as a result:

$$\begin{aligned} g_{A'}^{(i)}(\mathbf{x}) &= -\alpha \nabla f_{a'_i}(\mathbf{x}) - \alpha \nabla^2 f_{a'_i}(\mathbf{x}) \left(\boldsymbol{v}_{A'}^{(i)}(\mathbf{x}) \right) + \mathcal{O}(\alpha^3) \\ &= -\alpha \nabla f_{a'_i}(\mathbf{x}) + \alpha \nabla^2 f_{a'_1}(\mathbf{x}) \left(\sum_{j=1}^{i-1} \nabla f_{a'_j}(\mathbf{x}) \right) + \mathcal{O}(\alpha^3). \end{aligned} \tag{10}$$

Similarly, for gradient descent on the mean objective, we have:

$$\begin{aligned} \boldsymbol{v}_{GD}^{(i)}(\mathbf{x}) &= \left(\sum_{j=1}^{i-1} g_{GD}^{(j)}(\mathbf{x}) \right) \\ &= \sum_{j=1}^{i-1} -\alpha \nabla f_A(\mathbf{x}) - \alpha \nabla^2 f_A(\mathbf{x}) \boldsymbol{v}_A^{(j)}(\mathbf{x}) + \mathcal{O}(\alpha^3) \\ &= \sum_{j=1}^{i-1} -\alpha \nabla f_A(\mathbf{x}) - \alpha \nabla^2 f_A(\mathbf{x}) \left(-\alpha \left(\sum_{l=1}^{j-1} g_{GD}^{(l)}(\mathbf{x}) \right) \right) + \mathcal{O}(\alpha^3) \\ &= \sum_{j=1}^{i-1} -\alpha \nabla f_A(\mathbf{x}) + \mathcal{O}(\alpha^2). \end{aligned}$$

Therefore, the i_{th} gradient step for gradient descent on the mean objective is given by:

$$\begin{aligned} g_{GD}^{(i)}(\mathbf{x}) &= -\alpha \nabla f_A(\mathbf{x}) - \alpha \nabla^2 f_A(\mathbf{x}) \boldsymbol{v}_{GD}^{(i)}(\mathbf{x}) + \mathcal{O}(\alpha^3) \\ &= -\alpha \nabla f_A(\mathbf{x}) + \alpha^2 \nabla f_A(\mathbf{x}) \left(\sum_{j=1}^{i-1} \nabla f_A(\mathbf{x}) \right) + \mathcal{O}(\alpha^3) \\ &= -\alpha \nabla f_A(\mathbf{x}) + \alpha^2 \nabla f_A(\mathbf{x}) \left((i-1) \nabla f_A(\mathbf{x}) \right) + \mathcal{O}(\alpha^3). \end{aligned}$$

The expected difference between the parameters reached after K steps of SGD using the corresponding mini-batches in A and K steps of GD on the mean objective $f_A(\mathbf{x}) = \frac{1}{K}\sum_{i=1}^{K} f_{a_i}(\mathbf{x})$ with initial parameters $\mathbf{x}$ is then given by:

$$
\begin{aligned}
&\mathbb{E}_{A'}\left[\sum_{i=1}^{K}(g_{A'}^{(i)}(\mathbf{x}) - g_{GD}^{(i)}(\mathbf{x}))\right] \\
=&\mathbb{E}_{A'}\left[-\alpha\nabla f_{a'_i}(\mathbf{x}) + \alpha\nabla^2 f_{a'_1}(\mathbf{x})\left(\sum_{j=1}^{i-1}\nabla f_{a'_j}(\mathbf{x})\right) + \mathcal{O}(\alpha^3)\right] \\
+&\mathbb{E}_{A'}\left[\sum_{i=1}^{K}\alpha\nabla f_A(\mathbf{x}) + \alpha^2\nabla f_A(\mathbf{x})\left((i-1)\nabla f_A(\mathbf{x})\right) + \mathcal{O}(\alpha^3)\right] \\
=&\sum_{A'\in S_K} P(A')\left(\alpha^2(\sum_{i=1}^{K}\sum_{j=1}^{i-1}\nabla^2 f_{a'_i}(\mathbf{x})\nabla_{a'_j} f(\mathbf{x})) - \alpha\frac{K(K-1)}{2}\nabla^2 f(\mathbf{x})_A\nabla f_A(\mathbf{x}) + \mathcal{O}(\alpha^3)\right) \\
=&\frac{1}{2}\left(\sum_{A\in[m]^K} P(A')\left(\alpha^2(\sum_{i=1}^{K}\sum_{j=1}^{i-1}\nabla^2 f_{a'_i}(\mathbf{x})\nabla_{a'_j} f(\mathbf{x})) - \alpha\frac{K(K-1)}{2}\nabla^2 f(\mathbf{x})_A\nabla f_A(\mathbf{x}) + \mathcal{O}(\alpha^3)\right)\right. \\
&\left. + \sum_{A\in[m]^K} P(A'_{-1})\left(\alpha^2(\sum_{i=1}^{K}\sum_{j=1}^{i-1}\nabla^2_{a'_{K+1-i}} f(\mathbf{x})\nabla_{a_{K+1-j}} f(\mathbf{x})) - \alpha\frac{K(K-1)}{2}\nabla^2 f(\mathbf{x})_A\nabla f_A(\mathbf{x}) + \mathcal{O}(\alpha^3)\right)\right) \\
=&\sum_{A'\in S_K} P(A')\left(\alpha^2(\sum_{i=1}^{K}\sum_{j=1}^{K}\nabla^2 f_{a'_i}(\mathbf{x})\nabla_{a'_j} f(\mathbf{x})) - \frac{\alpha^2}{2}(\sum_{i=1}^{K}\sum_{j=1}^{K}\nabla^2 f_{a'_i}(\mathbf{x})\nabla f_{a'_j}(\mathbf{x})) + \alpha^2\frac{K}{2}\nabla^2 f_A(\mathbf{x})\nabla f_A(\mathbf{x}) + \mathcal{O}(\alpha^3)\right).
\end{aligned}
$$

Now, since each A' corresponds to a re-ordering of the given (multi)set A, the above expression simplifies to:

$$
\begin{aligned}
&\mathbb{E}_{A'}\left[\sum_{i=1}^{K}(g_{A'}^{(i)}(\mathbf{x}) - g_{GD}^{(i)}(\mathbf{x}))\right] \\
=&\alpha^2\frac{K^2}{2}\nabla^2 f_A(\mathbf{x})\nabla f_A(\mathbf{x}) - \frac{\alpha^2}{2}\sum_{i=1}^{K}\nabla^2 f_{a_i}(\mathbf{x})\nabla f_{a_i}(\mathbf{x}) - \alpha^2\frac{K^2}{2}\nabla^2 f_A(\mathbf{x})\nabla f_A(\mathbf{x}) + \alpha^2\frac{K}{2}\nabla^2 f_A(\mathbf{x})\nabla f_A(\mathbf{x}) + \mathcal{O}(\alpha^3) \\
=&-\mathbb{E}_{A'}\left[\frac{\alpha^2}{2}(\sum_{i=1}^{K}(\nabla^2 f_{a_i}(\mathbf{x})\nabla f_{a_i}(\mathbf{x}) - \nabla^2 f_{a_i}(\mathbf{x})\nabla f_A(\mathbf{x}) - \nabla^2 f_A(\mathbf{x})\nabla f_{a_i}(\mathbf{x}) + \nabla^2 f_A(\mathbf{x})\nabla f(\mathbf{x})))\right] + \mathcal{O}(\alpha^3) \\
=&-\frac{\alpha^2}{4}(\sum_{i=1}^{K}(\nabla^2 f_{a_i}(\mathbf{x}) - \nabla^2 f_A(\mathbf{x}))(\nabla f_{a_i}(\mathbf{x}) - \nabla f_A(\mathbf{x}))) + \mathcal{O}(\alpha^3) \\
=&-\frac{\alpha^2}{4}(\sum_{i=1}^{K}(\nabla^2 f_{a_i}(\mathbf{x}) - \nabla^2 f_A(\mathbf{x}))(\nabla f_{a_i}(\mathbf{x}) - \nabla f_A(\mathbf{x}))) + \mathcal{O}(\alpha^3) \\
=&-\frac{\alpha^2}{4}\nabla_{\mathbf{x}}\Big(\sum_{i=1}^{K}\|\nabla f_{a_i}(\mathbf{x}) - \nabla f_A(\mathbf{x})\|^2\Big) = -\frac{K\alpha^2}{2}\nabla r_A(\mathbf{x}) + \mathcal{O}(\alpha^3).
\end{aligned}
$$

□

A.4.3. Approximating K SGD steps with K GD steps on the Regularized Objective

In this section, we prove that for a given sequence A of K minibatches, the expected difference between K updates using SGD and gradient descent on the mean objective (Equation (5)) can be approximated through gradient descent on the regularized mean objective $\hat{f}_A(\mathbf{x}) = f_A(\mathbf{x}) + \frac{\alpha}{2}r_A(\mathbf{x})$ Similar to the proof for Theorem 1, for a given sequence A, we denote the i_{th} gradient step and the displacement from $\mathbf{x}$ prior to it under the mean objective $f_A(\mathbf{x})$ and the regularized mean objective $\hat{f}_A(\mathbf{x})$ by $g^{(i)}_{GD}(\mathbf{x}), \boldsymbol{v}^{(i)}_{GD}(\mathbf{x})$ and $\hat{g}^{(i)}_{GD}(\mathbf{x}), \hat{\boldsymbol{v}}^{(i)}_{GD}(\mathbf{x})$ respectively.

We have:

$$\begin{aligned}
\hat{g}^{(i)}_{GD} &= -\alpha \nabla \hat{f}_A(\mathbf{x} + \hat{\boldsymbol{v}}^{(i)}_{GD}(\mathbf{x})) = -\alpha \left(\nabla f_A(\mathbf{x} + \hat{\boldsymbol{v}}^{(i)}_{GD}(\mathbf{x})) + \frac{\alpha}{2} \nabla r(\mathbf{x} + \hat{\boldsymbol{v}}^{(i)}_{GD}(\mathbf{x})) \right) \\
&= -\alpha \left(\nabla f_A(\mathbf{x}) + \nabla^2 f_A(\mathbf{x}) \hat{\boldsymbol{v}}^{(i)}_{GD}(\mathbf{x}) + \mathcal{O}(\|\boldsymbol{v}^{\hat{(i)}}_{GD}\|^2) + \frac{\alpha}{2} \nabla r(\mathbf{x}) - \frac{\alpha}{2} \nabla^2 r(\mathbf{x}) \hat{\boldsymbol{v}}^{(i)}_{GD}(\mathbf{x}) + \mathcal{O}(\|\hat{\boldsymbol{v}}^{(i)}_{GD}\|^2) \right)
\end{aligned} \tag{11}$$

For $i = 2$, we get:

$$\begin{aligned}
\hat{g}^{(2)}_{GD} &= -\alpha \left(\nabla f_A(\mathbf{x}) - \alpha \nabla^2 f_A(\mathbf{x}) \left(\nabla f_A(\mathbf{x}) + \frac{\alpha}{2} \nabla r_A(\mathbf{x}) \right) + \mathcal{O}(\alpha^2) \right) \\
&\quad - \alpha \left(\frac{\alpha}{2} \nabla r(\mathbf{x}) - \frac{\alpha^2}{2} \nabla^2 r(\mathbf{x}) \left(\nabla f_A(\mathbf{x}) + \frac{\alpha}{2} \nabla r_A(\mathbf{x}) \right) + \mathcal{O}(\alpha^2) \right) \\
&\quad - \alpha \nabla f_A(\mathbf{x}) + \alpha^2 \nabla^2 f_A(\mathbf{x}) \nabla f_A(\mathbf{x})) - \frac{\alpha^2}{2} \nabla r(\mathbf{x}) + \mathcal{O}(\alpha^3).
\end{aligned}$$

By inductively applying Equation (11) for $i = 3, \cdots, K$, we obtain:

$$\begin{aligned}
\hat{\boldsymbol{v}}^{(i)}_{GD}(\mathbf{x}) &= \left(\sum_{j=1}^{i-1} \hat{g}^{(j)}_{GD}(\mathbf{x}) \right) \\
&= \sum_{j=1}^{i-1} -\alpha \nabla f_A(\mathbf{x}) - \alpha \nabla^2 f_A(\mathbf{x}) \boldsymbol{v}^{(j)}_A(\mathbf{x}) + \mathcal{O}(\alpha^3) \\
&= \sum_{j=1}^{i-1} -\alpha \nabla f_A(\mathbf{x}) - \alpha \nabla^2 f_A(\mathbf{x}) \left(-\alpha \left(\sum_{l=1}^{j-1} g^{(l)}_{GD}(\mathbf{x}) \right) \right) + \mathcal{O}(\alpha^3) \\
&= \sum_{j=1}^{i-1} -\alpha \nabla f_A(\mathbf{x}) + \mathcal{O}(\alpha^2),
\end{aligned}$$

and therefore,

$$\begin{aligned}
\hat{g}^{(i)}_{GD}(\mathbf{x}) &= -\alpha \nabla \hat{f}_A(\mathbf{x} + \hat{\boldsymbol{v}}^{(i)}_{GD}(\mathbf{x})) \\
&= -\alpha \nabla f_A(\mathbf{x}) - \alpha \nabla^2 f_A(\mathbf{x}) \hat{\boldsymbol{v}}^{(i)}_{GD}(\mathbf{x}) - \frac{\alpha^2}{2} \nabla r(\mathbf{x}) - \frac{\alpha^2}{2} \nabla^2 r(\mathbf{x}) \hat{\boldsymbol{v}}^{(i)}_{GD}(\mathbf{x}) + \mathcal{O}(\alpha^3) \\
&= -\alpha \nabla f_A(\mathbf{x}) + \alpha^2 \nabla^2 f_A(\mathbf{x}) \left(\sum_{j=1}^{i-1} \nabla f_A(\mathbf{x}) \right) - \frac{\alpha^2}{2} \nabla r(\mathbf{x}) + \mathcal{O}(\alpha^3).
\end{aligned}$$

Thus the difference between the parameters reached by K gradient descent steps on the regularized mean objective and the mean objective, denoted by $\hat{\mathbf{x}}_{GD,A}$ and $\mathbf{x}_{GD,A}$ respectively is given by:

$$\begin{aligned}
&\hat{\mathbf{x}}_{GD,A} - \mathbf{x}_{GD,A} = \sum_{i=1}^{K} \left(\hat{g}^{(i)}_{GD} - g^{(i)}_{GD} \right) \\
&= \sum_{i=1}^{K} \left(-\alpha \nabla f_A(\mathbf{x}) + \alpha^2 \nabla^2 f_A(\mathbf{x}) \left(\sum_{j=1}^{i-1} \nabla f_A(\mathbf{x}) \right) - \frac{\alpha^2}{2} \nabla r(\mathbf{x}) + \mathcal{O}(\alpha^3) \right) \\
&- \left(-\alpha \nabla f_A(\mathbf{x}) + \alpha^2 \nabla f_A(\mathbf{x}) \left((i-1) \nabla f_A(\mathbf{x}) \right) + \mathcal{O}(\alpha^3) \right) \\
&= \sum_{i=1}^{K} -\frac{\alpha^2}{2} \nabla r(\mathbf{x}) + \mathcal{O}(\alpha^3) \\
&= -\frac{\alpha^2 K}{2} \nabla r(\mathbf{x}) + \mathcal{O}(\alpha^3).
\end{aligned}$$

A.4.4. THEOREM 2: GRADALIGN

Proof. Using Lemma 1, the gradient step $g_i(\mathbf{x})$ for the i_{th} mini-batch obtained after displacement through $\boldsymbol{v}_i(\mathbf{x}) = -\beta\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right)$ with starting parameters $\mathbf{x}$, can be expressed as:

$$\begin{aligned} g_i &= -\alpha\nabla f_i(\mathbf{x} + \boldsymbol{v}_i(\mathbf{x})) = -\alpha\left(\nabla f_i(\mathbf{x}) + \nabla^2 f_i(\mathbf{x})\boldsymbol{v}_i(\mathbf{x}) + \mathcal{O}(\|\boldsymbol{v}_i\|^2)\right) \\ &= -\alpha\left(\nabla f_i(\mathbf{x}) - \beta\nabla^2 f_i(\mathbf{x})\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right) + \mathcal{O}(\|\beta\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right)\|^2)\right) \\ &= -\alpha\nabla f_i(\mathbf{x}) + \alpha\beta\nabla^2 f_i(\mathbf{x})\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right) + \mathcal{O}(\alpha\beta^2). \end{aligned} \tag{12}$$

Therefore, we obtain:

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

□

A.4.5. THEOREM 4: FEDGA

Proof. Analogous to the proof for 1, we denote the local displacement for client i from the starting point $\mathbf{x}$ prior to the k_{th} step for FedAvg and FedGA by $\boldsymbol{v}_{i,FedAvg}^{(k)}, \boldsymbol{v}_{i,FedGA}^{(k)}$ respectively and the corresponding k_{th} gradient step by $g_{i,FedAvg}^{(k)}(\mathbf{x}), g_{i,FedGA}^{(k)}(\mathbf{x})$ respectively. For a given client i, the K local updates in FedAvg are equivalent to a sequnce of SGD updates on the sampled K minibatches. Thus, using Equations (9),(10), we have:

$$\begin{aligned} \boldsymbol{v}_{i,FedAvg}^{(k)}(\mathbf{x}) &= \sum_{j=1}^{k-1} g_{i,FedAvg}^{(j)}(\mathbf{x}) \\ &= \sum_{j=1}^{k-1} -\alpha\nabla f_i(\mathbf{x};\zeta_{i,j}) - \alpha\nabla^2 f_i(\mathbf{x};\zeta_{i,j})\boldsymbol{v}_i^{(j)}(\mathbf{x}) + \mathcal{O}(\alpha^3) \\ &= \sum_{j=1}^{k-1} -\alpha\nabla f_i(\mathbf{x};\zeta_{i,j}) - \alpha\nabla^2 f_i(\mathbf{x};\zeta_{i,j})\left(\sum_{l=1}^{j-1} g_{i,FedAvg}^{(l)}(\mathbf{x})\right) + \mathcal{O}(\alpha^3) \\ &= \sum_{j=1}^{i-1} -\alpha\nabla f_i(\mathbf{x};\zeta_{i,j}) + \mathcal{O}(\alpha^2), \end{aligned}$$

and

$$\begin{aligned} g_{i,FedAvg}^{(k)}(\mathbf{x}) &= -\alpha\nabla f_i(\mathbf{x};\zeta_{i,k}) - \alpha\nabla^2 f_i(\mathbf{x})\left(\boldsymbol{v}_{i,FedAvg}^{(k)}(\mathbf{x})\right) + \mathcal{O}(\alpha^3) \\ &= -\alpha\nabla f_i(\mathbf{x};\zeta_{i,k}) + \alpha\nabla^2 f_i(\mathbf{x};\zeta_{i,k})\left(\sum_{j=1}^{k-1}\nabla f_i(\mathbf{x};\zeta_{i,j})\right) + \mathcal{O}(\alpha^3). \end{aligned}$$

Whereas for FedGA, we include an additional gradient alignment displacement $-\beta\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right)$ for each local update to obtain:

$$\begin{aligned}
\boldsymbol{v}_{i,FedGA}^{(k)}(\mathbf{x}) &= -\beta\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right) + \sum_{j=1}^{k-1} g_{i,FedGA}^{(j)}(\mathbf{x}) \\
&= -\beta\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right) - \sum_{j=1}^{k-1} \alpha \nabla f_i(\mathbf{x}; \zeta_{i,j}) + \mathcal{O}(\alpha^2) + \mathcal{O}(\alpha\beta)
\end{aligned}$$

and

$$\begin{aligned}
&g_{i,FedGA}^{(k)}(\mathbf{x}) = -\alpha \nabla f_i(\mathbf{x}; \zeta_{i,k}) - \alpha \nabla^2 f_i(\mathbf{x}) \left(\boldsymbol{v}_{i,FedGA}^{(k)}(\mathbf{x})\right) + \mathcal{O}(\alpha^3) + \mathcal{O}(\alpha\beta^2) \\
&= -\alpha \nabla f_i(\mathbf{x}; \zeta_{i,k}) - \alpha \nabla^2 f_i(\mathbf{x}; \zeta_{i,k}) \left(-\beta\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right) - \sum_{j=1}^{k-1} \alpha \nabla f_i(\mathbf{x}; \zeta_{i,j})\right) + \mathcal{O}(\alpha^3) + \mathcal{O}(\alpha\beta^2).
\end{aligned}$$

The expected difference between the parameters obtained after one round of FedGA and FedAvg is then given by:

$$\mathbb{E}\left[\mathbf{x}_{FedGA} - \mathbf{x}_{FedAVG}\right] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} g_{i,FedGA}^{(k)}(\mathbf{x}) - \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} g_{i,FedAvg}^{(k)}(\mathbf{x})\right].$$

Where the expectation is over random variables $\{\zeta_{i,k}\}_{k=1}^{K}$ controlling the stochasticity of the local updates for each client i. Linearity of expectation allows us to couple the local updates for FedGA and FedAvg by using the same $\zeta_{i,k}$ for both the algorithms for each client i and update k. We obtain:

$$\begin{aligned}
&\mathbb{E}\left[\mathbf{x}_{FedGA} - \mathbf{x}_{FedAVG}\right] = -\mathbb{E}\left[\frac{\alpha}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} \nabla f_i(\mathbf{x}; \zeta_{i,l})\right] \\
&+ \mathbb{E}\left[\frac{\alpha}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} \nabla^2 f_i(\mathbf{x}; \zeta_{i,k})\left(\beta\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right) + \alpha\sum_{l=1}^{k-1} \nabla f_i(\mathbf{x}; \zeta_{i,l})\right)\right] + \mathcal{O}(\alpha\beta^2) \\
&- \mathbb{E}\left[(-\frac{\alpha}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} \nabla f_i(\mathbf{x}; \zeta_{i,k}) + \frac{\alpha}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} \nabla^2 f_i(\mathbf{x}; \zeta_{i,k})(\alpha\sum_{l=1}^{k-1} \nabla f_i(\mathbf{x}; \zeta_{i,l}))\right] + \mathcal{O}(\alpha\beta^2) \\
&= Ea\frac{\alpha\beta}{n}\sum_{i=1}^{n}\left(\sum_{k=1}^{K} \nabla^2 f_i(\mathbf{x}; \zeta_{i,k})\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right)\right) + \mathcal{O}(\alpha\beta^2) \\
=& -\frac{\alpha\beta K}{n}(\sum_{i=1}^{n}(\nabla^2 f_i(\mathbf{x})\nabla f_i(\mathbf{x}) - \nabla^2 f_i(\mathbf{x})\nabla f(\mathbf{x}) - \nabla^2 f(\mathbf{x})\nabla f_i(\mathbf{x}) + \nabla^2 f(\mathbf{x})\nabla f(\mathbf{x}))) + \mathcal{O}(\alpha\beta^2) \\
=& -\frac{\alpha\beta K}{n}(\sum_{i=1}^{n}(\nabla^2 f_i(\mathbf{x}) - \nabla^2 f(\mathbf{x}))(\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})))) + \mathcal{O}(\alpha\beta^2) \\
=& -\frac{\alpha\beta K}{2n}\nabla_{\mathbf{x}}\left(\sum_{i=1}^{n}\|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\|^2\right) + \mathcal{O}(\alpha\beta^2)
\end{aligned}$$

□

A.5. Implicit cancellation in FedGA

In this section, we describe the equivalence between using the displacement $-\beta\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right)$ only once at the beginning of each round for each client i in FedGA,and using the same displacement, but on each of the K local updates. The former version of the algorithm is described in Algorithm 2 while the latter is described below in Algorithm 3.

Algorithm 3 Federated Gradient Alignment (FedGA)

```
 1: learning rate $\alpha$
 2: Initial model parameters :$\mathbf{x}$
 3: Mean of initial gradients for clients in $[n]$: $\nabla f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x})$
 4: while not done do
 5:     $\nabla f(\mathbf{x}) \leftarrow \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x})$ {Update the mean gradient}
 6:     for Client $i$ in $[1, \cdots, n]$ do
 7:         Obtain the displacement of the mean gradient as $\boldsymbol{v}_i \leftarrow -\beta\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right)$
 8:         $\mathbf{x}_i^{(0)} \leftarrow \mathbf{x}$
 9:         for $k$ in $[1, \cdots, K]$ do
10:             $\mathbf{x}_i^{(k)} \leftarrow \mathbf{x}_i^{(k-1)} - \alpha\nabla f_i(\mathbf{x}_i^{(k-1)} + \boldsymbol{v}_i; \zeta_{i,k})$ {Obtain gradient after displacement}
11:         end for
12:     end for
13:     $\mathbf{x} \leftarrow \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i^{(K)}$
14: end while
```

Notice that to compute $\mathbf{x}_i^{(k)}$ in line 3 of Algorithm 3 we could instead follow these 3 steps: (1) $\mathbf{x}_i^{(k)} \leftarrow \mathbf{x}_i^{(k)} + \boldsymbol{v}_i$, then (2) $\mathbf{x}_i^{(k)} \leftarrow \mathbf{x}_i^{(k)} - \alpha\nabla f_i(\mathbf{x}_i^{(k)})$, and finally $\mathbf{x}_i^{(k)} \leftarrow \mathbf{x}_i^{(k)} - \boldsymbol{v}_i$ to arrive at the same point obtained in line 3. Since $\boldsymbol{v}_i$ remains constant throughout the K steps in one round, the displacement in step (1) and step (3) cancel between consecutive local updates. Thus, we are left with the first and last displacement only. Furthermore, since the displacements average to 0 i.e $\sum_{i=1}^{n} \boldsymbol{v}_i = \sum_{i=1}^{n} -\beta\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right) = 0$, we do not need to take the final step either, and hence we are left with the formulation of Algorithm 2.

A.6. SCAFFOLD

The full SCAFFOLD algorithm (Karimireddy et al., 2020) is described in Algorithm 4. For simplicity, we assume that the displacement we use is computed only among the sampled clients.

Algorithm 4 Scaffold

```
learning rate $\alpha$
Initial model parameters :$\mathbf{x}$
while not done do
    $\nabla f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\mathbf{x})$ {Compute each $\nabla f_i(\mathbf{x};)$ in parallel}
    for Client $i$ in $[1, \cdots, n]$ do
        $\mathbf{x}_i^{(0)} \leftarrow \mathbf{x}$
        for $k$ in $[1, \cdots, K]$ do
            $\mathbf{x}_i^{(k)} \leftarrow \mathbf{x}_i^{(k-1)} - \alpha\left(\nabla f_i(\mathbf{x}_i^{(k-1)}; \zeta_{i,k}) + \nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right)$
        end for
    end for
    $\mathbf{x} \leftarrow \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i^{(K)}$
end while
```

We observe that unlike $FedGA$, the displacement $\boldsymbol{v}_{i,SCAFFOLD}^{(k)}$ from the starting parameters $\mathbf{x}$ prior to the k_{th} gradient step for SCAFFOLD, involves $k-1$ drift correction terms $-\alpha\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right)$ in addition to the $k-1$ local gradient steps. Thus we have:

$$\boldsymbol{v}_{i,SCAFFOLD}^{(k)}(\mathbf{x}) = -(k-1)\alpha\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right) + \sum_{j=1}^{k-1} g_{i,SCAFFOLD}^{(j)}(\mathbf{x}),$$

where $g_{i,SCAFFOLD}^{(j)}(\mathbf{x})$ denotes the j_{th} gradient step for client i. Smilar to FedGA and SGD, $g_{i,SCAFFOLD}^{(j)}(\mathbf{x})$ can be evaluated by inductively computing the local displacements and gradient steps to obtain:

$$\begin{aligned}
&g_{i,SCAFFOLD}^{(k)}(\mathbf{x}) = -\alpha\nabla f_i(\mathbf{x};\zeta_{i,k}) \\
&- \alpha\nabla^2 f_i(\mathbf{x})\left(\boldsymbol{v}_{i,SCAFFOLD}^{(k)}(\mathbf{x})\right) + \mathcal{O}(\alpha^3) \\
&= -\alpha\nabla f_i(\mathbf{x};\zeta_{i,k}) - \alpha\nabla^2 f_i(\mathbf{x};\zeta_{i,k})\left(-(k-1)\alpha\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right) - \sum_{j=1}^{k-1}\alpha\nabla f_i(\mathbf{x};\zeta_{i,j})\right) + \mathcal{O}(\alpha^3).
\end{aligned}$$

The expected difference between the parameters obtained after one round of SCAFFOLD and FedAvg is then given by:

$$\begin{aligned}
&\mathbb{E}\left[\mathbf{x}_{SCAFFOLD} - \mathbf{x}_{FedAVG}\right] \\
&= \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{k=1}^{K} g_{i,SCAFFOLD}^{(k)}(\mathbf{x}) - \alpha\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right)\right) - \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} g_{i,FedAvg}^{(k)}(\mathbf{x})\right] \\
=& -\mathbb{E}\left[\frac{\alpha}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}\left(\nabla f_i(\mathbf{x};\zeta_{i,k}) + \left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right)\right)\right] \\
&+ \mathbb{E}\left[\frac{\alpha}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}\nabla^2 f_i(\mathbf{x};\zeta_{i,k})\left(\alpha(k-1)\left(\nabla f(\mathbf{x}) - \nabla f_i(\mathbf{x})\right) + \alpha\sum_{l=1}^{k-1}\nabla f_i(\mathbf{x};\zeta_{i,l})\right)\right] + \mathcal{O}(\alpha^3) \\
&+ \mathbb{E}\left[(\frac{\alpha}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}\nabla f_i(\mathbf{x};\zeta_{i,k}) - \frac{\alpha}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}\nabla^2 f_i(\mathbf{x};\zeta_{i,k})(\alpha\sum_{l=1}^{k-1}\nabla f_i(\mathbf{x};\zeta_{i,l})) + \mathcal{O}(\alpha^3))\right] \\
&= -\mathbb{E}\left[\frac{\alpha^2 K(K-1)}{2n}(\sum_{i=1}^{n}(\nabla^2 f_i(\mathbf{x};\zeta_{i,l})\left(\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\right)))\right] + \mathcal{O}(\alpha^3) \\
=& -\frac{\alpha^2 K(K-1)}{2n}(\sum_{i=1}^{n}(\nabla^2 f_i(\mathbf{x})\nabla f_i(\mathbf{x}) - \nabla^2 f_i(\mathbf{x})\nabla f(\mathbf{x}) - \nabla^2 f(\mathbf{x})\nabla f_i(\mathbf{x}) + \nabla^2 f(\mathbf{x})\nabla f(\mathbf{x}))) + \mathcal{O}(\alpha^3) \\
=& -\frac{\alpha^2 K(K-1)}{2n}(\sum_{i=1}^{n}(\nabla^2 f_i(\mathbf{x}) - \nabla^2 f(\mathbf{x}))(\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})))) + \mathcal{O}(\alpha^3) \\
=& -\frac{\alpha^2 K(K-1)}{4n}\nabla_{\mathbf{x}}((\sum_{i=1}^{n}\|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\|^2) + \mathcal{O}(\alpha^3).
\end{aligned}$$

A.7. Limitations

While we present concrete and sound theoretical results, they heavily rely on Taylor's theorem, which only provides accurate information in the vicinity of the studied point. Thus, one might need to account for the impact of the error term once we start moving away from the studied point. Nevertheless, our experiments with finite step sizes, strongly support our theoretical analysis.

Indeed, the main point of our experiments is to show that our theoretical results carry on to practical settings. We do not claim, however that our algorithm achieves state-of-the-art results, but sheds light on the impact that implicit regularization might have on the training of neural networks on non-artificial data sets.

Both federated learning and distributed datacenter settings studied in this work heavily depend on many hyperparameters (server momentum, normalization, learning rate decay scheduling, etc.) that we decided to ignore in this work. This allowed us to isolate the effect of implicit regularization, but it remains to study the interplay they have with FedGA.

Lastly, the overhead in communication and computation cost in federated and distributed learning due to the calculation of the drift limits the scalability of our approach. Nevertheless, several techniques could be used to alleviate this issue (Karimireddy et al., 2020).

A.8. Societal Impact

We believe that collaborative learning schemes such as federated learning are an important element towards enabling privacy-preserving training of ML models, as well as for a better alignment of each participating individual's data ownership with the resulting utility from a jointly trained machine learning model, especially in applications where data is user-provided and privacy sensitive (Kairouz et al., 2019; Nedic, 2020).

In addition to privacy, efficiency gains in distributed training reduce the environmental impact of training large machine learning models. The study of limitations of such methods in the realistic setting of heterogeneous data and algorithmic and practical improvements to the efficiency of such methods, is expected to help as a step towards achieving the goal of collaborative privacy-preserving and efficient decentralized learning.

B. Experiments Appendix

B.1. Model architectures

For the EMNIST experiments, we trained a CNN model with 2 convolutional layers followed by a fully connected layer.

For the CIFAR10 experiments, we trained a CNN model with 2 convolutional layers followed by three fully connected layers.

B.2. Experiment Hyperparameters

B.2.1. Federated Learning

We used a constant learning rate for each experiment, and we did not use momentum. For each algorithm we tuned the learning rate from {0.05, 0.1, 0.2, 0.4}. We tuned our algorithms with two batch sizes: 2400 corresponding to the entire dataset in each of the 47 workers, and 240 corresponding to 10% of the worker's data. Weight decay (L_2 regularization) was tuned from {0.001, 0.0001}, where the former achieved better test accuracy in all reported cases.

The number of local steps of each algorithm was tuned from {1, 10, 20, 40}, which corresponds to 1, 10, 20, and 40 local epochs with batch-size 2400, respectively, and 0.1, 1, 2, and 4 local epochs with batch-size 240, respectively. In the IID setting, using batch-size 240 always achieved higher test accuracy. Furthermore, better generalization was achieved using either 10 local steps. Thus, the use of more local epochs might increase convergence speed in terms of the number of rounds, but has only a detrimental effect on the maximum test accuracy achievable; see Section B.4.

The most challenging parameter to tune was β, the constant in front of the displacement in FedGA; see Algorithm 2. We started with a coarse grid search with β tuned from {0.01, 0.1, 1.0, 5.0}. After finding the best value in each of the two settings (IID and heterogeneous), we perform a fine grid search around it. For the IID setting where the gradient variance is much smaller, we used a fine grid search with β tuned from {0.5, 1.5, 2.5, 3.5}, with the best results for β between 1.5 and 2.5. In the heterogeneous setting, where the variance is much larger, we used a fine grid search with β in {0.01, 0.025 0.05, 0.1}, with the best results between 0.025 and 0.05; orders of magnitude smaller than for the IID case. For more details, see Section B.3.

B.2.2. Datacenter distributed learning

We used a constant learning rate for each experiment, and we did not use momentum. For each algorithm we tuned the learning rate from {0.05, 0.1, 0.2, 0.4}. Weight decay (L_2 regularization) was tuned from {0.001, 0.0001}, where the former achieved better Test Accuracy in all reported cases.

Sampling all clients Due to hardware and time constraints, we limited our search to a batch size of 125, which represents 2.5% of the 5000 data examples in each worker. The number of local steps of each algorithm was tuned within {10, 20}. In a federated learning setting, one might try to increase these numbers, but in the datacenter distributed setting, we assume that communication is not the bottleneck. Thus, while further experiments could be done, we believe our settings represent well the objectives of this paper. Furthermore, while we did not perform an exhaustive study for this task/architecture as in Section B.4, we also notice that an increase in local steps has not further benefit in test accuracy.

As in the federated learning setting, the most challenging parameter to tune was β. We started with a coarse grid search with β tuned from $\{0.01, 0.1, 1.0, 5.0\}$. After finding the best interval, we perform a fine grid search. For this datacenter distributed setting, we found the gradient variance to be also quite small. Thus, we used a fine grid search with β tuned from $\{0.5, 1.5, 2.5, 3.5\}$, with the best results for $\beta = 2.5$.

Minimizing number of updates. In these settings, we are restricted to use exactly one local step in each round. We tuned our algorithms with two batch sizes: 1000 and 5000, corresponding to 20% and 100% of the worker's data, respectively. The tuning of the β parameter was performed in the same way as in the above setting, and the results were quite similar, with β between 1.5 and 2.5 being the best range of values.

B.3. Tuning the β parameter of FedGA

As mentioned in Section B.2, it was challenging to tune β as it depends heavily on the variance of the gradients. In our experiments, we used a coarse level grid search, follower by a fine-tuning. However, as depicted in Figure 5, it might seem that the test accuracy as a function of beta might be a concave function, which can greatly help with its optimization. While the possibility of modifying β seems to offer an advantage over SCAFFOLD, it brings along the additional challenge of tuning it.

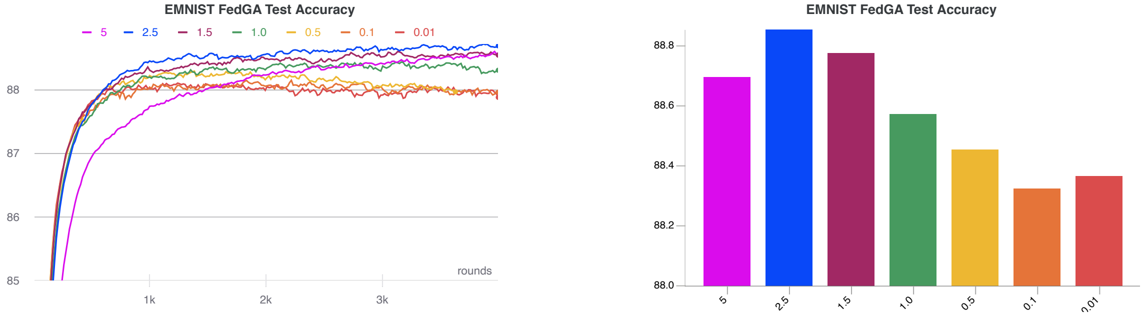

Figure 5. Depicts the effect of tuning the parameter β for one of the gird search settings we tried. For this example, we fix the batch size to 240 in the IID setting, weight decay 0.001, learning rate 0.2, and 10 local steps. From our experiments, it seems that the test accuracy as a function of beta is concave, which might help with its optimization. The experiments were performed with the same initial random seed.

B.4. Effect of local epochs

For our grid-search with the number of steps in {1,10,20,40}, which corresponds to 0.1, 1, 2, and 4 local epochs, we notice that beyond 10 local steps, there is no generalization benefit. Moreover, we can see a detriment in the maximum test accuracy; see Figure 6. There is, however, a much faster convergence using more local steps, but to a model with worse test accuracy. Similar behavior was spotted in FedGA and Scaffold. While this phenomenon might be overcome by a further reduction of the learning rate, this was beyond the parameters in our grid search.

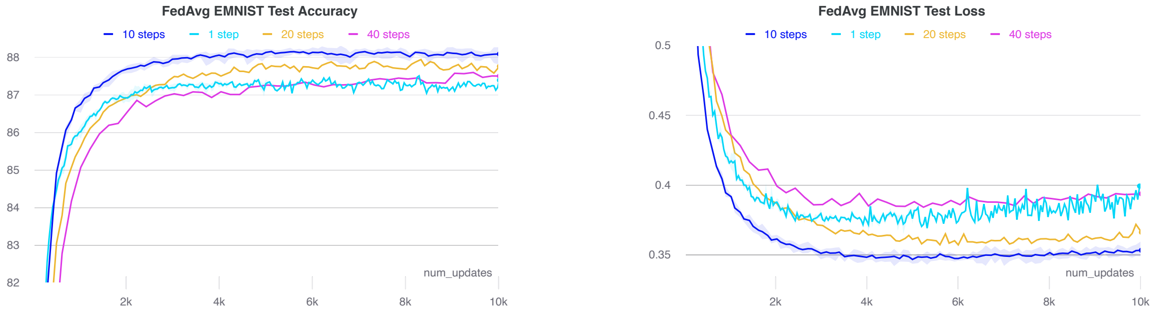

Figure 6. Best performances of FedAvg with batch size 240 and IID data distribution within our grid search. The x-axis shows the total number of (local) updates performed by the algorithm.