ANALYZING IMPLICIT REGULARIZATION IN FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Backward error analysis is a powerful technique that can check how much the path of the gradient flow is modified under the influence of a finite learning rate. Through this technique, it is also possible to find an *implicit regularizer* that affects the convergence behavior of an optimizer. With a backward error analysis, this paper seeks a more intuitive but quantitative way to understand the convergence behaviour under various federated learning algorithms. We prove that the implicit regularizer for FedAvg disperses the gradient of each client from the average gradient, increasing the gradient variance. We then theoretically present that the implicit regularizer of FedAvg hampers the convergence if the variance of gradients from clients decreases following the gradient of the cost function. In order to verify our analysis, we run experiments on FedAvg with and without the drifting term and confirm that FedAvg without the drifting term shows higher test accuracies. Our analysis also explains the convergence behavior of variance reduction methods such as SCAFFOLD, FedDyn, and FedSAM to show that the implicit regularizers of those methods have a smaller or zero drifting effect when the learning rate is small. Especially, we provide a possible reason FedSAM can perform better than FedAvg but might not perform as well as other stable variance reduction methods under data heterogeneity.

1 INTRODUCTION

A machine learning task is data hungry by nature. Massive data must be sent to a server from users in order to train a single neural network. Not only can this cause a severe communication overhead, in the users' perspective, such data collection itself can be seen as an invasion of privacy. Federated learning (McMahan et al., 2017) has emerged as a solution to this problem. In federated learning, training data remains on the users' device and only the parameter of locally trained model is transmitted to the server. The server aggregates the parameters sent from users and build a generalized global model. Since the transmission of raw user data is not needed in federated learning, a communication overhead can be reduced and the users' privacy can be protected.

The most popular optimization algorithm for federated learning is FedAvg (McMahan et al., 2017), which computes the parameter of the global model by simply averaging the parameters of local devices. Though FedAvg gained popularity for its simplicity, the convergence speed of FedAvg is known to be far slower than centralized learning and the final performance of the neural network trained with FedAvg is subpar (Zhao et al., 2018) (Karimireddy et al., 2020). This is due to an endemic problem originated from the nature of distributed optimization algorithms: client drift. Many works as Wang et al. (2019), Yu et al. (2019), Li et al. (2020), Khaled et al. (2020), Haddadpour & Mahdavi (2019) have already done sharp analysis on the convergence of FedAvg on non-IID data, focusing on the asymptotic number of communication rounds for FedAvg to reach a certain accuracy. On the other hand, in this paper, we borrow the concept of implicit regularization to explain in depth how the distributed property of FedAvg algorithm itself affects the convergence.

The concept of implicit regularization was originally introduced to explain the generalization behavior of gradient descent (GD) and stochastic gradient descent (SGD) (Barrett & Dherin, 2020) (Smith et al., 2021). Their approach, heavily inspired by backward error analysis (Hairer et al., 2006), was to check the path on which discrete updates of gradient descent lie. The path of the gradient flow by discrete updates follows not an original loss, but a loss modified under the influence of a small

but finite learning rate. The approximation of modified loss function is comprised of two terms: the original loss and an implicit regularizer, which is proportional to the learning rate. In GD and SGD, the implicit regularizer seeks a flat minima by penalizing the Euclidian norm of the gradient and contributes to the generalization performance of GD and SGD.

Main Result. The same backward error analysis technique in Smith et al. (2021) can be used to find the implicit regularizer for federated learning. Firstly, we define the cost function of k-th mini-batch sample of j-th client in the i-th round as $C_{ijk}(\omega)$. The mean cost functions are each defined as $C_{ij}(\omega) = \frac{1}{E}\sum_{k=0}^{E-1} C_{ijk}(\omega)$, $C_i(\omega) = \frac{1}{m}\sum_{j=0}^{m-1} C_{ij}(\omega)$, and $C(\omega) = \frac{1}{n}\sum_{i=0}^{n-1} C_i(\omega)$ while the number of samples in a round, clients in a round, and the total rounds are defined as E, m, and n. When the learning rate is ϵ, we found that the modified loss for federated learning is given by Equation 1. It is possible to observe that the implicit regularizer for federated learning is different from the one for SGD. Here, the regularizer is comprised of two terms: $\frac{E\epsilon}{4mn}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\|\nabla C_i(\omega) - \nabla C_{ij}(\omega)\|^2$ and $\frac{\epsilon}{4mnE}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\sum_{k=0}^{E-1}\|\nabla C_{ijk}(\omega)\|^2$, called the *drifting term* and the *generalizing term* in this paper respectively.

$$\tilde{C}_{AVG}(\omega) = C(\omega) - \frac{E\epsilon}{4mn}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\|\nabla C_i(\omega) - \nabla C_{ij}(\omega)\|^2 + \frac{\epsilon}{4mnE}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\sum_{k=0}^{E-1}\|\nabla C_{ijk}(\omega)\|^2 \tag{1}$$

While the generalizing term has the same format as the implicit regularizer of SGD, the drifting term is different. When the modified loss is reduced, the drifting term increases the distance between the gradient of each client from the average gradient of clients in a round. We theoretically show that this might hamper the convergence of FedAvg when the gradient variance decreases following the gradient of the cost function while it might improve the convergence in the opposite situation.

One way to confirm the effect of the drifting term is to compare the convergence behaviour of FedAvg with and without the drifting term. For verification, we explicitly remove the drifting term from the updated parameter and the modified loss has the form of

$$\tilde{C}_{modified}(\omega) = C(\omega) + \frac{\epsilon}{4mnE}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\sum_{k=0}^{E-1}\|\nabla C_{ijk}(\omega)\|^2 \tag{2}$$

Now the modified loss does not increase the gradient variance and the convergence behaviour also changes. Through a series of experiments, we verify that removing the drifting term leads to a better convergence behaviour under the influence of a finite learning rate.

Explicitly removing the drifting term is not the only method to mitigate the drifting effect. Many variance reduction methods like SCAFFOLD (Karimireddy et al., 2020) and FedDyn (Acar et al., 2021) enforce the client gradients to be close to each other and automatically reduce the effect of the drifting term. In this paper, we analyze the implicit regularizer of SCAFFOLD and FedDyn and show that the drifting term is removed when training is done on those methods. Also, we show that FedSAM (Caldarola et al., 2022) (Qu et al., 2022) also reduces the drifting term and mitigates the drifting effect, but might not perform well as other stable variance reduction methods.

Contributions. We summarize our main contributions below.

- Through a backward error analysis, we provide a more intuitive way to understand the convergence behaviour of FedAvg. We also provide an empirical and a rough theoretical explanation of how a drifting term affects the convergence behaviour.
- We show that our analysis can also explain why previous variance reduction methods such as SCAFFOLD, FedDyn, and FedSAM perform better than FedAvg. Especially, we provide an explanation why FedSAM can show better results than FedAvg but might not show better results than other stable variance reduction methods under data heterogeneity.

2 Related work

Implicit regularization. Implicit regularization is the key baseline concept of this work. Through a backward error analysis, it is possible to observe which path the gradient flow actually takes under

the influence of implicit regularization caused by a finite learning rate. While the concept was originally used to assess the modified flow of GD and SGD, in this work, we adapt this concept to analyze the gradient flow of FedAvg, SCAFFOLD, FedDyn, and FedSAM.

Variance reduction. Variance reduction methods such as the use of control variates were originally used to reduce variance in Monte Carlo sampling (Glasserman, 2004). Similar approaches are now being used when minimizing functions that have a finite sum structure. SAGA (Defazio et al., 2014) and SVRG (Johnson & Zhang, 2013) are representative examples of such stochastic variance reduction algorithms. These algorithms enforce the gradients to be close to the flow of finite sum objectives using control variates. Inspired by these techniques, there were many attempts to adapt variance reduction methods to federated learning setting. SCAFFOLD (Karimireddy et al., 2020) is a good example - it uses control variates to remove client-variance in the gradients and shows a superior convergence rate compared to FedAvg. FedDyn (Acar et al., 2021) is another example of variance reduction method, but it shows a better convergence rate than SCAFFOLD and works without transmission of extra variables.

Sharpness-aware minimization. Penalizing the gradient norm is similar to sharpness-aware minimization (SAM) (Foret et al., 2021) in the sense that both actions guide gradient descent toward flat minima. SAM has shown excellent generalization performance in various experiments and been applied to various areas, including federated learning. FedASAM (Caldarola et al., 2022) and FedSAM (Qu et al., 2022) are good examples that applied sharpness-aware minimization to federated learning. Both algorithms have shown a faster convergence speed than FedAvg. From a certain point of view, this work can be seen as an explanation of how sharpness-aware minimization reduces client drifts and leads to faster convergence in federated learning from a different perspective.

3 A BACKWARD ERROR ANALYSIS OF FEDERATED LEARNING

As in Smith et al. (2021), we now use backward error analysis to estimate the implicit bias under federated learning. After providing a brief overview on the idea of backward error analysis, we prove that the implicit regularizer of FedAvg increases the variance of gradients from clients.

3.1 THE IDEA OF BACKWARD ERROR ANALYSIS

This whole section briefly explains the idea of backward error analysis done in Barrett & Dherin (2020) and Smith et al. (2021). The basic idea starts from regarding gradient descent as integrating an ODE of the form $\dot{\omega} = -\nabla C(\omega)$. Then discrete iterates of gradient descent can be seen as solving the integration problem with explicit Euler method of order 1, like $\omega(t+\epsilon) \approx \omega(t) - \epsilon\nabla C(\omega(t))$. Unfortunately, when the step size is finite, discreteness of gradient descent steps will bring about a certain discrepancy from the exact solution of differential equation. To bridge the gap, when $\dot{\omega} = -\nabla C(\omega) = f(\omega)$, we introduce the approximated flow of $\dot{\omega} = \tilde{f}(\omega)$, where $\tilde{f}(\omega)$ is expressed in powers of step size ϵ.

$$\tilde{f}(\omega) = f(\omega) + \epsilon f_1(\omega) + \epsilon^2 f_2(\omega) + \dots \tag{3}$$

This is viable when the learning rate ϵ is finite but relatively small. Now the role of backward error analysis is to find the function for each correction term $f_i(\omega)$. The idea here is to take infinite number of infinitesimal steps along the modified flow to reach the solution $\omega(t+\epsilon)$. Starting from ω_t, the parameter becomes ω_{t+n} after taking n Euler steps with infinitesimal step size α and for ω_{t+n},

$$\omega_{t+n} = \omega_t + \alpha\tilde{f}(\omega_t) + \alpha\tilde{f}(\omega_{t+1}) + \alpha\tilde{f}(\omega_{t+2}) + \dots \tag{4}$$

$$= \omega_t + \alpha\tilde{f}(\omega_t) + \alpha\tilde{f}(\omega_t + \alpha\tilde{f}(\omega_t)) + \alpha\tilde{f}(\omega_t + \alpha\tilde{f}(\omega_t) + \alpha\tilde{f}(\omega_t + \alpha\tilde{f}(\omega_t))) + \dots \tag{5}$$

$$= \omega + n\alpha\tilde{f}(\omega_t) + \frac{n(n-1)}{2}\alpha^2\nabla\tilde{f}(\omega_t)\tilde{f}(\omega_t) + O(n^3\alpha^3) \tag{6}$$

Equation 6 is done by taking the Taylor expansion of $\tilde{f}$. Now we let $\epsilon = n\alpha$ while $n \to \infty$ and assume that ϵ is relatively small but finite. Since ω_{t+n} reaches $\omega(t+\epsilon)$, assuming the Taylor

expansion of $\tilde{f}$ converges for ϵ, it is possible to say that

$$\omega(t+\epsilon) = \omega(t) + \epsilon\tilde{f}(\omega(t)) + \frac{\epsilon^2}{2}\nabla\tilde{f}(\omega(t))\tilde{f}(\omega(t)) + O(\epsilon^3) \tag{7}$$

$$= \omega(t) + \epsilon f(\omega(t)) + \epsilon^2(f_1(\omega(t)) + \frac{1}{2}\nabla f(\omega(t))f(\omega(t))) + O(\epsilon^3) \tag{8}$$

In order to derive the correction terms, what to do is identifying $\omega(t+\epsilon)$ with the parameter ω after taking a single or multiple steps of optimization. For gradient descent (GD), as a simple example, the parameter after a single discrete update, $\omega_{t+1} = \omega_t - \epsilon\nabla C(\omega_t)$, is identified with $\omega(t+\epsilon)$. Then $f(\omega)$ can be fixed as $-\nabla C(\omega)$. Also, at order ϵ^2, the coefficient must go to 0 which means that

$$f_1(\omega) = -\frac{1}{2}\nabla\nabla C(\omega)\nabla C(\omega) = -\frac{1}{4}\nabla\|\nabla C(\omega)\|^2 \tag{9}$$

Finally, it is possible to say that discrete GD iterates follow the path of an ODE with the form of

$$\dot{\omega} = -\nabla C(\omega) - \frac{\epsilon}{4}\nabla\|\nabla C(\omega)\|^2 + O(\epsilon^2) \tag{10}$$

If the step size ϵ is small enough to ignore high order terms, the modified loss that gradient flow of GD follows can be expressed as

$$\tilde{C}_{GD}(\omega) = C(\omega) + \frac{\epsilon}{4}\|\nabla C(\omega)\|^2 \tag{11}$$

3.2 Backward error analysis for FedAvg

The same technique can be used to explain convergence behavior under FedAvg. Unlike in GD, where only a single step was taken into consideration, we take the result of discrete steps of multiple rounds to match the solution $\omega(t+\epsilon)$ of the continuous flow.

Firstly, we define the necessary variables beforehand. The cost function of k-th mini-batch sample of j-th client in the i-th round is defined as $C_{ijk}(\omega)$. The mean cost functions are each defined as $C_{ij}(\omega) = \frac{1}{E}\sum_{k=0}^{E-1} C_{ijk}(\omega)$, $C_i(\omega) = \frac{1}{m}\sum_{j=0}^{m-1} C_{ij}(\omega)$, and $C(\omega) = \frac{1}{n}\sum_{i=0}^{n-1} C_i(\omega)$ while the number of samples in a round, clients in a round, and the total rounds are defined as E, m, and n, respectively. The parameter, ω^{ij}, is trained on the j-th client in the i-th round and these parameters are aggregated to form ω^i in the end of the i-th round. The whole aggregated global parameter is ω.

For one client in one round, the learning procedure is the same as the one in SGD. Borrowing the result from Smith et al. (2021), for j-th client in the i-th round, discrete updates of the parameter ω^{ij} during E steps with the learning rate ϵ can be expressed as

$$\omega_E^{ij} = \omega_0^{ij} - \epsilon\nabla C_{ij0}(\omega_0^{ij}) - \epsilon\nabla C_{ij1}(\omega_1^{ij}) - \epsilon\nabla C_{ij2}(\omega_2^{ij}) - \cdots - \epsilon\nabla C_{ij(E-1)}(\omega_{E-1}^{ij}) \tag{12}$$

$$= \omega_0^{ij} - \epsilon\sum_{k=0}^{E-1}\nabla C_{ijk}(\omega_0^{ij}) + \epsilon^2\sum_{k=0}^{E-1}\sum_{l<k}\nabla\nabla C_{ijk}(\omega_0^{ij})\nabla C_{ijl}(\omega_0^{ij}) + O(E^3\epsilon^3) \tag{13}$$

$$= \omega_0^{ij} - \epsilon\sum_{k=0}^{E-1}\nabla C_{ijk}(\omega_0^{ij}) + \epsilon^2\xi(\omega_0^{ij}) + O(E^3\epsilon^3) \tag{14}$$

However, the function ξ depends on the order of mini-batch samples, which makes it hard to interpret the meaning of bias. Knowing that mini-batch samples are randomly shuffled during training, instead, we obtain the expectation value of ξ^{ij} and get the expectation value of ω_E^{ij} for easy understanding.

$$\mathbb{E}(\xi^{ij}) = \frac{1}{2}\sum_{k=0}^{E-1}\sum_{l\neq k}\nabla\nabla C_{ijk}\nabla C_{ijl} \tag{15}$$

$$= \frac{E^2}{2}\nabla\nabla C_{ij}\nabla C_{ij} - \frac{1}{2}\sum_{k=0}^{E-1}\nabla\nabla C_{ijk}\nabla C_{ijk} \tag{16}$$

$$= \frac{E^2}{4}\nabla(\|\nabla C_{ij}\|^2 - \frac{1}{E^2}\sum_{k=0}^{E-1}\|\nabla C_{ijk}\|^2) \tag{17}$$

$$\mathbb{E}(\omega_E^{ij}) = \omega_0^{ij} - E\epsilon\nabla C_{ij}(\omega_0^{ij}) + \frac{E^2\epsilon^2}{4}\nabla(\|\nabla C_{ij}(\omega_0^{ij})\|^2 - \frac{1}{E^2}\sum_{k=0}^{E-1}\|\nabla C_{ijk}(\omega_0^{ij})\|^2) + O(E^3\epsilon^3) \tag{18}$$

The same iterations proceed on multiple clients during multiple rounds. If there are m clients participating in each round, the expectation value of ω_E^i, the parameter after one round would be

$$\mathbb{E}(\omega_E^i) = \omega_0^i - E\epsilon\nabla C_i(\omega_0^i) + \frac{E^2\epsilon^2}{4m}\nabla(\sum_{j=0}^{m-1}\|\nabla C_{ij}(\omega_0^i)\|^2 - \frac{1}{E^2}\sum_{j=0}^{m-1}\sum_{k=0}^{E-1}\|\nabla C_{ijk}(\omega_0^i)\|^2) + O(E^3\epsilon^3) \tag{19}$$

Using this result, we can obtain the expectation value of ω_{nE}, which is the parameter after going through n rounds. But before we get $\mathbb{E}(\omega_{nE})$, we get the value of ω_{nE} itself first.

$$\omega_{nE} = \omega_0 - E\epsilon\nabla C_0(\omega_0) + E^2\epsilon^2\xi_0(\omega_0) - E\epsilon\nabla C_1(\omega_E) + E^2\epsilon^2\xi_1(\omega_E) - \cdots + O(E^3\epsilon^3) \tag{20}$$

$$= \omega_0 - E\epsilon\sum_{i=0}^{n-1}\nabla C_i(\omega_0) + E^2\epsilon^2\sum_{i=0}^{n-1}\sum_{l<i}\nabla\nabla C_i(\omega_0)\nabla C_l(\omega_0) + E^2\epsilon^2\sum_{i=0}^{n-1}\xi_i(\omega_0) + O(E^3\epsilon^3) \tag{21}$$

In order to obtain $\mathbb{E}(\omega_{nE})$, we combine the results from Equation 18, 19, and 21.

$$\mathbb{E}(\omega_{nE}) = \omega_0 - nE\epsilon\nabla C(\omega_0) + \frac{n^2E^2\epsilon^2}{4}\nabla(\|\nabla C(\omega_0)\|^2 - \frac{1}{n^2}\sum_{i=0}^{n-1}\|\nabla C_i(\omega_0)\|^2$$
$$+ \frac{1}{mn^2}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\|\nabla C_{ij}(\omega_0)\|^2 - \frac{1}{mn^2E^2}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\sum_{k=0}^{E-1}\|\nabla C_{ijk}(\omega_0)\|^2) + O(n^3E^3\epsilon^3) \tag{22}$$

Reminding of Equation 8, if we let ϵ in Equation 8 to be $nE\epsilon$, ω_{nE} must have the form of

$$\omega_{nE} = \omega_0 - nE\epsilon\nabla C(\omega_0) + n^2E^2\epsilon^2(f_1(\omega_0) + \frac{1}{4}\nabla\|\nabla C(\omega_0)\|^2) + O(n^3E^3\epsilon^3) \tag{23}$$

Then it is possible to assess the correction term f_1 to be

$$f_1 = -\frac{1}{4n^2}\sum_{i=0}^{n-1}\|\nabla C_i\|^2 + \frac{1}{4mn^2}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\|\nabla C_{ij}\|^2 - \frac{1}{4mn^2E^2}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\sum_{k=0}^{E-1}\|\nabla C_{ijk}\|^2 \tag{24}$$

Since an ODE has the approximated form of $\dot{\omega} = -\nabla C(\omega) + nE\epsilon f_1(\omega) = -\nabla\tilde{C}_{AVG}(\omega)$, the modified loss that the gradient flow follows is

$$\tilde{C}_{AVG}(\omega) = C(\omega) - \frac{E\epsilon}{4mn}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\|\nabla C_i(\omega) - \nabla C_{ij}(\omega)\|^2 + \frac{\epsilon}{4mnE}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\sum_{k=0}^{E-1}\|\nabla C_{ijk}(\omega)\|^2 \tag{25}$$

and we deduce Equation 25 noting that $\sum_{j=0}^{m-1}(\nabla C_i(\omega) - \nabla C_{ij}(\omega)) = 0$. Also note that equations above are viable only when n, E, and ϵ are small enough to neglect high order terms.

Unlike in GD or SGD, the implicit regularizer for FedAvg is composed of two terms: $\frac{E\epsilon}{4mn}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\|\nabla C_i(\omega) - \nabla C_{ij}(\omega)\|^2$ and $\frac{\epsilon}{4mnE}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\sum_{k=0}^{E-1}\|\nabla C_{ijk}(\omega)\|^2$. We call each term the *drifting term* and the *generalizing term* in this paper. The generalizing term, which has the same format as the implicit regularizer of SGD, is the term known to aid generalization and help the converged model to achieve higher accuracies (Smith et al., 2021). The drifting term is the one that makes a difference. When the modified loss is reduced, the drifting term increases the distance between the gradient of each client and the average gradient of clients in a round. This will increase the gradient variance which can be a problem to the convergence of FedAvg in certain stages of training. We theoretically provide a rough indication how and when the drifting term hampers the convergence of FedAvg by increasing the gradient variance in the appendix.

However one thing to note is that, though we expressed that the drifting term *disperses* the client gradients for easy understanding, it would be actually more accurate to say that the drifting term *allows* the client gradients to disperse from the average. This is because an implicit regularizer is merely a tool to show the path of the gradient flow, not a real loss function. Therefore, the effect of the increased gradient variance will become prominent if and only if the data setting is not completely i.i.d, where same all data is shared among clients.

4 INSPECTING THE EFFECT OF THE DRIFTING TERM

One way to inspect the effect of the drifting term is to compare the convergence behaviour of FedAvg with and without the drifting term. In order to empirically check the effect of the drifting term, we manually removed the drifting term from the updated parameter. We first calculated the average of client gradients $\nabla C_i(\omega_0^i)$ in the server and sent it to the client. After the client receiving the averaged gradient, we calculated the average of its mini-batch gradients $\nabla C_{ij}(\omega_0^i)$ in the client. We then obtained $\frac{E^2\epsilon^2}{4}\nabla\|\nabla C_i(\omega_0^i) - \nabla C_{ij}(\omega_0^i)\|^2$ and subtracted it from the parameter of the client. The modified loss of the aggregated parameter has the form below with no term of a drifting effect.

$$\tilde{C}_{modified}(\omega) = C(\omega) + \frac{\epsilon}{4mnE}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\sum_{k=0}^{E-1}\|\nabla C_{ijk}(\omega)\|^2 \tag{26}$$

From a certain view, removing the term can be regarded as simulating SGD. We ran a series of simple experiments with the original FedAvg and FedAvg with the method above to compare the results. We confirmed that the FedAvg with the method above showed higher test accuracies on non-IID settings and was able to show that the drifting term hampers convergence of FedAvg.

4.1 FEDSAM FOR VARIANCE REDUCTION

Though seeking flat minima seems far from removal of the drifting term, we suggest that seeking flat minima is highly related to mitigating the drifting effect. In this section, we analyze how FedSAM (Caldarola et al., 2022) (Qu et al., 2022) seeks flat minima to mitigate the drifting term and acquires better performance than FedAvg. For clarity, we refer to an approximation method in Zhao et al. (2022) and Geiping et al. (2022) that is used for computation of Hessian-vector product. Through a finite-differences approximation, the corresponding gradient to a gradient norm penalty is computed

$$\nabla\frac{1}{2}\|\nabla C(\omega)\|^2 \approx \frac{C(\omega + \varepsilon\nabla C(\omega)) - C(\omega)}{\varepsilon} \tag{27}$$

when the magnitude of ε is very small like $0.01/\|\nabla C(\omega)\|$ as in Foret et al. (2021). If we put this to a loss function as an explicit regularizer with coefficient λ, the approximated loss will be

$$C(\omega) + \frac{\lambda}{2}\|\nabla C(\omega)\|^2 \approx \frac{\lambda}{\varepsilon}C(\omega + \varepsilon\nabla C(\omega)) + (1 - \frac{\lambda}{\varepsilon})C(\omega) \tag{28}$$

If $\lambda = \varepsilon$, the loss function is the same as the one of sharpness-aware minimization. This shows that sharpness-aware minimization is equivalent to training a model with a gradient norm penalty. However, ε of sharpness-aware minimization could be smaller than $E\epsilon/2$ if the norm of the gradient is large. Also, FedSAM gives penalty to the mini-batch gradients instead of the norm of an average gradient of all mini-batches of a client. In this case, if ε is smaller than $E\epsilon/2$ and a subsidiary implicit regularization caused by the gradient penalty itself is ignored, the modified loss becomes

$$\begin{aligned}\tilde{C}_{FEDSAM}(\omega) = C(\omega) + \frac{\varepsilon}{2mn}\sum_{i=0}^{n-1}\|\nabla C_i(\omega)\|^2 - (\frac{E\epsilon}{4mn} - \frac{\varepsilon}{2mn})\sum_{j=0}^{m-1}\|\nabla C_i(\omega) - \nabla C_{ij}(\omega)\|^2 \\ + \frac{\varepsilon}{2mn}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\sum_{k=0}^{E-1}\|\nabla C_{ij}(\omega) - \nabla C_{ijk}(\omega)\|^2\end{aligned} \tag{29}$$

The implicit regularizer for FedSAM is composed of three terms is under the influence of ε. It is possible to observe that the presence of ε reduces the magnitude of the drifting term and mitigates the drifting effect. Also, the last term of the implicit regularizer minimizes the mini-batch gradient variance. If ε is smaller than $E\epsilon/2$, such effects will 'partially' mitigate the drifting effect and enhance the performance of the converged model. We discuss further in the appendix with experiments.

4.2 BACKWARD ERROR ANALYSIS FOR SCAFFOLD AND FEDDYN

Another simple solution to mitigate the drifting effect would be enforcing the client gradients to be close to each other. Variance reduction methods like Karimireddy et al. (2020) and Acar et al. (2021) are exactly the ones to do that kind of stuff. Utilizing the information from gradients of previous rounds, those methods prevent local gradients from deviating and mitigate the client drifting effect.

As done for FedAvg, a backward error analysis can be done for SCAFFOLD to explain the convergence behaviour. For convenience, we now assume that all the clients are fully participating in each round and each client is going through a full-batch training, which makes C_i, C_{ij} and C_{ijk} the same. Regardless of the change of the situation, the loss is modified under the influence of a finite learning rate and the modified loss under SCAFFOLD can be expressed as

$$\tilde{C}_{SCAFFOLD}(\omega) = C(\omega) + \frac{\epsilon}{4}\|\nabla C(\omega)\|^2 \tag{30}$$

when the learning rate is small enough to make $O(E^2\epsilon^2)$ negligible.

The same technique can be applied to FedDyn to analyze the convergence behaviour. Though FedDyn was influenced by SCAFFOLD, its method shows a big difference in one important regard: the time when the server control variate is added to the parameter. Unlike SCAFFOLD, which adds the server control variate every time the client updates its parameter, FedDyn adds the server control variate only when the server performs aggregation. Not only does this method make a difference to the communication overhead, but also makes a difference to the gradient flow. The loss is modified under the influence of finite learning rate and the modified loss under FedDyn can be expressed as

$$\tilde{C}_{FEDDYN}(\omega) \approx C(\omega) + \frac{E\epsilon}{4}\|\nabla C(\omega)\|^2 \tag{31}$$

when the learning rate is small enough to make the gradients from the previous and the current round close to each other. The loss above is viable only when $\alpha = 1/E\epsilon$. If α is smaller, FedDyn acts as if the learning rate has become larger than ϵ and the modified loss can be expressed as

$$\tilde{C}_{FEDDYN}(\omega) \approx C(\omega) + \frac{1}{4\alpha}\|\nabla C(\omega)\|^2 \tag{32}$$

We show a detailed analysis on SCAFFOLD and FedDyn in appendix.

The difference in the modified loss of FedDyn from the one of SCAFFOLD is the magnitude of the generalizing term. Due to a longer interval between additions of the server control variate, it shows an effect as if the learning rate has become larger. The magnitude of the generalizing term of FedDyn is E-times larger than the one of SCAFFOLD.

5 EXPERIMENTS

We now run experiments on multiple datasets to assess our analysis. We run experiments on FedAvg and multiple variance reduction methods: SCAFFOLD, FedDyn, FedSAM, and our method that removes the drifting term. We compare the results of these methods and observe how the drifting term and the generalizing term of the modified loss affects the convergence behaviour.

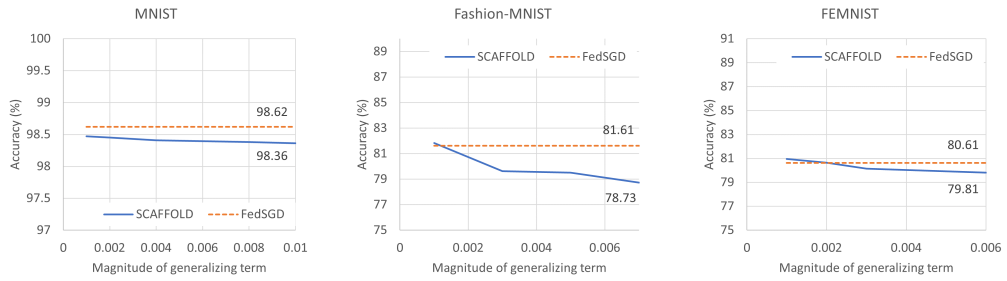
5.1 SETUP

Datasets and models. For benchmark, we use famous image datasets used in previous FL works as MNIST (LeCun et al., 1998), FEMNIST (Caldas et al., 2018), and Fashion-MNIST (Xiao et al., 2017). Except FEMNIST, which is naturally non-IID, all datasets were trained on both IID and non-IID environment. The IID environment was produced by randomly assigning samples to each client. For non-IID setting, we use a Dirichlet distribution with parameter 0.2 for uneven sampling of label ratios. 100 clients have been trained on MNIST, Fashion-MNIST and 520 clients have been trained on FEMNIST. As in previous works by Karimireddy et al. (2020) and Acar et al. (2021), training with full participation and with partial participation ratio of 0.1 were done on each dataset.

For MNIST and Fashion-MNIST, we use a CNN model consisting of 2 convolutional layers with 10 and 20 5×5 filters, fully-connected layers 50 neurons, and a softmax layer. For FEMNIST, we use a CNN model consisting of 2 convolutional layers with 32 and 64 7×7 filters and a softmax layer.

Participation	Dataset	FedAvg	SCAFFOLD	FedDyn	FedSGD	FedSAM	Ours
10%	Dirichlet(.2)						
	MNIST	96.73	97.82	97.8	98.11	96.94	98.19
	FMNIST	76.8	77.78	77.49	80.88	78.25	82.79
	IID						
	MNIST	98.03	98.04	98.03	98.12	98.05	98.11
	FMNIST	82.18	82.2	82.26	81.94	82.25	82.37
	non-IID						
	FEMNIST	73.83	76.56	76.92	76.47	75.24	76.52
100%	Dirichlet(.2)						
	MNIST	96.71	97.95	97.9	98.1	97.03	98.29
	FMNIST	77.98	78.86	79.24	81.61	78.65	82.88
	IID						
	MNIST	98.03	98.09	98.08	98.08	98.1	98.1
	FMNIST	82.15	82.09	82.19	81.58	82.21	82.23
	non-IID						
	FEMNIST	73.23	76.62	77.26	76.03	75.1	76.5

Table 1: Test accuracies(%) of federated learning optimization strategies on various datasets. To consider the stability of each method, we show the average of five highest accuracies that each optimization method has achieved. It is possible to observe that FedSGD and our method shows higher test accuaracies on non-IID settings. This results verify our backward error analysis on the drifting effect of FedAvg.



(a) (b) (c)

Figure 1: Test accuaries(%) of SCAFFOLD according to the magnitude of the generalizing term. The dotted line denotes the test accuracy of FedSGD with the learning rate of 0.001. For all datasets the test accuracy slightly decreases as the magnitude of the generalizing term increases. On the other hand, the magnitude of the generalizing term for FedSGD was 0.1 on MNIST and Fashion-MNIST, and 0.038 on FEMNIST which is larger than the maximum value of the x-axis. Despite its large generalizing term, FedSGD shows better performance than SCAFFOLD in most situations, which means that high-order terms of the modified loss have an impact that cannot be easily ignored.

Hyperparameters. To fully observe the effect of variance reduction methods, we used a normal SGD optimizer with no momentum and learning rate decay for all experiments. The learning rate was 0.001 for all datasets, which is small enough to satisfy the assumption of our analysis. The models were trained for 1000 rounds. For all datasets, the batch size was 30 and local epoch was 5.

For FedSAM, as in Foret et al. (2021), the value of ε was set as $0.01/\|\nabla C(\omega)\|$. For FedDyn, the setting of hyperparameter was an important issue. In the appendix, we found that setting α as a smaller value would make FedDyn show an effect as if the learning rate was bigger. For fair comparison on the effect of the drifting term, we decided to set α as $1/E\epsilon$ in all experiments. We obtained the average number of mini-batch samples of clients as E, and used $1/E\epsilon$ as α. The value of α was 10 for MNIST and Fashion-MNIST, 26.31 for FEMNIST. On the other hand, the same method was used to decide the value of E for our method with gradient penalty. The value of $E\epsilon$ was 0.1 for MNIST, Fashion-MNIST, and 0.038 for FEMNIST.

5.2 RESULTS

The effect of the drifting term The first experiment was to check the effect of the drifting term. We ran experiments of FedAvg with and without the drifting term on MNIST, Fashion-MNIST, and FEMNIST with non-IID settings and compared the results. We also compared the results with FedSAM for verification of our analysis. As shown in Table 1, the test accuracies were increased with our method and the performance was certainly better than the result of FedAvg. This result shows that the drifting term hampers the convergence of FedAvg in late stages of training.

Also, our method showed better performance than FedSAM under data heterogeneity. This result shows that FedSAM only partially mitigated the drifting effect as our expectation. To discuss more, we ran an experiment of switching the value of ε of FedSAM to $E\epsilon/2$ to inspect the effect of the implicit regularization of FedSAM. We describe details on the empirical analysis in the appendix.

The generalizing term and high-order terms The next experiment was to check the effect of a generalizing term and high-order terms. We ran experiments on SCAFFOLD and FedSGD. We did not use other method due to several reasons: the modified loss of FedDyn we acquired is in an fairly approximated form, and our method is prone to the gradient variance which makes it prone to a large learning rate as well. SCAFFOLD has a generalizing term with a coefficient $\epsilon/4$ while FedSGD with a learning rate of $E\epsilon$ is E-times larger. We compared results of two methods to inspect the effect of a generalizing term and high-order terms of an implicit regularizer.

In order to inspect the effect of the generalizing term, we ran SCAFFOLD with various learning rates so that the magnitude of the generalizing term would also be various. We ran experiments for a long period of time - for epochs whose number exceeds 1500×0.001 when multiplied by the learning rate. As shown in Figure 1, the test accuracy of the converged model decreased as the magnitude of the generalizing term increased, which is different from the result on SGD from Smith et al. (2021).

Another thing to observe was that the performance of FedSGD was superior than SCAFFOLD in most cases though the magnitude of the generalizing term for FedSGD was large - it was 0.1 for MNIST, Fashion-MNIST and 0.038 for FEMNIST which are larger than in all cases of SCAFFOLD on each dataset. We pointed out the presence of high-order terms of the modified loss as the cause of this phenomenon. Still, in Table 1, the gap between the results of FedAvg and SCAFFOLD were much bigger than the ones between SCAFFOLD and FedSGD and we concluded that the effect of the drifting term is much larger than the effect of high-order terms.

6 CONCLUSION

In this work, we used the backward error analysis technique to find the implicit regularizer for federated learning and analyze the convergence behaviour in a more intuitive way. The implicit regularizer for federated learning was composed of two terms: the drifting term, which disperses the gradient of each client from the average gradient of clients in a round, and the generalizing term, which penalizes the norm of mini-batch gradients. We inspected various methods that can mitigate the effect of the drifting term: SCAFFOLD and FedDyn, and FedSAM, etc. We also employed the backward error analysis technique to analyze those methods and showed that each method has its own modified loss with a smaller drifting term which leads to a better performance.

However, there exist limitations to our analysis. As in Smith et al. (2021), our analysis is valid only when the learning rate should be small enough to high-order terms insignificant. Such an assumption of a small learning rate was also made in previous works, for example, Karimireddy et al. (2020) assumed that the local learning rate is smaller than a threshold inversely proportional to the number of local updates. Another thing to notice is that our analysis does not consider the variance of a parameter due to a shuffled order of mini-batches, while such factors can affect the convergence behaviour. Some of those limitations was actually shown to be effective in our experimental results.

Despite these limitations, our analysis succeeded its purpose to provide an intuitive way to understand the behaviour of various federated learning methods. Also, the experimental results fairly corresponded to our predictions, which means that our analysis still holds. We expect our works to be used for easy understanding of convergence behaviours of various federated learning methods.

REFERENCES

Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, Matthew Mattina, Paul Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=B7v4QMR6Z9w.

David GT Barrett and Benoit Dherin. Implicit gradient regularization. *arXiv preprint arXiv:2009.11162*, 2020.

Debora Caldarola, Barbara Caputo, and Marco Ciccone. Improving generalization in federated learning by seeking flat minima. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIII*, pp. 654–672. Springer, 2022.

Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=6Tm1mposlrM.

Jonas Geiping, Micah Goldblum, Phil Pope, Michael Moeller, and Tom Goldstein. Stochastic training is not necessary for generalization. In *International Conference on Learning Representations*, 2022.

Paul Glasserman. *Monte Carlo methods in financial engineering*, volume 53. Springer, 2004.

Farzin Haddadpour and Mehrdad Mahdavi. On the convergence of local descent methods in federated learning. *arXiv preprint arXiv:1910.14425*, 2019.

Ernst Hairer, Marlis Hochbruck, Arieh Iserles, and Christian Lubich. Geometric numerical integration. *Oberwolfach Reports*, 3(1):805–882, 2006.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pp. 5132–5143. PMLR, 2020.

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*, pp. 4519–4529. PMLR, 2020.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data, 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

Zhe Qu, Xingyu Li, Rui Duan, Yao Liu, Bo Tang, and Zhuo Lu. Generalized federated learning via sharpness aware minimization. In *International Conference on Machine Learning*, pp. 18250–18280. PMLR, 2022.

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.

Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE journal on selected areas in communications*, 37(6):1205–1221, 2019.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5693–5700, 2019.

Yang Zhao, Hao Zhang, and Xiuyuan Hu. Penalizing gradient norm for efficiently improving generalization in deep learning. In *International Conference on Machine Learning*, pp. 26982–26992. PMLR, 2022.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

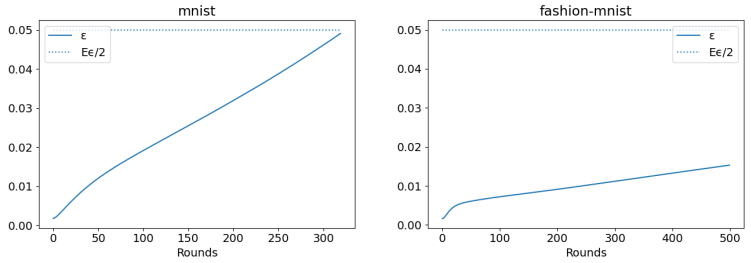
A CONVERGENCE BEHAVIOUR OF FEDSAM

The modified loss of FedSAM has the form below

$$\tilde{C}_{FEDSAM}(\omega) = C(\omega) + \frac{\varepsilon}{2mn}\sum_{i=0}^{n-1}\|\nabla C_i(\omega)\|^2 - (\frac{E\epsilon}{4mn} - \frac{\varepsilon}{2mn})\sum_{j=0}^{m-1}\|\nabla C_i(\omega) - \nabla C_{ij}(\omega)\|^2 + \frac{\varepsilon}{2mn}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\sum_{k=0}^{E-1}\|\nabla C_{ij}(\omega) - \nabla C_{ijk}(\omega)\|^2$$

if we ignore the subsidiary implicit regularizer caused by frequent application of mini-batch gradient norm penalty. The implicit regularizer for FedSAM is composed of three terms: a generalizing term that penalizes the gradient norm, a drifting term that disperses the client gradients, and a new term. One thing to notice is that all three terms depend on the variable ε. The presence of ε in the drifting term decreases the magnitude of the drifting term, reducing dispersion of client gradients. Also, it is possible to predict that the variance of mini-batch gradients will decrease when the new term, with a form of $\frac{\varepsilon}{2mn}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\sum_{k=0}^{E-1}\|\nabla C_{ij}(\omega) - \nabla C_{ijk}(\omega)\|^2$, is minimized.

We empirically checked the effect of those terms by changing the value of ε to $E\epsilon/2$, which can make the magnitude of the drifting term zero while increasing the magnitude of the new term. We have done experiments on MNIST and Fashion-MNIST on non-IID settings and full client participation. The learning rate was 0.001 and we trained the model for 300 rounds for MNIST, 500 rounds for Fashion-MNIST. The model was different from the main experiments: we used a CNN model consisting of 2 convolutional layers with 32 and 64 7×7 filters and a softmax layer for all three experiments, which is a model bigger than the model used in previous experiments for MNIST and Fashion-MNIST. We used a larger model for stability of mini-batch gradients.

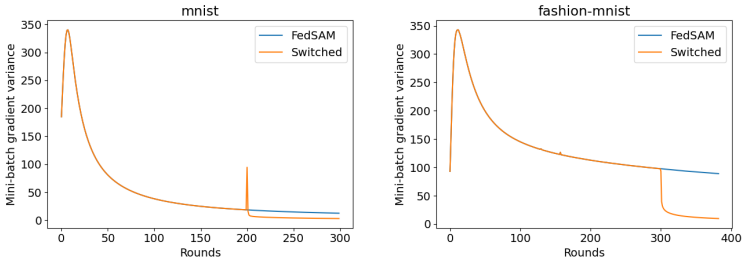


(a) (b)

Figure A.1: The value of ε during training. Training was done on (a) MNIST and (b) Fashion-MNIST with non-IID settings and full client paricipation. The dotted line denotes $E\epsilon/2$. It is possible to see that ε is smaller than $E\epsilon/2$.

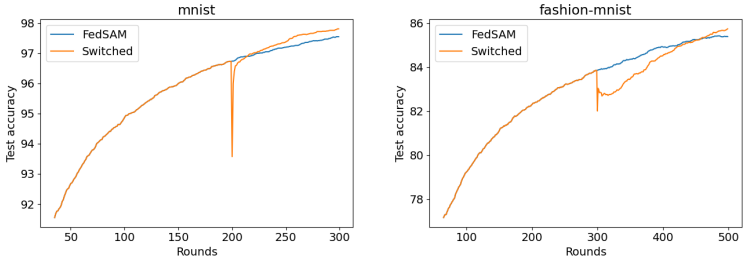
First thing done was investigating the value of ε during a normal training with FedSAM. The value of ε was set as $0.01/\|\nabla C_{ijk}(\omega)\|$. As shown in Figure A.1, while the value of ε mildly fluctuated during training, the value stayed below $E\epsilon/2$, which accords with our assumption on the value of ε. Next thing done was inspecting the effect of the mini-batch gradient variance. One attempt we made was to change the value of ε to $E\epsilon/2$ in the early stages of training. As a result, the gradient exploded and the loss became NaN, which means that the mini-batch gradient variance is heavily affecting the convergence behaviour. One of the reason for such an explosion was due to the increased variance of the mini-batch gradients. As shown in Figure A.2, the variance of mini-batch gradients rapidly increased in the early stages of training and slowly decreased in the later stages. The increased magnitude of the term $\sum_{j=0}^{m-1}\sum_{k=0}^{E-1}\|\nabla C_{ij}(\omega) - \nabla C_{ijk}(\omega)\|^2$ could heavily affect the gradient variance in the early stages.

Therefore, in additional experiments, we switched the value of ε at the late stage of training where the gradient variance is reduced and the training procedure is stable. We switched the value at 200-th round on MNIST and 300-th round on Fashion-MNIST. As a result, though the convergence behaviour became extremely unstable initially after switching, the performance quickly catched up



(a) (b)

Figure A.2: Mini-batch gradient variance during training of (a) MNIST and (b) Fashion-MNIST. Though the variance may rapidly increase initially after the switch, the variance decreases and becomes smaller than the one of FedSAM in later stages.



(a) (b)

Figure A.3: Test accuracy(%) during training of (a) MNIST and (b) Fashion-MNIST. We switched the value of ε to $E\epsilon/2$ at 200-th round for MNIST, 300-th round for Fashion-MNIST. Though the convergence behaviour became extremely unstable initially after switching, the performance quickly catched up to or surpassed the ones of FedSAM.

to or surpassed the ones of FedSAM. In spite of a huge gap between the value of ε and $E\epsilon/2$ in late rounds, the performance was similar or better. The results show that the drifting effect is mitigated more when ε is as big as $E\epsilon/2$.

Through a backward error analysis and experiments, we concluded that FedSAM enhances the convergence behaviour by mildly and gradually removing the drifting term of the modified loss, which is similar to performing an 'interpolation' between FedAvg and FedSGD.

B How a drifting term affects the convergence speed of FedAvg

In this section, we do not explicitly show an exact bound of convergence rate of FedAvg. Instead, we provide a brief overview how a drifting term slows the convergence speed of FedAvg. For ease of analysis, we only consider a situation where all clients participate in all rounds. Ignoring a generalizing term, we consider the following optimization problem:

$$\min_{\omega} \left\{ \tilde{C}(\omega) := C(\omega) - \frac{E\epsilon}{4m} \sum_{j=0}^{m-1} \|\nabla C(\omega) - \nabla C_j(\omega)\|^2 \right\} \tag{33}$$

Then we make a few assumptions on cost functions $C_0, \ldots, C_{m-1}$ for an easier analysis.

Assumption 1. *Cost functions $C_0, \ldots, C_{m-1}$ are all L-smooth: $\|\nabla C_j(x) - \nabla C_j(y)\|_2 \leq L\|x - y\|_2$ for any x, y and $L > 0$.*

Assumption 2. *The learning rate or step size ϵ is bounded: $\epsilon \leq 1/L$.*

Assumption 3. *The gradient norm of the variance of cost functions divided by the gradient norm of the average cost function is bounded:* $\frac{\|\nabla \mathrm{Var}[\nabla C_j(\omega)]\|_2}{\|\nabla C(\omega)\|_2} = \frac{\|\frac{1}{m}\sum_{j=0}^{m-1}\nabla\|\nabla C(\omega)-\nabla C_j(\omega)\|_2^2\|_2}{\|\nabla C(\omega)\|_2} \leq \tilde{\sigma}$.

Overview. Our assumption that cost functions are L-smooth implies that the average cost function C is also L-smooth. This property leads to the following inequality:

$$C(y) \leq C(x) + \langle \nabla C(x), y - x\rangle + \frac{L}{2}\|y - x\|_2^2 \tag{34}$$

If we let $x = \omega$ and $y = \omega^+ = \omega - \epsilon\nabla\tilde{C}(\omega) = \omega - \epsilon\nabla C(\omega) + \frac{E\epsilon^2}{4m}\sum_{j=0}^{m-1}\nabla\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2$, which is a one-step update of gradient descent from ω, we then get

$$\begin{aligned}
C(\omega^+) &= C(\omega - \epsilon\nabla C(\omega) + \frac{E\epsilon^2}{4m}\sum_{j=0}^{m-1}\nabla\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2) \\
&\leq C(\omega) - \langle \nabla C(\omega), \epsilon\nabla C(\omega) - \frac{E\epsilon^2}{4m}\sum_{j=0}^{m-1}\nabla\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2\rangle \\
&+ \frac{L}{2}\|\epsilon\nabla C(\omega) - \frac{E\epsilon^2}{4m}\sum_{j=0}^{m-1}\nabla\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2\|_2^2 \\
&= C(\omega) - \epsilon(1 - \frac{L\epsilon}{2})\|\nabla C(\omega)\|_2^2 + \frac{E\epsilon^2}{4}(1 - L\epsilon)\langle \nabla C(\omega), \frac{1}{m}\sum_{j=0}^{m-1}\nabla\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2\rangle \\
&+ \frac{LE^2\epsilon^4}{32}\|\frac{1}{m}\sum_{j=0}^{m-1}\nabla\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2\|_2^2
\end{aligned} \tag{35}$$

Now an important factor for bounding $C(\omega^+)$ is whether the term $\langle \nabla C(\omega), \frac{1}{m}\sum_{j=0}^{m-1}\nabla\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2\rangle$ is not smaller than 0. To know if it is true, we should know what the term means: it indicates whether $\frac{1}{m}\sum_{j=0}^{m-1}\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2$ increases or decreases if the parameter is one-step-updated following the cost function C. If $\frac{1}{m}\sum_{j=0}^{m-1}\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2$ decreases, the term $\langle \nabla C(\omega), \frac{1}{m}\sum_{j=0}^{m-1}\nabla\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2\rangle$ is larger than 0.

Another thing to notice is that $\frac{1}{m}\sum_{j=0}^{m-1}\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2$ indicates the variance of client gradients. In the experiments by Johnson & Zhang (2013), the gradient variance has decreased during training in the overall perspective, which indicates that $\frac{1}{m}\sum_{j=0}^{m-1}\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2$ will also decrease if the parameter is one-step-updated following the cost function C. This empirical evidence implies that the assumption $\langle \nabla C(\omega), \frac{1}{m}\sum_{j=0}^{m-1}\nabla\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2\rangle \geq 0$ is not implausible at least in intermediate stages of training. In this paper, we do not exactly show that $\frac{1}{m}\sum_{j=0}^{m-1}\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2$ will decrease during training but instead show that the approximation of the gradient variance decreases when the current parameter is far from the optimal point.

$$\begin{aligned}
\langle \nabla C(\omega), \frac{1}{m}\sum_{j=0}^{m-1}\nabla\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2\rangle &= \langle \nabla C(\omega), \frac{1}{m}\sum_{j=0}^{m-1}\nabla\|\nabla C(\omega)\|_2^2 - \nabla\|\nabla C_j(\omega)\|_2^2\rangle \\
&= \frac{2}{m}\sum_{j=0}^{m-1}\langle \nabla C_j(\omega), \nabla\nabla C_j(\omega)\nabla C(\omega)\rangle \\
&- 2\langle \nabla C(\omega), \nabla\nabla C(\omega)\nabla C(\omega)\rangle
\end{aligned} \tag{36}$$

Here, as a rough explanation, we regard the current minimization problem as a maximum likelihood estimation problem and use the Fisher information matrix as an expectation of Hessian, or the current problem can be regarded as a sort of least square minimization problem and we can use a Gauss-

Newton approximation of Hessian. We use $\nabla\nabla C_j(\omega) \approx \nabla C_j(\omega)\nabla C_j(\omega)^T$ then we get

$$\langle \nabla C(\omega), \frac{1}{m}\sum_{j=0}^{m-1} \nabla\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2 \rangle \approx \frac{2}{m}\sum_{j=0}^{m-1} \|\nabla C_j(\omega)\|_2^2 \langle \nabla C_j(\omega), \nabla C(\omega)\rangle - 2\|\nabla C(\omega)\|_2^4 \tag{37}$$

If the current parameter is far enough from the optimal point so that the angle between $\nabla C(\omega)$ and $\nabla C_j(\omega)$ is small, then we can approximate the equation above as

$$\langle \nabla C(\omega), \frac{1}{m}\sum_{j=0}^{m-1} \nabla\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2 \rangle \approx \frac{2}{m}\sum_{j=0}^{m-1} \|\nabla C_j(\omega)\|_2^3 \|\nabla C(\omega)\|_2 - 2\|\nabla C(\omega)\|_2^4 \tag{38}$$

Reminding that $\frac{1}{m}\sum_{j=0}^{m-1} \nabla C_j(\omega) = \nabla C(\omega)$, by Jensen's inequality,

$$\langle \nabla C(\omega), \frac{1}{m}\sum_{j=0}^{m-1} \nabla\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2 \rangle \approx \frac{2}{m}\sum_{j=0}^{m-1} \|\nabla C_j(\omega)\|_2^3 \|\nabla C(\omega)\|_2 - 2\|\nabla C(\omega)\|_2^4 \geq 0 \tag{39}$$

Now we assume that $\langle \nabla C(\omega), \frac{1}{m}\sum_{j=0}^{m-1} \nabla\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2 \rangle \geq 0$, with assumption 2 and 3 we can bound $C(\omega^+)$ as

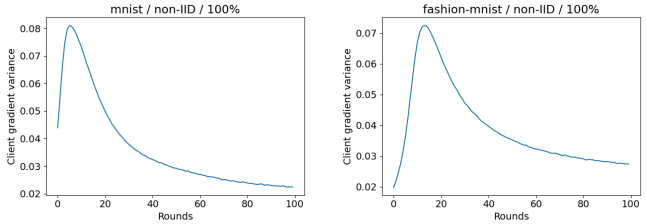
$$\begin{aligned} C(\omega^+) &\leq C(\omega) - \epsilon(1 - \frac{L\epsilon}{2})\|\nabla C(\omega)\|_2^2 + \frac{E\epsilon^2}{4}(1 - L\epsilon)\tilde{\sigma}\|\nabla C(\omega)\|_2^2 + \frac{E^2\epsilon^3}{32}\tilde{\sigma}^2\|\nabla C(\omega)\|_2^2 \\ &= C(\omega) - \epsilon(1 - \frac{L\epsilon}{2} - \frac{E\epsilon}{4}(1 - L\epsilon)\tilde{\sigma} - \frac{E^2\epsilon^2}{32}\tilde{\sigma}^2)\|\nabla C(\omega)\|_2^2 \end{aligned} \tag{40}$$

Compare this bound to the bound of gradient descent:

$$C(\omega^+) \leq C(\omega) - \epsilon(1 - \frac{L\epsilon}{2})\|\nabla C(\omega)\|_2^2 \tag{41}$$

It is possible to observe that the upper bound has become larger and the effective learning rate has decreased due to the presence of the drifting term, which hampers the convergence of FedAvg.

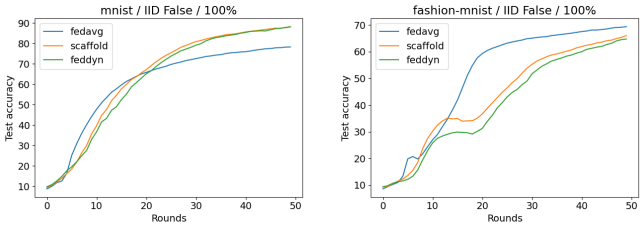
While the assumption of decreasing gradient variance might be viable in the intermediate stages of training, such an assumption can be incorrect in the very early stages of training where the norms of gradients tend to increase rapidly as depicted in Figure A.2. If the variance of gradients increase due to the increasing norm, $\langle \nabla C(\omega), \frac{1}{m}\sum_{j=0}^{m-1} \nabla\|\nabla C(\omega) - \nabla C_j(\omega)\|_2^2 \rangle$ will become negative and FedAvg might show a faster convergence than variance reduction methods. In fact, we were able to observe such a phenomenon in early stages of training during our experiments.



(a) (b)

Figure B.1: The variance of pseudo-gradients from clients. The variance rapidly increases in the initial stage and starts to decrease after a certain number of rounds. This also shows that the assumption of the decreasing gradient variance is viable in real training situations.

We checked the value of the gradient variance during training on MNIST and Fashion-MNIST to examine if there is a correlation between the accuracy of FedAvg and the fluctuation of the gradient variance. Not only were we able to verify that FedAvg can be faster in the early stages of training, but also observe that the time FedAvg starts to become slower coincides with the time the gradient variance starts to decrease. These results indicate that our analysis on the drifting term not only explains why FedAvg overall performs worse than gradient descent, but also explains why and when FedAvg can perform better than other optimization algorithms.



(a) (b)

Figure B.2: Test accuracies of various federated learning methods on (a) MNIST and (b) Fashion-MNIST. It is able to observe that FedAvg converges faster than SCAFFOLD and FedDyn in the very early stages of training. Another thing to observe is that the time FedAvg starts to become slower coincides with the time the gradient variance starts to decrease.

C IMPLICIT REGULARIZER OF CONVENTIONAL VARIANCE REDUCTION METHODS

In this section, we define some of the variables otherwise. Here, the cost function of j-th client is $C_j(\omega)$, while the client is performing a full-batch training with the learning rate ϵ. The average of all cost functions of clients is $C(\omega)$. The training is done for n rounds with m clients, and the number of iterations of each round is E. We assume that all clients participate in training for all rounds.

C.1 IMPLICIT REGULARIZER OF SCAFFOLD

During training of the j-th client at the i-th round, the parameter ω^{ij} is being updated based on the client control variate c_{ij} and the server control variate c_i. First, we need to approximate c_{ij} and c_i, which we will approximate as $\nabla C_j(\omega_0^{(i-1)j})$ and $\nabla C(\omega_0^{(i-1)j})$. Since the first round of SCAFFOLD starts with c_{ij} and c_i as zero which makes the first round equivalent to FedAvg, we consider that $\omega_0^{ij} = \omega_0^{(i-1)j} - E\epsilon\nabla C(\omega_0^{(i-1)j}) + O(E^2\epsilon^2)$ assuming the parameter updates are similar to the ones of FedAvg. It is possible to check that this assumption can be applied to all later rounds when the first round is equivalent to FedAvg. Then we can obtain $\nabla C_j(\omega_0^{(i-1)j})$ with

$$\begin{aligned}\nabla C_j(\omega_0^{(i-1)j}) &= \nabla C_j(\omega_0^{ij} + E\epsilon\nabla C(\omega_0^{(i-1)j}) + O(E^2\epsilon^2)) \\ &= \nabla C_j(\omega_0^{ij} + E\epsilon\nabla C(\omega_0^{ij} + E\epsilon\nabla C(\omega_0^{(i-1)j})) + O(E^2\epsilon^2)) \\ &= \nabla C_j(\omega_0^{ij} + E\epsilon\nabla C(\omega_0^{ij})) + O(E^2\epsilon^2) \end{aligned} \tag{42}$$

Since we will always multiply control variates with $E\epsilon$, we ignore high-order terms and approximate c_{ij} as $\nabla C_j(\omega_0^{ij} + E\epsilon\nabla C(\omega_0^{ij}))$. In the same way, we approximate c_i as $\nabla C(\omega_0^{ij} + E\epsilon\nabla C(\omega_0^{ij}))$. Now the discrete updates of the parameter ω^i during E steps can be expressed step-by-step.

$$\omega_1^{ij} = \omega_0^{ij} - \epsilon(\nabla C_j(\omega_0^{ij}) - c_{ij} + c_i) \tag{43}$$

$$\omega_2^{ij} = \omega_0^{ij} - \epsilon(\nabla C_j(\omega_0^{ij} - \epsilon(\nabla C_j(\omega_0^{ij}) - c_{ij} + c_i)) - c_{ij} + c_i) \tag{44}$$

$$\cdots$$

$$\omega_E^{ij} = \omega_0^{ij} - E\epsilon(\nabla C_j(\omega_0^{ij}) - c_{ij} + c_i) + \frac{E(E-1)}{2}\epsilon^2\nabla\nabla C_j(\omega_0^{ij})(\nabla C_j(\omega_0^{ij}) - c_{ij} + c_i) + O(E^3\epsilon^3) \tag{45}$$

Neglecting $O(E^3\epsilon^3)$ terms, parameter ω_{ij} is expressed as

$$\begin{aligned}\omega_E^{ij} = \omega_0^{ij} &- E\epsilon\nabla C(\omega_0^{ij}) - E^2\epsilon^2\nabla C(\omega_0^{ij})\nabla\nabla(C(\omega_0^{ij}) - C_j(\omega_0^{ij})) \\ &+ \frac{E(E-1)}{2}\epsilon^2\sum_{i=0}^{m-1}\nabla\nabla C_j(\omega_0^{ij})\nabla C(\omega_0^{ij}) + O(E^3\epsilon^3)\end{aligned} \tag{46}$$

After the i-th round, the client parameters are aggregated and form a parameter ω^i.

$$\omega_E^i = \omega_0^i - E\epsilon\nabla C(\omega_0^i) + \frac{E(E-1)}{4}\epsilon^2\nabla\|\nabla C(\omega_0^i)\|^2 + O(E^3\epsilon^3) \tag{47}$$

After n rounds of training, the expectation value of global parameter ω will become as

$$\begin{aligned}\mathbb{E}(\omega_{nE}) &= \omega_0 - nE\epsilon\nabla C(\omega_0) + \frac{n^2E^2\epsilon^2}{4}\nabla(\|\nabla C(\omega_0)\|^2 - \frac{1}{n}\|\nabla C(\omega_0)\|^2 \\ &+ \frac{1}{n}\|\nabla C(\omega_0)\|^2 - \frac{1}{nE}\|\nabla C(\omega_0)\|^2) + O(n^3E^3\epsilon^3) \qquad (48)\\ &= \omega_0 - nE\epsilon\nabla C(\omega_0) + \frac{n^2E^2\epsilon^2}{4}\nabla(\|\nabla C(\omega_0)\|^2 - \frac{1}{nE}\|\nabla C(\omega_0)\|^2) + O(n^3E^3\epsilon^3) \qquad (49)\end{aligned}$$

Then the modified loss under SCAFFOLD is

$$\tilde{C}_{SCAFFOLD}(\omega) = C(\omega) + \frac{\epsilon}{4}\|\nabla C(\omega)\|^2 \tag{50}$$

C.2 Implicit regularizer of FedDyn

In FedDyn, for the j-th client at the i-th round, firstly the local parameter ω^{ij} is updated along the loss function $\mathfrak{R}_j(\omega^{ij}) = C_j(\omega^{ij}) - \langle\nabla C_j(\omega_E^{(i-1)j}), \omega^{ij}\rangle + \frac{\alpha}{2}\|\omega^{ij} - \omega_0^{ij}\|^2$ while $\nabla C_j(\omega_E^{(i-1)j})$ acts as a client control variate and $\omega_{(i-1)E}$ is a global parameter from a previous round. The discrete updates of the parameter ω^{ij} during E steps can be expressed step-by-step.

$$\begin{aligned}\omega_1^{ij} &= \omega_0^{ij} - \epsilon\nabla C_j(\omega_0^{ij}) + \epsilon\nabla C_j(\omega_E^{(i-1)j}) - \epsilon\alpha(\omega_0^{ij} - \omega_0^{ij}) \qquad (51)\\ \omega_2^{ij} &= \omega_1^{ij} - \epsilon\nabla C_j(\omega_0^{ij}) + \epsilon\nabla C_j(\omega_E^{(i-1)j}) - \epsilon\alpha(\omega_1^{ij} - \omega_0^{ij})\\ &= (1-\epsilon\alpha)\omega_1^{ij} + \epsilon\alpha\omega_0^{ij} - \epsilon\nabla C_j(\omega_0^{ij}) + \epsilon\nabla C_j(\omega_E^{(i-1)j}) \qquad (52)\\ &\cdots\\ \omega_E^{ij} &= (1-\epsilon\alpha)\omega_{E-1}^{ij} + \epsilon\alpha\omega_0^{ij} + \epsilon\nabla C_j(\omega_E^{(i-1)j}) - \epsilon\nabla C_j(\omega_{E-1}^{ij}) \qquad (53)\end{aligned}$$

Then, after we organize the equations above, the updated parameter ω_E^{ij} will become

$$\begin{aligned}\omega_E^{ij} &= (1-\epsilon\alpha)^E\omega_0^{ij} + \epsilon\alpha\omega_0^{ij}(1 + \cdots + (1-\epsilon\alpha)^{E-1}) + \sum_{k=0}^{E-1}(1-\epsilon\alpha)^{E-1-k}\epsilon(\nabla C_j(\omega_E^{(i-1)j}) - \nabla C_j(\omega_k^{ij}))\\ &= \omega_0^{ij} + \frac{1-(1-\epsilon\alpha)^E}{\epsilon\alpha}\epsilon\nabla C_j(\omega_E^{(i-1)j}) - \sum_{k=0}^{E-1}(1-\epsilon\alpha)^{E-1-k}\epsilon\nabla C_j(\omega_k^{ij}) \qquad (54)\end{aligned}$$

If we ignore high-order terms, it is possible to express ω_E^{ij} as

$$\begin{aligned}\omega_E^{ij} &= \omega_0^{ij} + (E\epsilon - \frac{E(E-1)}{2}\alpha\epsilon^2)\nabla C_j(\omega_E^{(i-1)j}) - \sum_{k=0}^{E-1}(\epsilon - (E-1-k)\alpha\epsilon^2)\nabla C_j(\omega_k^{ij}) + O(E^3\epsilon^3)\\ &= \omega_0^{ij} + (E\epsilon - \frac{E(E-1)}{2}\alpha\epsilon^2)\nabla C_j(\omega_E^{(i-1)j}) - (\epsilon - (E-0)\alpha\epsilon^2)\nabla C_j(\omega_0^{ij})\\ &- (\epsilon - (E-1)\alpha\epsilon^2)\nabla C_j(\omega_0^{ij} - \epsilon\nabla C_j(\omega_0^{ij}) + \epsilon\nabla C_j(\omega_E^{(i-1)j}) + O(E^2\epsilon^2)) - \ldots\\ &= \omega_0^{ij} - E\epsilon(\nabla C_j(\omega_0^{ij}) - \nabla C_j(\omega_E^{(i-1)j})) + \frac{E(E-1)}{2}\epsilon^2(\alpha(\nabla C(\omega_0^{ij}) - \nabla C_j(\omega_E^{(i-1)j}))\\ &+ \nabla\nabla C(\omega_0^{ij})\nabla C(\omega_0^{ij}) - \nabla\nabla C(\omega_0^{ij})\nabla C_j(\omega_E^{(i-1)j})) + O(E^3\epsilon^3) \qquad (55)\end{aligned}$$

After the client updates, the server-side control variate is added to the aggregated parameter. Here, if the parameter ω^{ij} is sufficiently minimized throughout the round, the server-side control variate h^i becomes the average of local gradients, $\frac{1}{m}\sum_{j=0}^{m-1}\nabla C(\omega_0^{ij})$ (Acar et al., 2021). During the global parameter updates, we set $\alpha = \frac{1}{E\epsilon}$ then the discrete updates of the global parameter ω can be

expressed step-by-step as

$$\omega_E \approx \omega_0 - \frac{E\epsilon}{m}\sum_{j=1}^{m-1}(\nabla C_j(\omega_0) - \nabla C_j(\omega_E^{-1j}) + \nabla C_j(\omega_E^{0j})) + \frac{E(E-1)}{2}\epsilon^2\sum_{j=0}^{m-1}(\alpha(\nabla C(\omega_0) - \nabla C_j(\omega_E^{0j})) + \nabla\nabla C(\omega_0)\nabla C(\omega_0) - \nabla\nabla C(\omega_0)\nabla C_j(\omega_E^{0j})) + O(E^3\epsilon^3) \tag{56}$$

$$\begin{aligned}\omega_{2E} &\approx \omega_E - \frac{E\epsilon}{m}\sum_{j=1}^{m-1}(\nabla C_j(\omega_E) - \nabla C_j(\omega_E^{0j}) + \nabla C_j(\omega_E^{1j})) + \frac{E(E-1)}{2}\epsilon^2\sum_{j=0}^{m-1}(\alpha(\nabla C(\omega_E) - \nabla C_j(\omega_E^{1j})) \\ &+ \nabla\nabla C(\omega_E)\nabla C(\omega_E) - \nabla\nabla C(\omega_E)\nabla C_j(\omega_E^{1j})) + O(E^3\epsilon^3) \\ &= \omega_0 - \frac{E\epsilon}{m}\sum_{j=1}^{m-1}(\nabla C_j(\omega_0) - \nabla C_j(\omega_E^{-1j}) + \cancel{\nabla C_j(\omega_E^{0j})} \\ &+ \nabla C_j(\omega_E) - \cancel{\nabla C_j(\omega_E^{0j})} + \nabla C_j(\omega_E^{1j})) + \dots\end{aligned} \tag{57}$$

...

Ignoring high-order terms, then the expected value of ω_{nE} can be approximated as

$$\omega_{nE} \approx \omega_0 - nE\epsilon\nabla C(\omega_0) - E\epsilon\frac{1}{m}\sum_{j=0}^{m-1}\nabla C_j(\omega_E^{(n-1)j}) + \frac{n^2E^2\epsilon^2}{4}\nabla(\|\nabla C(\omega_0)\|^2 - \frac{1}{n}\|\nabla C(\omega_0)\|^2) + O(n^3E^3\epsilon^3) \tag{58}$$

If the parameter was updated enough, the modified loss can be approximated as

$$\tilde{C}_{FEDDYN}(\omega) \approx C(\omega) + \frac{E\epsilon}{4}\|\nabla C(\omega)\|^2 \tag{59}$$

If the value of α is smaller, FedDyn shows effect as if the learning rate has become larger to $1/\alpha$. With different α, if $1/\alpha$ is small enough, the aggregated parameter ω_{nE} can be approximated as

$$\omega_{nE} \approx \omega_0 - \frac{n}{\alpha}\nabla C(\omega_0) + \frac{n^2}{4\alpha^2}\nabla(\|\nabla C(\omega_0)\|^2 - \frac{1}{n}\|\nabla C(\omega_0)\|^2) + O(\frac{n^3}{\alpha^3}) \tag{60}$$

and the modified loss can be approximated as

$$\tilde{C}_{FEDDYN}(\omega) \approx C(\omega) + \frac{1}{4\alpha}\|\nabla C(\omega)\|^2 \tag{61}$$

However, there is one limitation to this analysis: as in Acar et al. (2021) we assumed that h^i becomes $\frac{1}{m}\sum_{j=0}^{m-1}\nabla C(\omega_0^{ij})$ when a round ends. Such an assumption is satisfied when the local loss function $\mathfrak{R}_j(\omega^{ij}) = C_j(\omega^{ij}) - \langle\nabla C_j(\omega_E^{(i-1)j}), \omega^{ij}\rangle + \frac{\alpha}{2}\|\omega^{ij} - \omega_0^{ij}\|^2$ is sufficiently minimized for a long period. The point of collision is another assumption of our analysis: a small magnitude of $E\epsilon$. A small magnitude of $E\epsilon$ hampers a sufficient minimization of the local loss function $\mathfrak{R}_j(\omega^{ij})$. Though we assumed that h^i becomes $\frac{1}{m}\sum_{j=0}^{m-1}\nabla C(\omega_0^{ij})$ following the assumption of Acar et al. (2021), it makes Equation 61 remain as an approximation of the modified loss of FedDyn.