RE-WEIGHTED SOFTMAX CROSS-ENTROPY TO CONTROL FORGETTING IN FEDERATED LEARNING

Gwen Legate
Concordia University, Mila

Lucas Caccia
McGill University, Mila

Eugene Belilovsky
Concordia University, Mila

ABSTRACT

In Federated Learning a global model is learned by aggregating model updates computed at a set of independent client nodes. To reduce communication costs, multiple gradient steps are performed at each node prior to aggregation. A key challenge in this setting is data heterogeneity across clients resulting in differing local objectives. This can lead clients to overly minimize their own local objective consequently diverging from the global solution. We demonstrate that individual client models experience a catastrophic forgetting with respect to data from other clients and propose an efficient approach that modifies the cross-entropy objective on a per-client basis by re-weighting the softmax logits prior to computing the loss. This approach shields classes outside a client's label set from abrupt representation change and we empirically demonstrate it can alleviate client forgetting and provide consistent improvements to standard federated learning algorithms. Our method is particularly beneficial under the most challenging federated learning settings where data heterogeneity is high and client participation in each round is low.

1 INTRODUCTION

Federated Learning (FL) is a distributed machine learning paradigm in which a shared global model is learned from a decentralized set of data located at a number of independent client nodes (McMahan et al., 2017; Konečnỳ et al., 2016). Driven by communication constraints, FL algorithms typically perform a number of local gradient update steps before synchronizing with the global model. This reduced communication strategy is very effective under independent and identically distributed (i.i.d.) settings, but data heterogeneity across clients has direct implications on the convergence and performance of FL algorithms (Zhao et al., 2018). FL was conceptualized as a learning technique to allow a decentralized group of users to collaboratively train a shared model without sharing sensitive user data but still allowing users to benefit from data stored at other nodes. Under realistic settings this user data will often have non-i.i.d. distributions. For the case of supervised multi-class classification, users may frequently have no data whatsoever from one or several classes. Data in-homogeneity across clients frequently induces client drift, a phenomenon in which clients progress too far towards optimizing their own local objective, leading to a solution that has severely "drifted" from an optimal global solution (Karimireddy et al., 2020).

In continual learning, a model is trained on a number of tasks sequentially and the learner needs to learn each new task without forgetting knowledge obtained from the preceding tasks. The tendency of a continual learning model to forget previously learned information when learning a new task is termed catastrophic forgetting (McCloskey & Cohen, 1989) and it is typically a focus of study in continual learning literature. Similar to FL, data heterogeneity in continual learning presents a challenge since different tasks typically contain data drawn from different underlying distributions. We can draw a connection between the catastrophic forgetting problem in continual learning and the client drift problem in federated learning. We consider one round of federated learning in which K random clients are selected and initialized with a copy of the current global model. Each client performs a pre-determined number of local update steps to optimize the objective on their local data. A round ends with an update to the global model achieved by aggregating the updates from each client. At the beginning of a round, the clients receive a model previously derived from training on other clients data. As local training proceeds, the model becomes increasingly biased towards a given client. As discussed in Lesort (2022); Gupta et al. (2022), this can cause client models to rely on spurious correlations to improve their in-distribution performance, creating a situation where local models experience catastrophic forgetting with respect to data of the other clients (drawn from distinctly different distributions). Naturally, aggregating models that have deviated from a joint solution will lead to degraded results with respect to the global objective. We denote this problem as *local client forgetting*. Reducing local client forgetting will moderate the decrease in performance with respect to other clients data at individual client models and we hypothesize this will increase the ability of local models to better generalize to the distributions of other clients, improving the loss of individual models over the combined

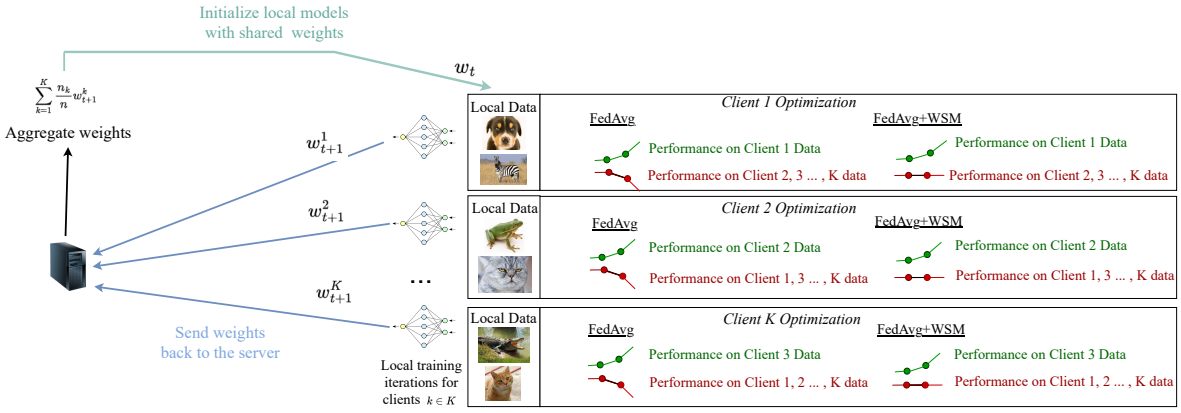

Figure 1: *Illustration of catastrophic forgetting within client rounds.* A global model with knowledge of all classes is sent to all clients participating in a given FL round. Local training increases the client model performance on the client's local distribution but tends to simultaneously decrease performance with respect other clients distributions which leads to poor aggregation and overall model performance.

data. We therefore propose to reduce client drift by tackling local client forgetting. Figure 1 illustrates local client forgetting within a round of federated learning.

There are numerous approaches to tackle catastrophic forgetting in the continual learning literature (Kirkpatrick et al., 2017; Li & Hoiem, 2017; Chaudhry et al., 2019; Schwarz et al., 2018; Davari et al., 2022). Many of these however are largely impractical in the FL setting. Experience Replay methods (Chaudhry et al., 2019) require access to other clients' data, violating the fundamental data communication constraints of FL. Similar concerns exist for many regularization methods such as elastic weight consolidation (EWC) Kirkpatrick et al. (2017) which require communicating additional information. Regularization methods can additionally require many steps to converge due to the additional conflicting objectives (Aljundi et al., 2019). This computational constraint can hurt convergence of the FL algorithm, a key desideratum. For the supervised continual learning setting, Caccia et al. (2022); Ahn et al. (2021) proposed a modification of the standard cross entropy objective function that truncates the softmax denominator, removing terms corresponding to classes from old tasks. A variant of this method inspired by the long-tailed recognition methods (Ren et al., 2020) was also recently introduced in Jodelet et al. (2022). This simple approach mitigates catastrophic forgetting by reducing the bias on the model to avoid predicting old classes.

Inspired by the parallels between client drift in FL and catastrophic forgetting in the continual learning case, we propose an adaptation of the CL method from Caccia et al. (2022); Ahn et al. (2021); Jodelet et al. (2022) to modify the loss function of each client based on its class distribution using a re-weighted softmax. We will empirically demonstrate that this approach can drastically reduce client level forgetting in the heterogeneous setting, leading to substantially improved overall global model convergence and final performance. We apply the re-weighted softmax to baseline methods FedAvg (McMahan et al., 2017), SCAFFOLD (Karimireddy et al., 2020), FedNova (Wang et al., 2020) and FedProx (Li et al., 2020) demonstrating performance improvements in all cases.

2 Related Work

Federated Learning The most commonly used baseline in federated learning is the FedAvg algorithm proposed by McMahan et al. (2017); it permits clients to train multiple local iterations successively thus reducing communication costs. Since the communication cost between two nodes is orders of magnitude larger than the cost between processor and memory on the same node, communication efficiency in federated learning of upmost importance (Konečnỳ et al., 2016). Convergence of FedAvg has been widely studied for both i.i.d. (Stich, 2018; Wang & Joshi, 2018) and non i.i.d. settings (Karimireddy et al., 2020; Li et al., 2020; Fallah et al., 2020; Yu et al., 2019), settings under which client data is heterogeneous offer additional challenges since convergence has been shown to deteriorate as a function of increasing heterogeneity (Li et al., 2020; Hsu et al., 2019).

Non-Heterogeneous Data Partitions and Client Drift One significant challenge encountered when training on decentralized data is data heterogeneity across clients (Konečnỳ et al., 2016; Li et al., 2021). Partitions contain data generated under different conditions, which can reasonably be expected to create different local distributions at each client. For the case of supervised multi-class classification, which is the focus of this work, users may frequently be

missing data from an entire class or multiple classes of the global underlying distribution. During FL training, data at each client are sampled from these local distributions which creates different local objectives. When clients progress too far towards minimizing their own objective, local models drift from one another, degrading the performance of the shared global model and slowing down convergence (Yao et al., 2021; Li et al., 2019; Karimireddy et al., 2020; Diao et al., 2020).

Several attempts have been made to alleviate client drift through various methods. One approach centers around knowledge distillation to regulate local training, (Zhu et al., 2021) and (Lin et al., 2020) ensemble information about the global data distribution and disseminate it to clients via additional models trained at the server. These methods possess the added risk of privacy attacks and while Zhu et al. (2021) take steps to mitigate this risk, their method requires the existence of an unlabeled dataset, which may not be available in all settings. Other approaches attempt to constrain gradient updates from the clients to reduce the impact of client drift. Karimireddy et al. (2020) propose SCAFFOLD, an algorithm to control client drift using control norms to modify client gradients. The control norms estimate the drift at each client and use that estimate to correct the local updates. FedProx (Li et al., 2020) adds a proximal term to the local objective to limit the impact of variation in local updates. This term is weighted with an additional hyper-parameter, μ which must be tuned appropriately. Both SCAFFOLD and FedProx require careful hyper-parameter tuning to be effective (Li et al., 2022). Like SCAFFOLD and FedProx Acar et al. (2021) also estimate client drift and use it as a means to constrain updates. However; their operations take place at the server level and corrections are applied to the server updates thus avoiding SCAFFOLD's use control norms and saving on the inherent communication burden. The general strategy for each of these methods is similar. They attempt to estimate client drift using gradient updates and then constrain the updates to reduce the drift. SCAFFOLD has additionally been shown to have unstable accuracy and has twice the communication burden of FedAvg due to the control norms required for each client Li et al. (2022). Our proposed method modifies the objective functions locally, creating no additional communication burdens and introducing no additional hyper-parameters. It also directly tackles the problem of the underlying distribution shift, unlike the other methods mentioned which attempt to address client heterogeneity through constrained gradient optimization and/or knowledge distillation. Tenison et al. (2022) propose a gradient masking technique that modifies the aggregation of updates on the server side. This method does directly tackle the problem of distribution shift and while it has been shown to be effective at achieving better generalization on non i.i.d. data, our re-weighted softmax is able to take advantage of the structure of the commonly used cross-entropy loss, making it simpler to implement. Additionally, since we modify the objective functions locally, the re-weighted softmax method is compatible with any federated optimization method in the literature.

Continual Learning Continual learning is a process by which tasks are learned sequentially over a period of time and knowledge of previous tasks is retained and leveraged to learn new tasks (Chen & Liu, 2018). Continual learning is made difficult by the fact that neural networks suffer from catastrophic forgetting, in which learning a new task overrides weights learned from past training, thus degrading model performance on previously learned tasks (McCloskey & Cohen, 1989). Several families of methods have been developed to mitigate catastrophic forgetting. The first class of methods are architecture based approaches Schwarz et al. (2018) that attempt to grow or modify an architecture over time to expand its knowledge. In the second class of methods approaches which store some subset of old data for rehearsal are applied (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019; Rebuffi et al., 2017). Finally, a third class of methods regularizes the learned parameters to limit drastic weight changes when learning a new task (Kirkpatrick et al., 2017; Zenke et al., 2017). None of these solutions are widely applicable in a FL setting since they typically require some sort of information sharing across client nodes which is prohibited in the FL framework. A federated continual learning setting has been considered in the literature (Yoon et al., 2021). Here each client in the federated network continuously collects data. Our work on the other hand considers the standard FL setting where each client maintains a fixed set of data and draws connections to a notion of forgetting across clients to motivate a modification of the loss function. Shoham et al. (2019) have used ideas from continual learning to propose FedCurv, based on the EWC algorithm (Kirkpatrick et al., 2017) from continual learning. FedCurv requires sending additional information and is not compatible with all FL methods. Along this line, Xu et al. (2022) also proposed an approach inspired from rehearsal methods, generating pseudo data and adding an additional regularization term. This requires an expensive pseudo data generating procedure and increases the local training time.

3 METHODS

Background In federated optimization, training data is distributed and optimization occurs over K clients with each client $k \in 1, ..., K$ possessing data $\mathbf{X}_k$ drawn from distribution D_k. We define $n_k = |\mathbf{X}_k|$ and $n = \sum_{k=1}^{K} n_k$ for n samples. The data $\mathbf{X}_k$ at each node may be drawn from different distributions and/or may be unbalanced with some clients possessing more training samples than others. The typical objective function for federated optimization is given

by

$$\min_{\mathbf{w}\in\mathbb{R}^d} \sum_{k=1}^{K} \frac{n_k}{n} \mathcal{L}(\mathbf{w}, \mathbf{X}_k), \tag{1}$$

with $\mathcal{L}(\mathbf{w}, \mathbf{X}_k)$ measuring client k's local objective, and $\mathbf{w}$ representing the global parameters.

In this work we will restrict ourselves to the common case where $\mathcal{L}$ is the cross entropy loss. There are many possible variations of FL algorithms. In general, they follow the similar structure to FedAvg (McMahan et al., 2017), which proceeds as follows :

- **Client selection:** for a set of K clients, $K * p$ are selected at each round $\{t_i\}_{i=1}^{T}$, where $0 < p \leq 1$ is a pre-determined proportion of clients.
- **Client updates:** At the beginning of round, client models are initialized with the current weights of the server model. Each client selected for the round performs E local iterations of SGD.
- **Server update:** The weights of the individual client models are aggregated to form an update to the shared global model.

Re-weighted Softmax Cross Entropy Consider a neural network $f : \mathcal{R}^D \rightarrow \mathcal{R}^C$ where C is the total number of classes. The standard cross entropy is given by equation 2 where $y(\mathbf{x})$ is the label of $\mathbf{x}$ and $\mathcal{C}$ is the set of all classes available to the clients.

$$\mathcal{L}_{CE}(\mathbf{X}_k, \mathbf{w}) = -\sum_{\mathbf{x}\in\mathbf{X}_k} \log \frac{\exp(f_{\mathbf{w}}(\mathbf{x})_{y(\mathbf{x})})}{\sum_{c\in\mathcal{C}} \exp(f_{\mathbf{w}}(\mathbf{x})_c)} \tag{2}$$

$$= -\sum_{\mathbf{x}\in\mathbf{X}_k} \Big[f_{\mathbf{w}}(\mathbf{x})_{y(\mathbf{x})} - \log\Big(\sum_{c\in\mathcal{C}} \exp(f_{\mathbf{w}}(\mathbf{x})_c\Big)\Big] \tag{3}$$

One interpretation of this classical loss function considers the two terms as a tightness term (the first term) which brings samples close to their representative classes and a contrast term (the second term) which pushes them apart from other classes (Boudiaf et al., 2020). We note similar loss functions can be interpreted from an energy modeling view (Liu, 2020).

We now modify the standard cross entropy using the re-weighted softmax (WSM) to give our per-client objective function in Equation 4 where $\boldsymbol{\beta_k}$ is a vector containing the proportions of each class present in the client dataset and $\beta_c \in \boldsymbol{\beta_k}$ is the proportion of label c present in the dataset.

$$\mathcal{L}_{WSM}(\mathbf{X}_k, \mathbf{w}) = -\sum_{\mathbf{x}\in\mathbf{X}_k} \Big[f_{\mathbf{w}}(\mathbf{x})_{y(\mathbf{x})} - \log\Big(\sum_{c\in\mathcal{C}} \beta_c \exp(f_{\mathbf{w}}(\mathbf{x})_c)\Big)\Big] \tag{4}$$

If we define $\mathbf{Y}_k$ as the set of labels of samples $\mathbf{X}_k$ belonging to client k then we note that in equation 4 the weighting β introduced in the second term is a function only of $\mathbf{Y}_k$ as opposed to the complete set of labels, $\mathbf{Y}$ in the cross entropy loss from Equation 2. In a highly imbalanced class scenario, as is often studied for FL, many β_c will be zero or contain very small values, thus removing or substantially degrading the contribution of that class to the contrast term. Intuitively, aggressively optimizing $\mathcal{L}_{CE}$ through multiple gradient steps during a client round can lead to a drastic increase in $\mathbb{E}_{x,y\sim D_{j\neq k}}[l_{CE}(\mathbf{x}, \mathbf{y})]$ where D_j are the distribution of clients other than client k. This is because the contrast term encourages classes not present on the client to never be predicted. The re-weighted softmax approach modifies the original local objective function to avoid excessive pressure that drives up the loss of other client data. Indeed, classes not present at the current client are ignored by the local optimization which forces the client to learn by adapting the model's internal representation of the classes present in its training data, rather than abruptly shifting representations of classes outside its training set (Caccia et al., 2022). We will empirically demonstrate that this leads to a reduction in local client forgetting defined below. We note that at test time we use the unweighted softmax as we train a global model that must be able to perform inference on unseen data and clients.

Local client forgetting Here we formalize the notion of local client forgetting discussed in Sec 1 for a multi-class classification problem. Denoting the accuracy on client k's local test data as $Acc_k(\mathbf{w})$, where $\mathbf{w}$ are the model parameters. We define local client forgetting according to equation 5 where $\mathbf{w}_t^i$ refers to the model of client i at

round t after it has completed local training (prior to aggregation) and $\mathbf{w}_{t-1}$ is the global model (after aggregation) at the end of round $t-1$.

$$F_{ki} = Acc_k(\mathbf{w}_{t-1}) - Acc_k(\mathbf{w}_t^i) \tag{5}$$

.

We define an average forgetting for a client k's model according to equation 6

$$F_k = \frac{1}{K-1} \sum_{i \neq k} F_{ki} \tag{6}$$

In what follows we will study these quantities for a standard FL setting.

4 Experiments

In this section, we present the empirical results for the WSM framework. We start by analyzing the notion of forgetting in the context of standard FedAvg, and then show how the application of the re-weighted softmax objective can resolve this. Subsequently, we study how WSM can enhance standard FL algorithms, improving their overall performance.

Datasets and Data Partitions We utilize CIFAR-10, CIFAR-100 (Krizhevsky & Hinton, 2009) and FEMNIST (Caldas et al., 2018) datasets in our experiments. Our primary evaluations consider 100 clients where each client requires their own training and validation sets according to their own unique distribution. To facilitate this, the entire training set is separated into equally sized non-i.i.d. partitions using the Dirichlet distribution parameterized by $\alpha = 0.1$, similar to the method of Hsu et al. (2019). To provide intuition into what client partitions parameterized by $\alpha = 0.1$ look like in practice, Fig. 2 shows the data distributions of ten randomly selected clients. These client partitions are then further separated into training (90%) and validation (10%) sets for each client. For example, 100 clients trained using CIFAR-10 which contains 50 000 training samples would each have 500 of these training samples. Of those 500 samples, 450 would be used for local model updates and 50 would be used exclusively for validation.

Settings We follow the experimental settings of Reddi et al. (2020). Clients are sampled without replacement for each round but can be selected again in subsequent rounds. The fraction of clients sampled is 10% for CIFAR-10 and FEMNIST datasets and 2% for CIFAR-100. Our primary evaluations train a ResNet-18 over 4000 communication rounds for 3 local epochs, using a mini-batch of size 64 and a learning rate of 0.05 for CIFAR-10 and CIFAR-100. FEMNIST, which converges faster due to its large size, is trained for 3000 rounds with all other settings the same as for the CIFAR datasets. We use SGD as our optimizer, with weight decay of 1×10^{-4} following Yao et al. (2021); Hsu et al. (2019). In these experiments, we observe that FedAvg often (though not always) performs better with group normalization as indicated by Hsieh et al. (2020) while FedAvg+WSM is able to perform well with both group and batch normalization, very frequently achieving the best results with batch normalization. We therefore treat the normalization method as a hyper-parameter and provide the best results for both batch and group normalization for both methods at each learning rate.

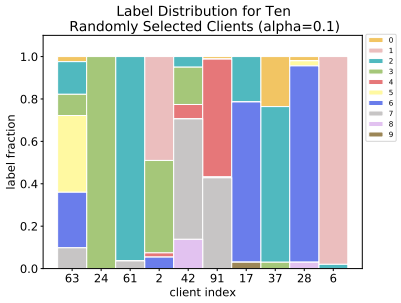

Figure 2: *Visualization of CIFAR-10 client distributions.* We show the data distributions of ten randomly selected clients with data partitioned according to a Dirichlet distribution parameterized by $\alpha = 0.1$.

Validation Throughout the training process the global model is periodically evaluated on the aggregation of the client validation sets to gauge overall training progress, the test set on the other hand is only used at the end of the training process. For lower values of α, as client distributions become more skewed, there can be significant changes in accuracy between training runs (Hsu et al., 2019). Since we focus our analysis on the highly heterogeneous case in which the Dirichlet distributions at each client are parameterized by $\alpha = 0.1$, we observe higher variance in our results. This is especially true for smaller datasets such as CIFAR-10 and CIFAR-100. To mitigate these effects on the validation statistics, we follow the lead of Reddi et al. (2020) and report our final accuracy as the average of the test accuracies taken over the last 100 rounds of training.

4.1 Forgetting During a Federated Round

We first study and build intuition on the effects of forgetting in a federated round for CIFAR-10 dataset. In Figure 3 we show the effects of forgetting during a round of federated learning. The heatmaps on the bottom row of Figure 3 are for round 1200 of training. In the forgetting heatmap we observe a lot of green in the off-diagonal terms indicating that

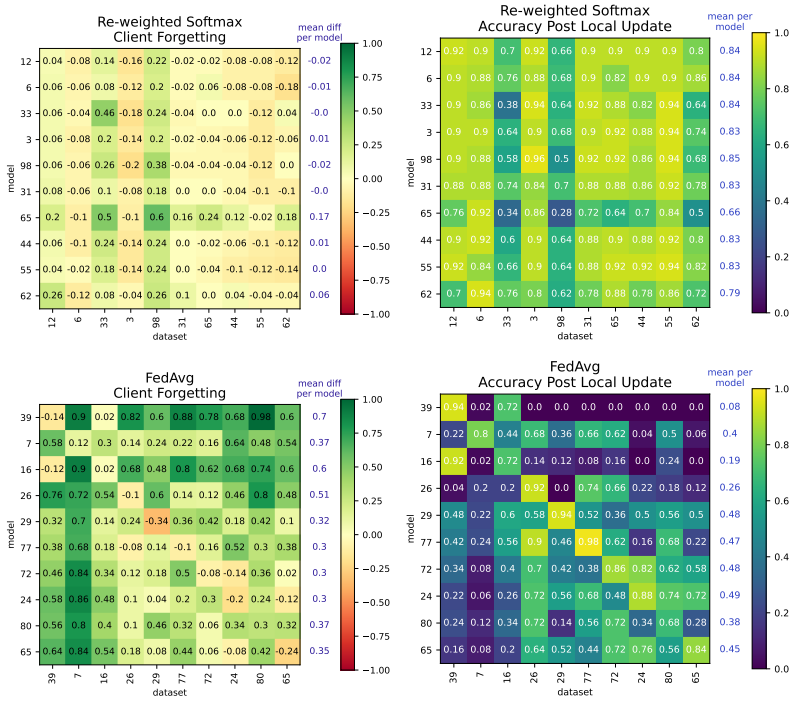

Figure 3: *Illustrating local client forgetting for a given round.* For each heatmap the y-axis indicates the model of client i and the x-axis indicates the local data of client k. In the left column we show the F_{ik} as defined in Eq. 5. Positive values of forgetting (green) indicate high forgetting. The right column show the accuracy of client i's model after training when evaluated on client k's data. We note that in some cases the accuracy can completely collapse on other client's data (particularly when they don't overlap in any classes). We see that FedAvg+WSM (top row) significantly reduces forgetting across clients.

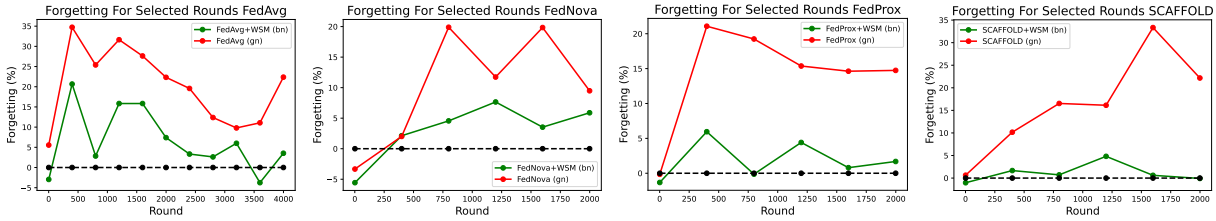

Figure 4: *Average forgetting (F_k) for selected rounds of training* We observe high forgetting for FedAvg (left) which is substantially reduced by applying WSM, especially towards the end of training. FedNova (center left) and FedProx (center right) exhibit less forgetting than FedAvg but still benefit substantially from the application on WSM. SCAFFOLD (right) seems to benefit the most from the application of WSM. Indeed, we observe that WSM produces levels of forgetting close to zero throughout training.

forgetting is very high when using standard cross entropy. The post local update heatmap for standard cross entropy shows a strong trend of better accuracies along the diagonal indicating model i does much better on dataset k when $i = k$, its own dataset.

When using WSM, we observe no preference for better accuracy along the diagonal as is the case for FedAvg without WSM. In fact, we notice lower accuracies are concentrated along the column, indicating a particularly difficult dataset for all local models or along the row, indicating a model that performs badly on all datasets including its own. These observations are supported by the forgetting heatmap for WSM which is predominantly yellow indicating very low forgetting values.

Our results show WSM has the ability to greatly limit the effects of local client forgetting by ignoring classes outside of its data distribution and focusing learning on the classes present. This narrower focus leads to better overall performance after aggregation. The heatmaps are shown for round 1200 of training, approximately a third of the way through training, however; the observation is further confirmed for other rounds as shown in Figure 4 where we plot average forgetting throughout training and by additional heatmaps shown in Appendix A.1. We observe on average that forgetting is high for FedAvg and substantially reduced especially at the end of training for WSM. Having shown that WSM can indeed reduce the local client forgetting, we now study its effect on the aggregated models.

Table 1: Accuracy results of FedAvg with and without WSM for different hyper-parameters. We observe that FedAvg+WSM consistently improves performance over FedAvg over a wider range, as well as having the highest overall accuracy by a substantial margin on the cifar datasets. WSM also makes the learning rate easier to tune since we observe a large hyper-parameter range

Method	lr	*Dataset* CIFAR-10	CIFAR-100	FEMNIST
FedAvg	0.5	0.326 ± 0.098	0.292 ± 0.012	0.542 ± 0.087
FedAvg+WSM (ours)		$\mathbf{0.792 \pm 0.006}$	$\mathbf{0.426 \pm 0.003}$	$\mathbf{0.837 \pm 0.002}$
FedAvg	0.3	0.791 ± 0.013	0.384 ± 0.013	0.769 ± 0.006
FedAvg+WSM (ours)		$\mathbf{0.834 \pm 0.008}$	$\mathbf{0.467 \pm 0.015}$	$\mathbf{0.844 \pm 0.010}$
FedAvg	0.1	0.724 ± 0.027	0.500 ± 0.016	0.835 ± 0.002
FedAvg+WSM (ours)		$\mathbf{0.855 \pm 0.004}$	$\mathbf{0.514 \pm 0.009}$	$\mathbf{0.848 \pm 0.006}$
FedAvg	0.07	0.826 ± 0.007	0.437 ± 0.007	$\mathbf{0.827 \pm 0.006}$
FedAvg+WSM (ours)		$\mathbf{0.856 \pm 0.005}$	$\mathbf{0.553 \pm 0.018}$	0.826 ± 0.019
FedAvg	0.05	0.827 ± 0.004	0.464 ± 0.001	$\mathbf{0.853 \pm 0.004}$
FedAvg+WSM (ours)		$\mathbf{0.858 \pm 0.003}$	$\mathbf{0.564 \pm 0.007}$	0.842 ± 0.005
FedAvg	0.03	0.836 ± 0.005	0.431 ± 0.020	$\mathbf{0.835 \pm 0.006}$
FedAvg+WSM (ours)		$\mathbf{0.857 \pm 0.005}$	$\mathbf{0.581 \pm 0.005}$	0.834 ± 0.003
FedAvg	0.01	0.815 ± 0.003	0.431 ± 0.005	$\mathbf{0.830 \pm 0.002}$
FedAvg+WSM (ours)		$\mathbf{0.845 \pm 0.006}$	$\mathbf{0.574 \pm 0.006}$	0.800 ± 0.019
FedAvg	0.007	0.817 ± 0.007	0.426 ± 0.005	$\mathbf{0.821 \pm 0.011}$
FedAvg+WSM (ours)		$\mathbf{0.841 \pm 0.004}$	$\mathbf{0.568 \pm 0.003}$	0.800 ± 0.007
FedAvg	0.005	0.802 ± 0.010	0.426 ± 0.002	$\mathbf{0.819 \pm 0.022}$
FedAvg+WSM (ours)		$\mathbf{0.826 \pm 0.009}$	$\mathbf{0.554 \pm 0.004}$	0.773 ± 0.013

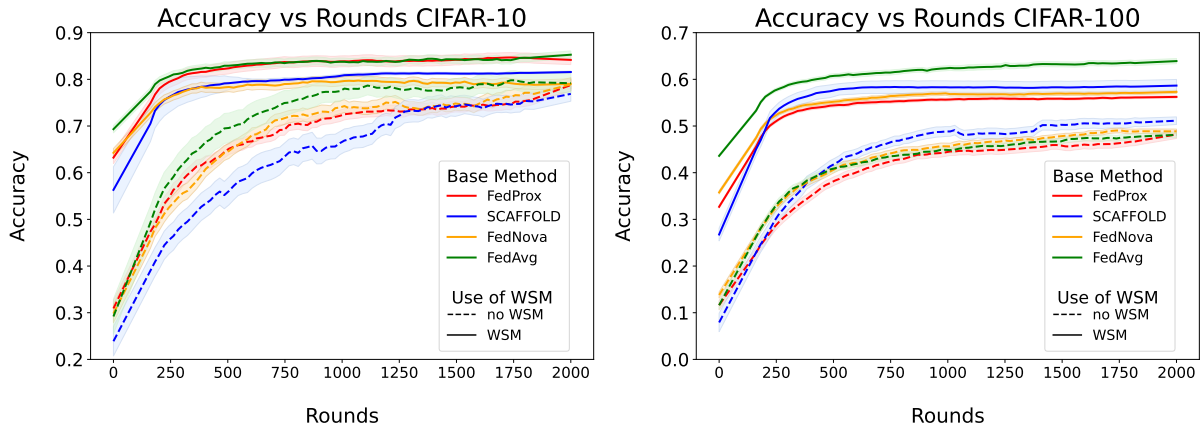

Figure 5: *Convergence plots of different algorithms with and without WSM.* We observe that WSM variants lead to substantially better convergence for all compared methods.

4.2 Evaluation of WSM

In this section we demonstrate how WSM used in combination with FedAvg can improve model performance and convergence. Table 1 shows model performance across a range of learning rates, using CIFAR-10, CIFAR-100, and FEMNIST datasets. The reported values are the average $\pm$ the standard deviation of three seeds. The best preforming model for each learning rate is shown in bold and the best overall result for each dataset is indicated by a green (red) box for WSM (FedAvg). This table demonstrates that WSM has a few benefits to offer during training; it substantially improves performance for both CIFAR datasets with a 2.2% and 1.3% improvement for CIFAR-10 and CIFAR-100, respectively. For FEMNIST, the results show the best performing models are within statistical error of one another. We do however observe strong results using WSM with higher learning rates under which regime we are able to obtain faster convergence. WSM also makes the hyper-parameters easier to tune since it performs well over a large range of learning rates, for example for learning rates between 0.03 and 0.1 FedAvg+WSM has remarkably steady performance between 85.5% to 85.7% while we observe no such consistent performance for vanilla FedAvg.

WSM for Heterogeneous FL methods So far we have demonstrated the ability of WSM to improve model performance for the FedAvg algorithm, now we demonstrate its effectiveness over a range of FL algorithms. In Table 2 we

Table 2: Accuracy results of FedAvg (McMahan et al., 2017), SCAFFOLD (Karimireddy et al., 2020) FedNova (Wang et al., 2020) and FedProx (Li et al., 2020) with and without WSM. We observe that even combined with FL optimization based methods designed to address heterogeneity, WSM provides consistent performance gains. Furthermore, the best performance overall is highlighted in red. We find that although SCAFFOLD and FedPROX can handle heterogeneity better than FedAvg, when combined with WSM FedAvg can obtain the best performance amongst all methods.

Method	CIFAR-10	CIFAR-100
FEDAVG	0.800 ± 0.033	0.494 ± 0.008
FEDAVG+WSM (OURS)	$\mathbf{0.864 \pm 0.008}$	$\mathbf{0.640 \pm 0.002}$
SCAFFOLD	0.794 ± 0.033	0.532 ± 0.003
SCAFFOLD+WSM (OURS)	$\mathbf{0.812 \pm 0.012}$	$\mathbf{0.572 \pm 0.028}$
FEDNOVA	0.780 ± 0.010	0.500 ± 0.005
FEDNOVA+WSM (OURS)	$\mathbf{0.789 \pm 0.13}$	$\mathbf{0.575 \pm 0.002}$
FEDPROX	0.776 ± 0.028	0.487 ± 0.10
FEDPROX+WSM (OURS)	$\mathbf{0.818 \pm 0.023}$	$\mathbf{0.562 \pm 0.005}$

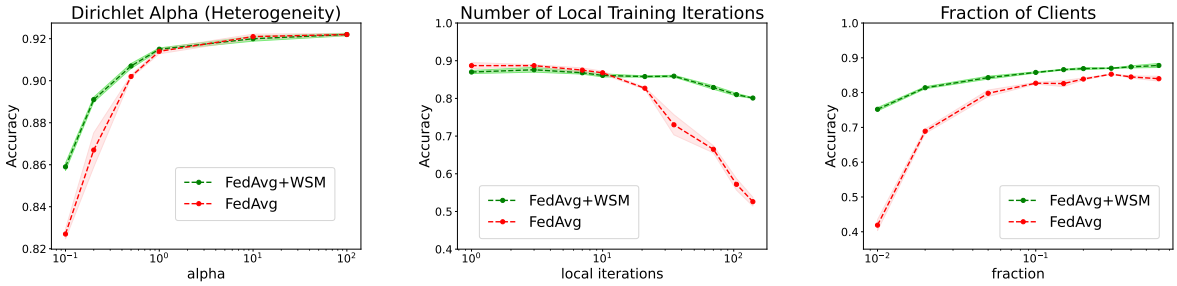

Figure 6: *Plots of ablations of dataset heterogeneity, number of local iterations and the fraction of clients selected at each.* We observe WSM provides performance increases under most of the conditions studied with the most significant advantages provided in scenarios with more heterogeneous client data and smaller fractions of clients are selected for training at each round.

apply WSM to SCAFFOLD, Fed Nova and FedProx. SCAFFOLD and FedProx are constrained optimization methods, specifically designed to address the problem of data heterogeneity, Fed Nova is also designed to improve performance on heterogeneous data. In these experiments we use 30 clients and train the models over only 2000 rounds, since some of these algorithms have a high overhead and many hyper-parameters to search. We keep the same local batch size of 64 and 3 local iterations from the base case. Combining each of these methods with WSM we confirm our hypothesis that improving local client forgetting with WSM provides improvement over the base cases for each method, in most cases this improvement is substantial. From Figure 5 we remark that combining WSM with a baseline method generally provides a stronger performance from the very beginning of training since for both CIFAR-10 and CIFAR-100 the curves showing the training progression for the methods combined with WSM all start off with higher reported accuracy than the baselines and this improvement in performance persists for the duration of training. Work on critical learning periods, where critical learning periods are defined as the early epochs of a training regime, have shown they can determine the final quality of a deep neural network for traditional ML methods (Jastrzębski et al., 2018; Achille et al., 2017; Yan et al., 2021) investigate critical learning periods in the FL setting and discover they do indeed exist consistently in FL. We can thus hypothesize that the early training advantage we see when applying WSM may be having a positive impact on its consistent ability to outperform other FL algorithms it is applied to.

The ease of application of WSM is a significant advantage. We note that even for algorithms that are already partially addressing the heterogeneity problem it can provide benefits. in Table 2 we observe that combining WSM with FedAvg provides the best overall results suggesting that algorithms based on constrained optimization, e.g. SCAFFOLD, may over constrain the improvement possible over a given round.

4.3 ABLATIONS

We now further study the behavior of WSM in combination with FedAvg under different data distributions, client participation settings, different numbers of local iterations and different model architectures. In Figure 6 ablations for the CIFAR-10 setting are shown where we ablate one setting at a time. Except for where we specify the value of the parameter being ablated, the hyper-parameters for the ablation studies are the same as for our base case described in section 4. We observe that under conditions of low α (high heterogeneity) and low fractions of participating clients, the use of WSM is particularly advantageous.

Table 3: Accuracy results of FedAvg with and without WSM for different settings of client learning rates using the LeNet architecture.

	CIFAR-10	CIFAR-100
FEDAVG	0.608 ± 0.010	0.198 ± 0.004
FEDAVG+WSM (OURS)	$\mathbf{0.624 \pm 0.009}$	$\mathbf{0.274 \pm 0.007}$

Parameter α of the Dirichlet Distribution The Dirichlet distribution is parameterized by α, as $\alpha \to 0$ the client distributions will become increasingly heterogeneous and as $\alpha \to \infty$, each client data edges closer towards the same i.i.d. distribution. In our base case we use $\alpha = 0.1$ and we now investigate how model performance is affected by changing the heterogeneity of the data partitions. Similar to the work of Hsu et al. (2019), we investigate $\alpha = \{0.01, 0.1, 0.2, 0.5, 1, 10, 100\}$, for a more explicit indication of how client distributions change as a function of α we refer the reader to section A.5. From Figure 6 we observe WSM has a more significant effect as α decreases and the largest margin of improvement over vanilla FedAvg occurs when $\alpha = 0.01$. As the data becomes increasingly homogeneous, the gap between cross entropy with and without WSM shrinks until the data is i.i.d. and the two methods are equivalent. From Figure 10 we see that by the time $\alpha = 0.5$ most of the clients possess all of the classes, albeit in very skewed proportions. This is also the point at which the performance of FedAvg and FedAvg+WSM become very close. While WSM continues to offer a performance increase over the entire range of α except for $\alpha = 100$, we conclude WSM is most advantageous when clients labels distributions are imbalanced.

Fraction of Participating Clients One of the characteristics of FL is that data is massively distributed and nodes may have limited communication with the central server as nodes may be offline or have slow connections limiting their communication (McMahan et al., 2017). The direct result of this limited connectivity is that only a small number of client nodes may participate in updates at any given time. These ablations focus on the fraction of clients participating in an update round. For each experiment the fraction of participating clients is constant for all rounds of training and we explore fractions from 1%, where only one client participates in the update, up to 60%. For the fraction of participating clients p, we observe the largest performance gap between FedAvg+WSM and FedAvg when the number of participating clients is low. This feature is significant since as explained above, low client participation is a known feature of real world federated learning settings. We hypothesize the performance gap as a function of client fraction between the FedAvg and FedAvg+WSM is due to the larger impact of local client forgetting when we limit communication capability. Unless we actively take steps to control forgetting, non-participating clients will have their data distributions forgotten because unlike participating clients, they will be unable to contribute their updates to the global model. As the fraction of clients selected at each round increases, we observe the performance gap between the two methods narrow since more clients will have the opportunity to be selected at each round and "remind" the model of their data distributions.

Local Iterations A local iteration is defined as one gradient step on the client during a federated round. In our reference setup 7 local iterations is equivalent to one local epoch. After 21 local iterations we observe a sharp decrease in accuracy for FedAvg as local iterations increase. While FedAvg+WSM also experiences a drop in accuracy with respect to increasing local iterations, this drop in accuracy is much less pronounced. This result is in line with our expectations; since WSM helps to alleviate the effects of local client forgetting, increasing the number of local training iterations will cause the effects of local client forgetting become more serious. The ability of WSM to allow the local models to train for more local iterations without forgetting has important implications for the communication costs in FL. Since communication is a serious bottleneck and is limited by client drift, our method offers a valuable option to speed up training in a federated setting.

Architecture We investigate the performance of WSM using the LeNet architecture (LeCun et al., 1998) on CIFAR-10 and CIFAR-100. While this model is not the considered state-of-the-art on these CIFAR datasets, it allows us to eliminate the dependence on the normalization as a hyper-parameter to investigate relative performance for the purposes of this investigation. As before we train for 4000 rounds with each client training for three local epochs. The data is divided between clients using a Dirichlet distribution parameterized by $\alpha = 0.1$. The reported values are the result of the average of three different seeds with the standard deviation indicating the variation between runs. Experiments were performed across a range of learning rates and the results of the best performing models with and without WSM are shown in Table 3. The complete set of results across all learning rates is provided in the appendix, section A.3. The difference between the best performing models of FedAvg and and FedAvg+WSM is 1.02% for CIFAR-10 and an impressive 7.6% for CIFAR-100.

5 CONCLUSION

In this paper we take a deeper look at the *local client forgetting* problem and show that when a client performs local updates during FL it risks overly optimizing its local objective leading to forgetting on other subsets of data. Local client

forgetting degrades the performance of the global model and we show this phenomenon is especially severe in cases where there is a significant distribution mismatch across clients. First making the connection with the catastrophic forgetting problem in the continual learning, we propose a client level modification of the objective function which we call the weighted softmax. We show empirically that WSM allows us to mitigate client level forgetting by demonstrating improved performance for the FedAvg algorithm when combined with WSM across a range of learning rates. We demonstrate these improvements are not limited to FedAvg since we also observe significant performance increases when WSM is applied to SCAFFOLD, FedNova and FedProx. An ablation study demonstrated WSM is particularly effective in the regime of highly heterogeneous client datasets and/or when a small percentage of clients are selected at each round. Our results indicate that addressing local client forgetting in general is an important consideration for federated learning optimization, one that bears closer scrutiny.

REFERENCES

Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263*, 2021.

Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep neural networks. *arXiv preprint arXiv:1711.08856*, 2017.

Hongjoon Ahn, Jihwan Kwak, Subin Lim, Hyeonsu Bang, Hyojun Kim, and Taesup Moon. Ss-il: Separated softmax for incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 844–853, 2021.

Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/15825aee15eb335cc13f9b559f166ee8-Paper.pdf.

Malik Boudiaf, Jérôme Rony, Imtiaz Masud Ziko, Eric Granger, Marco Pedersoli, Pablo Piantanida, and Ismail Ben Ayed. A unifying mutual information view of metric learning: cross-entropy vs. pairwise losses. In *European conference on computer vision*, pp. 548–564. Springer, 2020.

Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=N8MaByOzUfb.

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.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

Zhiyuan Chen and Bing Liu. Lifelong supervised learning. In *Lifelong Machine Learning*, pp. 33–74. Springer, 2018.

MohammadReza Davari, Nader Asadi, Sudhir Mudur, Rahaf Aljundi, and Eugene Belilovsky. Probing representation forgetting in supervised and unsupervised continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16712–16721, 2022.

Enmao Diao, Jie Ding, and Vahid Tarokh. Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *arXiv preprint arXiv:2010.01264*, 2020.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

Sharut Gupta, Kartik Ahuja, Mohammad Havaei, Niladri Chatterjee, and Yoshua Bengio. FL games: A federated learning framework for distribution shifts. In *Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022)*, 2022. URL https://openreview.net/forum?id=UyfdXCeelfy.

Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip Gibbons. The non-iid data quagmire of decentralized machine learning. In *International Conference on Machine Learning*, pp. 4387–4398. PMLR, 2020.

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, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. On the relation between the sharpest directions of dnn loss and the sgd step length. *arXiv preprint arXiv:1807.05031*, 2018.

Quentin Jodelet, Xin Liu, and Tsuyoshi Murata. Balanced softmax cross-entropy for incremental learning with and without memory. *Computer Vision and Image Understanding*, 225:103582, 2022.

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.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

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.

Timothée Lesort. Continual feature selection: Spurious features in continual learning. *arXiv preprint arXiv:2203.01012*, 2022.

Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. A survey on federated learning systems: vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pp. 965–978. IEEE, 2022.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. *Advances in Neural Information Processing Systems*, 33:2351–2363, 2020.

Weitang et al. Liu. Energy-based out-of-distribution detection. *Neural information processing systems*, 2020.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

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.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

Jiawei Ren, Cunjun Yu, Xiao Ma, Haiyu Zhao, Shuai Yi, et al. Balanced meta-softmax for long-tailed visual recognition. *Advances in neural information processing systems*, 33:4175–4186, 2020.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pp. 4528–4537. PMLR, 2018.

Neta Shoham, Tomer Avidor, Aviv Keren, Nadav Israel, Daniel Benditkis, Liron Mor-Yosef, and Itai Zeitak. Overcoming forgetting in federated learning on non-iid data. *arXiv preprint arXiv:1910.07796*, 2019.

Sebastian U Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

Irene Tenison, Sai Aravind Sreeramadas, Vaikkunth Mugunthan, Edouard Oyallon, Eugene Belilovsky, and Irina Rish. Gradient masked averaging for federated learning. *arXiv preprint arXiv:2201.11986*, 2022.

Jianyu Wang and Gauri Joshi. Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms. *arXiv preprint arXiv:1808.07576*, 2018.

Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.

Chencheng Xu, Zhiwei Hong, Minlie Huang, and Tao Jiang. Acceleration of federated learning with alleviated forgetting in local training. *arXiv preprint arXiv:2203.02645*, 2022.

Gang Yan, Hao Wang, and Jian Li. Critical learning periods in federated learning. *arXiv preprint arXiv:2109.05613*, 2021.

Dezhong Yao, Wanning Pan, Yutong Dai, Yao Wan, Xiaofeng Ding, Hai Jin, Zheng Xu, and Lichao Sun. Local-global knowledge distillation in heterogeneous federated learning with non-iid data. *arXiv preprint arXiv:2107.00051*, 2021.

Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. In *International Conference on Machine Learning*, pp. 12073–12086. PMLR, 2021.

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.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pp. 3987–3995. PMLR, 2017.

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.

Zhuangdi Zhu, Junyuan Hong, and Jiayu Zhou. Data-free knowledge distillation for heterogeneous federated learning. In *International Conference on Machine Learning*, pp. 12878–12889. PMLR, 2021.

A Appendix

A.1 Additional Forgetting studies

In this section we show an additional heatmaps, similar to those from figure 3. The heatmaps shown in figure 3 are for round 1200, $\frac{3}{10}$ of the way through training. Figures 8, 7, 9 show heatmaps for after round 1, at the very beginning of training, round 3600, $\frac{9}{10}$ of the way through training and round 4000, the last round of training. The heatmaps are structured the same way as for figure 3 where the y-axis indicates the model of client i and the x-axis indicates the local data of client k. These additional heatmaps illustrate that local client forgetting and the ability of WSM to reduce its effects are present throughout the training cycle.

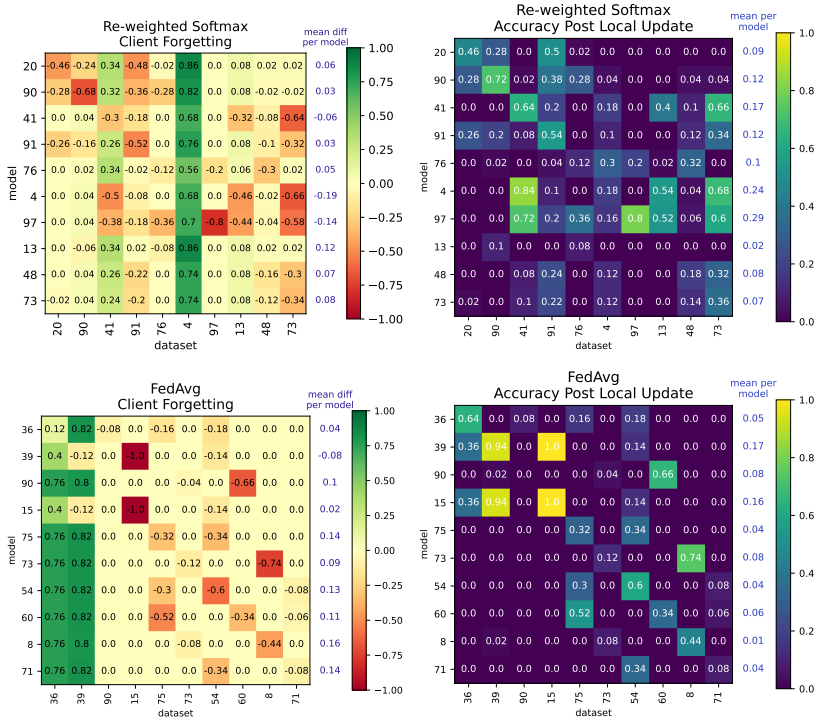

Figure 7: Local client forgetting after round 1 with and without WSM. We note that after the first round of training the model has not converged significantly so the advantage conferred by WSM is not yet as apparent as it will be in future rounds, the point we wish to illustrate here is that the effect of local client forgetting is apparent from the beginning of training

A.2 Complete Result Set Across Learning Rates and Normalization Methods

Table 4 shows the complete set of results across learning rates and normalization methods. We observe FedAvg+WSM has a strong performance with batch normalization contrary to the findings of Hsieh et al. (2020) while FedAvg preforms better using group norm.

A.3 LeNet Performance Across Multiple Learning Rates

Table 5 shows model performance using the LeNet architecture across learning rates. WSM outperforms vanilla FedAvg for each learning rate for both datasets. As with the ResNet-18 case, we continue to observe that WSM provides good performance over a larger range of learning rates than FedAvg which makes it easier to tune. We also observe WSM reduces the variance in model accuracy as evidenced by the typically lower standard deviations reported for the runs in each set.

A.4 WSM for Personalization

We note that in the setting of personalization we can use the WSM directly at inference time, utilizing the client's training data proportion. To illustrate this scenario we consider a WSM trained model with the setting of Sec. 4.2 (high heterogeneity) and demonstrate we can improve the performance by 9.5% at no additional cost under the personalization setting using this strategy.

A.5 Illustrating different Dirichlet parameters

Figure 10 offers a practical illustration of how client partitions change as a function of α. Clients with $\alpha = 0.01$ have only a small percentage of the classes while at $\alpha = 100$ clients have all 10 classes in proportions that are much more equal than we observe with the other two parameterizations.

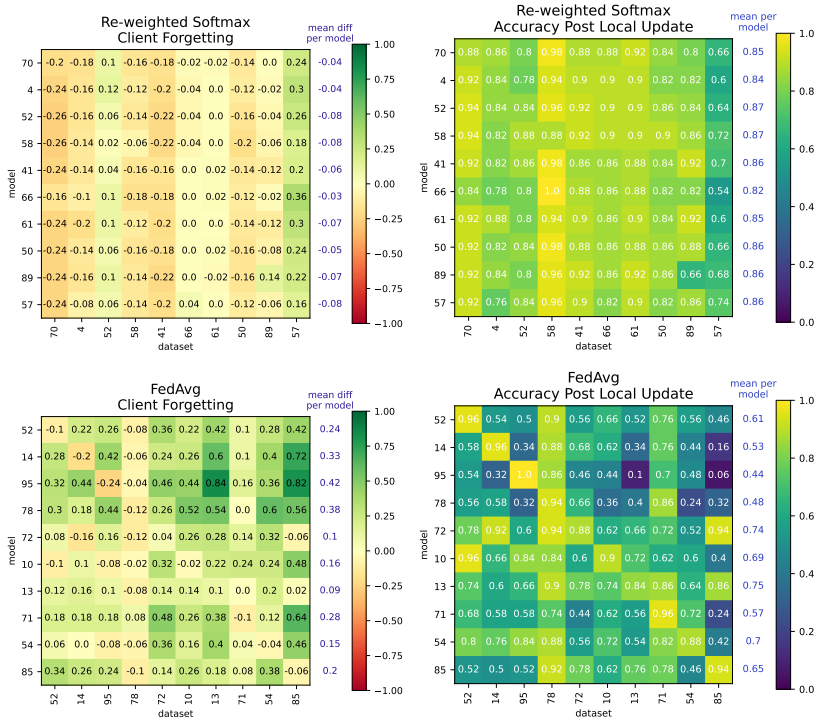

Figure 8: We show local client forgetting for round 3600 with and without WSM. Again at round 3600, much later in training than round 1200 shown in figure 3 we observe FedAvg+WSM (top row) significantly reduces forgetting across clients.

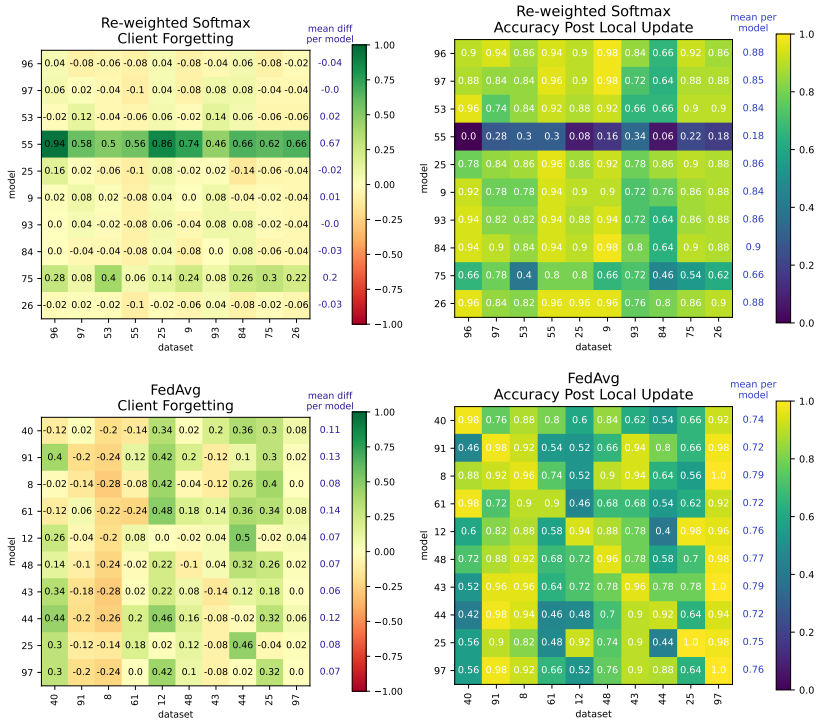

Figure 9: Local client forgetting for round 4000, the last round of training with and without WSM. Here we observe that in the last round of training local client forgetting still occurs for FedAvg. FedAvg+WSM on the other hand has relatively neutral levels of forgetting (close to 0) with the exception of client 55's model which appears to have had a bad round of training in which it forgot quite a bit of relevant information. In this case we point out that forgetting here has occurred equally across all client datasets including it's own indicating the training failure here is not due to local client forgetting.

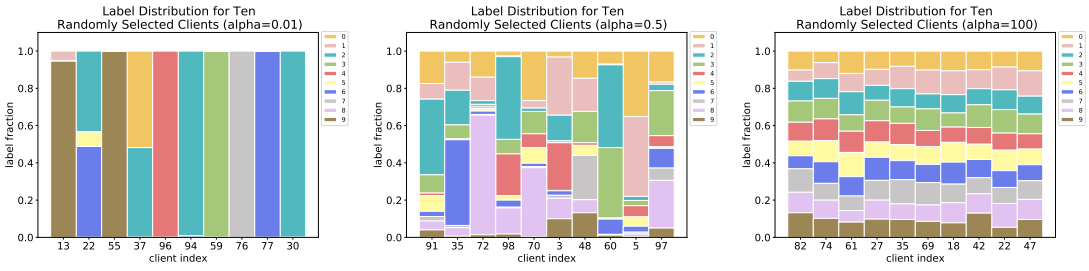

Figure 10: Percentages of each class label for ten randomly selected clients with $\alpha = 0.01, 0.5, 100$ from left to right

Table 4: Accuracy results of FedAvg with and without WSM for different hyper-parameters. We observe that FedAvg+WSM with batch normalization consistently improves performance over FedAvg, as well as having the highest overall accuracy by a large margin. WSM also makes the learning rate easier to tune since we observe a large hyper-parameter range

Method	*Hyper-params* lr	norm	*Dataset* CIFAR-10	CIFAR-100	FEMNIST
FedAvg	0.5	group	0.326 ± 0.098	0.292 ± 0.012	0.542 ± 0.087
FedAvg+WSM (ours)			0.452 ± 0.173	0.234 ± 0.191	0.418 ± 0.177
FedAvg		batch	0.742 ± 0.004	0.386 ± 0.003	0.812 ± 0.020
FedAvg+WSM (ours)			$\mathbf{0.792 \pm 0.006}$	$\mathbf{0.426 \pm 0.003}$	$\mathbf{0.837 \pm 0.002}$
FedAvg	0.3	group	0.791 ± 0.013	0.384 ± 0.013	0.769 ± 0.006
FedAvg+WSM (ours)			0.744 ± 0.007	0.388 ± 0.027	0.761 ± 0.118
FedAvg		batch	0.742 ± 0.004	0.412 ± 0.014	0.815 ± 0.008
FedAvg+WSM (ours)			$\mathbf{0.834 \pm 0.008}$	$\mathbf{0.467 \pm 0.015}$	$\mathbf{0.844 \pm 0.010}$
FedAvg	0.1	group	0.724 ± 0.027	0.500 ± 0.016	0.835 ± 0.002
FedAvg+WSM (ours)			0.794 ± 0.022	0.446 ± 0.004	0.827 ± 0.012
FedAvg		batch	0.820 ± 0.006	0.442 ± 0.016	0.806 ± 0.031
FedAvg+WSM (ours)			$\mathbf{0.855 \pm 0.004}$	$\mathbf{0.514 \pm 0.009}$	$\mathbf{0.848 \pm 0.006}$
FedAvg	0.07	group	0.826 ± 0.007	0.437 ± 0.007	$\mathbf{0.827 \pm 0.006}$
FedAvg+WSM (ours)			0.805 ± 0.007	0.484 ± 0.041	0.823 ± 0.006
FedAvg		batch	0.787 ± 0.006	0.513 ± 0.006	0.789 ± 0.003
FedAvg+WSM (ours)			$\mathbf{0.856 \pm 0.005}$	$\mathbf{0.553 \pm 0.018}$	0.826 ± 0.019
FedAvg	0.05	group	0.827 ± 0.004	0.464 ± 0.001	$\mathbf{0.853 \pm 0.004}$
FedAvg+WSM (ours)			0.791 ± 0.019	0.454 ± 0.015	0.841 ± 0.002
FedAvg		batch	0.790 ± 0.012	0.531 ± 0.007	0.833 ± 0.024
FedAvg+WSM (ours)			$\mathbf{0.858 \pm 0.003}$	$\mathbf{0.564 \pm 0.007}$	0.842 ± 0.005
FedAvg	0.03	group	0.836 ± 0.005	0.431 ± 0.020	$\mathbf{0.835 \pm 0.006}$
FedAvg+WSM (ours)			0.774 ± 0.035	0.472 ± 0.007	0.830 ± 0.006
FedAvg		batch	0.779 ± 0.028	0.561 ± 0.010	0.756 ± 0.015
FedAvg+WSM (ours)			$\mathbf{0.857 \pm 0.005}$	$\mathbf{0.581 \pm 0.005}$	0.834 ± 0.003
FedAvg	0.01	group	0.815 ± 0.003	0.431 ± 0.005	$\mathbf{0.830 \pm 0.002}$
FedAvg+WSM (ours)			0.785 ± 0.011	0.471 ± 0.010	0.800 ± 0.018
FedAvg		batch	0.787 ± 0.003	0.566 ± 0.009	0.744 ± 0.011
FedAvg+WSM (ours)			$\mathbf{0.845 \pm 0.006}$	$\mathbf{0.574 \pm 0.006}$	0.800 ± 0.019
FedAvg	0.007	group	0.817 ± 0.007	0.426 ± 0.005	$\mathbf{0.821 \pm 0.011}$
FedAvg+WSM (ours)			0.773 ± 0.014	0.476 ± 0.010	0.794 ± 0.006
FedAvg		batch	0.797 ± 0.008	$\mathbf{0.568 \pm 0.003}$	0.734 ± 0.018
FedAvg+WSM (ours)			$\mathbf{0.841 \pm 0.004}$	$\mathbf{0.568 \pm 0.003}$	0.800 ± 0.007
FedAvg	0.005	group	0.802 ± 0.010	0.426 ± 0.002	$\mathbf{0.819 \pm 0.022}$
FedAvg+WSM (ours)			0.768 ± 0.019	0.474 ± 0.006	0.781 ± 0.017
FedAvg		batch	0.783 ± 0.009	0.553 ± 0.005	0.743 ± 0.053
FedAvg+WSM (ours)			$\mathbf{0.826 \pm 0.009}$	$\mathbf{0.554 \pm 0.004}$	0.773 ± 0.013

Table 5: Accuracy results of FedAvg with and without WSM for different settings of client learning rates using the LeNet architecture.

	lr	CIFAR-10	CIFAR-100
FedAvg		0.589 ± 0.012	0.119 ± 0.002
FedAvg+WSM (ours)	0.07	$\mathbf{0.624 \pm 0.009}$	$\mathbf{0.180 \pm 0.013}$
FedAvg		0.580 ± 0.051	0.154 ± 0.011
FedAvg+WSM (ours)	0.05	$\mathbf{0.615 \pm 0.004}$	$\mathbf{0.217 \pm 0.009}$
FedAvg		0.605 ± 0.014	0.168 ± 0.001
FedAvg+WSM (ours)	0.03	$\mathbf{0.621 \pm 0.009}$	$\mathbf{0.251 \pm 0.007}$
FedAvg		0.608 ± 0.010	0.168 ± 0.044
FedAvg+WSM (ours)	0.01	$\mathbf{0.624 \pm 0.009}$	$\mathbf{0.274 \pm 0.007}$
FedAvg		0.591 ± 0.014	0.195 ± 0.004
FedAvg+WSM (ours)	0.007	$\mathbf{0.620 \pm 0.022}$	$\mathbf{0.274 \pm 0.001}$
FedAvg		0.585 ± 0.019	0.198 ± 0.004
FedAvg+WSM (ours)	0.005	$\mathbf{0.615 \pm 0.003}$	$\mathbf{0.270 \pm 0.001}$
FedAvg		0.545 ± 0.018	0.181 ± 0.009
FedAvg+WSM (ours)	0.003	$\mathbf{0.566 \pm 0.001}$	$\mathbf{0.247 \pm 0.004}$
FedAvg		0.441 ± 0.019	0.115 ± 0.007
FedAvg+WSM (ours)	0.001	$\mathbf{0.478 \pm 0.009}$	$\mathbf{0.186 \pm 0.003}$