Improving Generalization in Federated Learning by Seeking Flat Minima

Debora Caldarola*[1], Barbara Caputo[1,2], and Marco Ciccone*[1]

[1]Politecnico di Torino, [2]CINI
name.surname@polito.it

Appendix

A Background

In this section, we briefly review the details of Sharpness-Aware Minimization (SAM) [7], its adaptive version (ASAM) [16] and Stochastic Weight Averaging (SWA) [15].

A.1 SAM and ASAM: Overview

SAM aims at finding the solution θ surrounded by a neighborhood having uniform low training loss $\mathcal{L}_{\mathcal{D}}(\theta)$, *i.e.* located in a flat minimum. The *sharpness* of a training loss function is defined as:

$$\max_{||\epsilon||_p \leq \rho} \mathcal{L}_{\mathcal{D}}(\theta + \epsilon) - \mathcal{L}_{\mathcal{D}}(\theta) \tag{1}$$

where ρ is an hyper-parameter defining the neighborhood size and $p \in [1, \infty)$. SAM aims at minimizing the sharpness of the loss solving the following minmax objective:

$$\min_{\theta \in \mathbb{R}^d} \max_{||\epsilon||_p \leq \rho} \mathcal{L}_{\mathcal{D}}(\theta + \epsilon) + \lambda ||\theta||_2^2 \tag{2}$$

where λ is a hyper-parameter weighing the importance of the regularization term. In [7], it is shown that $p = 2$ is typically the optimal choice, hence, without loss of generality, we use the ℓ_2-norm in the maximization over ϵ and omit the regularization term for simplicity. In order to obtain the exact solution of the inner maximization problem $\epsilon^* \triangleq \arg\max_{||\epsilon||_2 \leq \rho} \mathcal{L}(\theta + \epsilon)$, the authors propose to employ a first-order approximation of $\mathcal{L}(\theta + \epsilon)$ around 0:

$$\epsilon^* \approx \arg \max_{||\epsilon||_2 \leq \rho} \mathcal{L}_{\mathcal{D}}(\theta) + \epsilon^T \nabla_\theta \mathcal{L}_{\mathcal{D}}(\theta) = \rho \frac{\nabla_\theta \mathcal{L}_{\mathcal{D}}(\theta)}{||\nabla_\theta \mathcal{L}_{\mathcal{D}}(\theta)||_2} =: \hat{\epsilon}(\theta) \tag{3}$$

Under this computationally efficient approximation, $\hat{\epsilon}(\theta)$ is nothing more than a scaled gradient of the current parameters θ. The sharpness-aware gradient is then defined as $\nabla_\theta \mathcal{L}_{\mathcal{D}}(\theta)|_{\theta + \hat{\epsilon}(\theta)}$ and used to update the model as

$$\theta_{t+1} \leftarrow \theta_t - \gamma \nabla_{\theta_t} \mathcal{L}_{\mathcal{D}}(\theta_t)|_{\theta_t + \hat{\epsilon}_t}, \tag{4}$$

* Equal contribution

where γ is an appropriate learning rate and $\hat{\epsilon}_t = \hat{\epsilon}(\theta_t)$. This two-steps procedure is iteratively applied to solve Eq. 2. Intuitively, SAM performs a first step of gradient ascent to estimate the point $(\theta_t + \hat{\epsilon}_t)$ at which the loss is approximately maximized and then applies gradient descent at θ_t using the just computed gradient.

ASAM In [16], the authors point out that sharpness defined in a rigid region with a fixed radius ρ (Eq. 1) is sensitive to parameter re-scaling, negatively affecting the connection between sharpness and generalization gap. If A is a scaling operator acting on the parameters space without changing the loss function, two neural networks with weights θ and $A\theta$ can have different values of sharpness while maintaining the same generalization gap, *i.e.* the sharpness is *scale-dependent*. As a solution, they introduce the concept of *adaptive* sharpness, defined as

$$\max_{||T_\theta^{-1}\epsilon||_p \leq \rho} \mathcal{L}_\mathcal{D}(\theta + \epsilon) - \mathcal{L}_\mathcal{D}(\theta) \tag{5}$$

where T_θ^{-1} is the normalization operator of θ such that $T_{A\theta}^{-1} A = T_\theta^{-1}$. Eq. 2 can be rewritten to define the Adaptive Sharpness-Aware Minimization (ASAM) problem as follows:

$$\min_{\theta \in \mathbb{R}^d} \max_{||T_\theta^{-1}\epsilon||_p \leq \rho} \mathcal{L}_\mathcal{D}(\theta + \epsilon) + \lambda ||\theta||_2^2 \tag{6}$$

For improving stability, T_θ is substituted by $T_\theta + \eta I_w$, where $\eta > 0$ is a hyper-parameter controlling the trade-off between stability and adaptivity, while w is the number of weight parameters of the model.

A.2 Stochastic Weight Averaging: Overview

SWA averages weights proposed by SGD, while using a learning rate schedule to explore regions of the weight space corresponding to high performing networks. At each step i of a cycle of length c, the learning rate is decreased from γ_1 to γ_2:

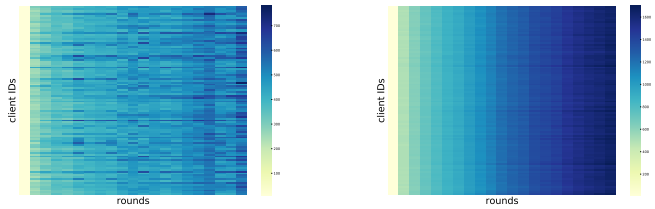
$$\gamma(i) = \big(1 - t(i)\big)\gamma_1 + t(i)\gamma_2, \quad t(i) = \frac{1}{c}\big(mod(i-1, c) + 1\big) \tag{7}$$

If $c = 1$ the learning rate is constant (γ_1), otherwise for $c > 1$ the learning schedule is cyclical. Starting from a pre-trained model $f_{\hat{\theta}}$, SWA captures all the updates θ at the end of each cycle and averages them as:

$$\theta_{\text{SWA}} \leftarrow \frac{\theta_{\text{SWA}} \cdot n_{\text{models}} + \theta}{n_{\text{models}} + 1} \tag{8}$$

obtaining the final model $f_{\theta_{\text{SWA}}}$, where n_{models} keeps track of the number of completed cycles.

In our method, SWA is applied on the server-side to make the learning process more robust. Adapting the scenario of [15] to FL, from 75% of the training onwards, the server keeps two models, f_θ and $f_{\theta_{\texttt{SWA}}}$ (f and $f_{\texttt{SWA}}$ to simplify the notation). f follows the standard FedAvg paradigm, while $f_{\texttt{SWA}}$ is updated every c rounds (Eq. 8). At each round, the cycling learning rate is computed (Eq. 7) and used for the clients' local training.



(a) (b)

Fig. 1: CIFAR100. L2-norm of global classifier output features as rounds pass, after receiving as input each client's local data. **(a)** with $\alpha = 0$, the model tends to focus on a different client's distribution, *i.e.* on a single class, at each round. **(b)** when $\alpha = 1000$, the model gives the same attention to each distribution.

A.3 Mixup and Cutout: Overview

Mixup and Cutout are recent methods for data augmentation, aiming to improve the learned models' generalization. We apply one of the two in the client-side training.

mixup [32] trains the neural network on convex combinations of images and their labels, exploiting the prior knowledge that linear interpolation of features leads to linear interpolations of their corresponding targets. Given two input images (x_i, x_j) and their corresponding one-hot label encodings (y_i, y_j) drawn from the k-th client's training data $\mathcal{D}_k$, virtual training examples are constructed as follows:

$$\begin{aligned} \bar{x} &= \lambda x_i + (1-\lambda) x_j \\ \bar{y} &= \lambda y_i + (1-\lambda) y_j \end{aligned} \tag{9}$$

with $\lambda \sim \text{Beta}(\alpha, \alpha)$ for $\alpha \in (0, \infty)$.

Cutout [5] regularizes learning by randomly masking out square regions of the input during training. At the implementation level, this corresponds to applying a fixed-size zero-mask to a random location of the image.

B Training in Heterogeneous Scenarios - Additional Material

In this section, we provide further analysis of the model's behavior in heterogeneous and homogeneous federated scenarios. As explained in Sec. 3.2, the model trained under a condition of statistical heterogeneity is subject to oscillations and loss in performance and generalization. Fluctuations in model predictions

Table 1: Datasets statistics

Dataset	Task	Train clients	Size imbalance	Train samples	Test samples
CIFAR10	Classification	100	✗	50,000	10,000
CIFAR100	Classification	100	✗	50,000	10,000
CIFAR100-PAM	Classification	500	✗	50,000	10,000
CIFAR10-C	DG	-	-	-	10,000
CIFAR100-C	DG	-	-	-	10,000
LANDMARKS-USER-160K	Classification	1,262	✓	164,172	19,526
CITYSCAPES (uniform)	SS	146	✓	2,975	500
CITYSCAPES (heterogeneous)	SS	144	✓		
IDDA (country)	SS+DG	90	✗	4,320	1,920
IDDA (rainy)	SS+DG	69	✗	3,312	2,928

can also be noted by looking at its output features, defined as $f_\theta(x)\ \forall x \in \mathcal{X}$. Fig. 1 shows the L2-norm of the output features computed using the current global model $f_\theta^t\ \forall t \in [T]$, given as input the local clients' data $\mathcal{D}_k\ \forall k \in [K]$, where a higher norm value corresponds to greater attention paid to that class by the network. The uniformity of the features obtained in the homogeneous setting contrasts with the chaotic distribution of the ones resulting when $\alpha = 0$, which significantly vary over time without following a constant trend.

C Experiments Details

Here we provide a detailed description of the datasets and models used in the paper, together with information regarding the chosen hyper-parameters and their fine-tuning intervals. All results presented in both the main text and the Appendix are averaged over the last 100 rounds for increased robustness and reliability. Unless otherwise specified, the framework is PyTorch [22] and experiments were run on one NVIDIA GeForce GTX 1070.

C.1 Datasets and Models

Table 1 summarizes the tasks and the statistics of the number of clients and examples for each dataset.

CIFAR10 and CIFAR100 We replicate the federated version of the CIFAR datasets proposed by [12]. Each dataset is split among 100 clients, receiving 500 images each according to the latent Dirichlet distribution (LDA) applied to the labels. The client's examples are selected following a multinomial distribution drawn from a symmetric Dirichlet distribution with parameter α. The higher the value of α the larger the number of classes locally seen , *i.e.* the more similar and homogeneous the clients' distributions are. We test $\alpha \in \{0, 0.05, 100\}$ on CIFAR10 and $\alpha \in \{0, 0.5, 1000\}$ on CIFAR100. The task is image classification on 10 (CIFAR10) and 100 (CIFAR100) classes.

Model: We train a Convolutional Neural Network (CNN) similar to LeNet5 [17] on both datasets, following the setting of [13]. The network has two 64-channels convolutional layers with kernel of size 5×5, each followed by a 2×2 max-pooling layer, ended by two fully connected layers with 384 and 192 channels respectively and a linear classifier.

Data pre-processing: The 32×32 input images are pre-processed following the standard pipeline: the training images are randomly cropped applying padding 4 with final size 32×32, randomly horizontally flipped with probability 0.5 and finally the pixel values are normalized with the dataset's mean and standard deviation; normalization is applied to test images as well.

CIFAR100-PAM We further extend our experiments to a more complex version of CIFAR100, *i.e.* CIFAR100-PAM proposed by [23], reflecting the "coarse" and "fine" label structure of the dataset for a more realistic partition. The dataset is split among 500 clients - with 100 images each - following the Pachinko Allocation Method (PAM) [19], on the result of which LDA is applied.

Model: We train a modified ResNet18, replacing Batch Normalization [14] layers with group normalization (GN) ones [29], as suggested by [11]. We use two groups for each GN layer. Experiments have been run using `FedJAX` [24] on a cluster with NVIDIA V100 GPUs.

Data pre-processing: CIFAR100-PAM images are pre-processed as the CIFAR LDA versions described above.

CIFAR10-C and CIFAR100-C are the corrupted versions of the CIFAR datasets. They are part of the benchmark proposed by [10], used for testing the image classifiers' robustness. The $10k$ images-test set is modified according to a given *corruption* and a corresponding level of *severity*. There are 19 possible corruptions (brightness, contrast, elastic blur, elastic transform, fog, frost, Gaussian blur, Gaussian noise, glass blur, impulse noise, JPEG compression, motion blur, pixelate, saturate, short noise, snow, spatter, speckle noise, zoom blur), while the severity ranges from 1 (low) to 5 (high).

Model: The same model described for CIFAR10 and CIFAR100 is used here. To test the generalization ability of our method, we test the model trained with CIFAR10/100 on the corresponding corrupted dataset.

Landmarks-User-160k Introduced by [13], the Landmarks-User-160k dataset comprises 164,172 training images belonging to 2,028 landmarks. The dataset is created according to the authorship information from the large-scale dataset Google Landmarks v2 (GLv2) [28]. Each author owns at least 30 pictures depicting 5 or more landmarks, while each location is depicted by at least 30 images and was visited by no less than 10 users. The authors in the test set do not overlap with the ones appearing in the training split.

Model: We follow a setting similar to the one proposed by [13] and use a MobileNetV2 [25] network pre-trained on ImageNet [4] with with GroupNorm layers in place of BatchNorm. Since no details on the model are available, we set the network feature multiplier $\alpha = 1$ and use 8 groups for the GN layers. We did not apply a bottleneck layer before the classifier as specified in [13]. To reduce training time, we use `Flax` [9] for both pre-training and centralized baselines, and `FedJAX` [24] for the implementation of the federated algorithms. Both libraries are based on `JAX` [2] and allow for efficient data parallelization. Implementation of the MobileNetV2 backbone used for all the experiments is available here[1]. All large-scale classification experiments have been performed using an NVIDIA DGX A100 40GB.

The model trained on ImageNet reaches $\approx 68\%$ top-1 accuracy on the validation set. In our experience, GroupNorm tends to perform slightly worse than BatchNorm when trained on ImageNet. However, since we did not extensively tune the hyper-parameters, getting better final performance is possible. For the ImageNet training, we used 8 GPUs with a total batch size of 2048 images.

Data pre-processing: We applied the same data augmentation for training the model on ImageNet and fine-tuning on GLv2: we crop and resize the input images to 224×224 with random scale and aspect ratio as described in [27]. The data augmentation pipeline used for the experiments can be found here[2]. We also adapted the GLv2 TensorFlow Federated data pipeline[3] to be compatible with `FedJAX`.

Cityscapes [3] is a popular dataset for Semantic Segmentation and contains 2,975 real photos taken in the streets of 50 different cities under good weather conditions. Annotations are provided for 19 semantic classes. We refer to the federated splits proposed in the FedDrive benchmark [6]. The *uniform* version of the dataset randomly assigns each image to one of the 146 users. In order to account for the distribution heterogeneity appearing in real-world scenarios, an ulterior version is proposed, referred to as *heterogeneous*: every client only accesses images from one of the 18 training cities. In both cases, the test set contains pictures of unseen cities.

Model: As proposed by the authors of FedDrive, we employ the lightweight network BiSeNetv2 [31] for training, accounting for possible lower computational capabilities of the edge devices.

Data pre-processing: The images are randomly scaled in the range (0.5, 1.5) and cropped to a 512×1024 shape.

[1] https://github.com/rwightman/efficientnet-jax/tree/a65811fbf63cb90b9ad0724792040ce93b749303

[2] https://github.com/google/flax/blob/571018d16b42ce0a0387515e96ba07130cbf79b9/examples/imagenet/input_pipeline.py#L90-L108

[3] https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets/gldv2/load_data

Table 2: Best performing training parameters

Dataset	Client learning rate	Batch size	Weight decay	Epochs	Client momentum	Rounds	Clients per round
CIFAR10	0.01	64	$4 \cdot 10^{-4}$	1	0	$10k$	$\{5, 10, 20\}$
CIFAR100	0.01	64	$4 \cdot 10^{-4}$	1	0	$20k$	$\{5, 10, 20\}$
CIFAR100-PAM	0.01	20	$4 \cdot 10^{-4}$	1-2	0.9	$10k$	$\{10, 20\}$
LANDMARKS-USER-160K	0.1	64	$4 \cdot 10^{-5}$	5	0	$5k$	10
CITYSCAPES (unif.)	0.05	8	$5 \cdot 10^{-4}$	2	0.9	$1.5k$	5
CITYSCAPES (het.)	0.05	8	$5 \cdot 10^{-4}$	2	0.9	$1.5k$	5
IDDA (country)	0.1	8	0	2	0.9	$1.5k$	5
IDDA (rainy)	0.1	8	0	2	0.9	$1.5k$	5

IDDA [1] is a synthetic dataset for semantic segmentation, specific for the field of autonomous driving. In addition to the annotations for 16 semantic classes, the driving conditions are further characterized by three axes: a city among the 7 available, ranging from Urban to Rural environments; one of 5 viewpoints, simulating different vehicles; an atmospheric condition among 3 possible choices (Noon, Sunset, Rainy), for a total of 105 *domains*. As done for Cityscapes, we refer to FedDrive [6] for the federated splits. In the *uniform* distribution of IDDA, each client has access to 48 images randomly drawn from the whole dataset. The *heterogeneous* version is built so that every user only sees a single domain. Two distinct testing scenarios are proposed to assess the generalization abilities of the learned model: one with images belonging to domains likely already seen at training time ("seen" in Table 8 of the main text) and another one containing a never-seen one ("unseen"). The unseen domain either contains images taken in the countryside ("country") to analyze the *semantic* shift or in rainy conditions ("rainy") for studying the shift in *appearance*.

Model: As done for Cityscapes, BiSeNetv2 is the model of choice.

Data pre-processing: The images are randomly scaled in the range (0.5, 2.0) and cropped to a 512×928 shape.

C.2 Hyper-parameters Tuning

We consider a different hyper-parameters setup for each dataset. The final choices of training hyper-parameters are summarized in Table 2. Table 3 and 4 respectively show the values used for `SAM`/`ASAM` and `SWA`.

CIFAR10 and CIFAR100 For both datasets, the training hyper-parameters follow the choice of [13]. The client learning rate is tuned between the values $\{0.01, 0.1\}$ and set to 0.01, the batch size is 64, $E \in \{1, 2\}$ is tested for the number of local epochs and the former is chosen. As for the weight decay the value $4 \cdot 10^{-4}$ leads to better performances than 0. The local optimizer is SGD with no momentum. No learning rate scheduler is used for simplicity. We optimize the cross-entropy loss. As for the server-side, we compare the behavior of different optimizers (*i.e.* SGD, Adam, AdaGrad) with learning rates in $\{0.001, 0.01, 0.1, 1\}$

Table 3: `FedSAM` and `FedASAM` hyper-parameters

Dataset	Distribution	`SAM`	`ASAM`	
		ρ	ρ	η
CIFAR10	$\alpha = 0$	0.1	0.7	0.2
	$\alpha = 0.05$	0.1	0.7	0.2
	$\alpha = 100$	0.02	0.05	0.2
CIFAR100	$\alpha = 0$	0.02	0.5	0.2
	$\alpha = 0.5$	0.05	0.5	0.2
	$\alpha = 1000$	0.05	0.5	0.2
CIFAR100-PAM	$\alpha = 0.1$	0.05	0.5	0/0.2
LANDMARKS-USER-160K	-	0.05	0.5	0/0.2
CITYSCAPES	het/unif	0.01	0.1	0.2
IDDA	het/unif	0.01	0.5	0.2

(results in Appendix E.1), following the setup of [23], and find out that `FedAvg`, *i.e.* SGD with learning rate 1, is the best choice. When testing `FedAvgM`, the server-side momentum $\beta = 0.9$. As for the other SOTAs, we choose $\mu = 0.1$ in `FedProx` and $\alpha = 0.01$ in `FedDyn` from $\{0.001, 0.01, 0.1\}$; in `AdaBest`, we tune $\beta \in \{0.8, 0.9\}$ and $\mu \in \{0.01, 0.02\}$ and pick $(0.9, 0.02)$ for CIFAR10 and $(0.8, 0.02)$ for CIFAR100. The training proceeds for $10k$ rounds on CIFAR10 and $20k$ rounds on CIFAR100.

Mixup/Cutout: Following the setup of [32], we fix $\alpha_{\text{mixup}} = 1$, resulting in λ uniformly distributed between 0 and 1. As for Cutout instead, we select a cutout size of 16×16 pixels for CIFAR10 and 8×8 for CIFAR100, as done by [5].

SAM/ASAM: The parameter ρ of `SAM` is searched in $\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$. As for `ASAM`, the value of ρ is tuned in $\{0.05, 0.1, 0.2, 0.5, 0.7, 1.0, 2.0\}$ and $\eta \in \{0.0, 0.01, 0.1, 0.2\}$. The choices made for each dataset and α are shown in Table 3. There is no distinction of values as clients vary per round.

SWA: We test `SWA`'s starting round in $\{5\%, 25\%, 50\%, 75\%\}$ of the rounds budget and as expected [15] the best contribution is given if applied from 75% of the training onwards (see Appendix E). We set the value of the learning rate γ_1 to 0.01 and test $\gamma_2 \in \{10^{-5}, 10^{-4}, 10^{-3}\}$, selecting $\gamma_2 = 10^{-4}$. The cycle length c is tested in $\{5, 10, 20\}$ and set to 10 for CIFAR10 and 20 for CIFAR100. Table 4 summarizes the choices.

CIFAR100-PAM The hyper-parameters follow the same choice of [23] (see Table 2). We report accuracy at $5K$ and $10K$ communication rounds.

Mixup/Cutout: Same as CIFAR100.

SAM/ASAM: We search hyperpameters in the same values as CIFAR100. For ρ we found 0.05 and 0.5 to be the best values respectively for `SAM` and `ASAM` in all configurations. For `ASAM` we found that $\eta = 0.2$ is working fine when cutout or no augmentations are applied, while $\eta = 0$ works best in the case of Mixup.

Table 4: `SWA` hyper-parameters

Dataset	c	γ_1	γ_2	Start round
CIFAR10	10	10^{-2}	10^{-4}	7500
CIFAR100	20	10^{-2}	10^{-4}	15000
CIFAR100-PAM	5	10^{-2}	10^{-4}	15000
LANDMARKS-USER-160K	5	10^{-1}	10^{-3}	3750/5000
CITYSCAPES	5	$5 \cdot 10^{-2}$	$5 \cdot 10^{-4}$	1125
IDDA	5	10^{-1}	10^{-3}	1125

SWA: Same as CIFAR100.

Landmarks-User-160k We start from the hyper-parameters proposed by [13]. In contrast with the original paper, we found that `FedAvgM` with momentum $\beta = 0.9$ is unstable with 10 participating clients and requires reducing the server learning rate to 0.1 to train the model. Better performance and faster convergence can be obtained with 50 clients per round and $\beta = 0.9$. However, we use 10 clients per round and `FedAvg` as the baseline because of our limited resources and to maintain consistency with other experiments. All hyper-parameters are described in Table 2.

SAM/ASAM: The parameter ρ of `SAM` is searched in $\{0.01, 0.05, 0.1\}$. As for `ASAM`, the value of ρ is tuned in $\{0.1, 0.3, 0.5\}$ and $\eta \in \{0.0, 0.1, 0.2\}$.

SWA: We tested both `SWA` starting at the 75% and 100% of training, *i.e.* the 3750-*th* and 5000-*th* rounds. We tested different combinations of cycle lengths $c \in \{5, 10, 20\}$ and learning rate $\gamma_2 \in \{10^{-2}, 10^{-3}, 10^{-4}\}$. The best performing learning rates (γ_1, γ_2) are respectively $(10^{-1}, 10^{-3})$ and the cycle length is 5.

Cityscapes and IDDA For both Cityscapes and IDDA, we maintain the choice of hyper-parameters of [6]. The clients' initial learning rate is 0.05 on Cityscapes and 0.1 on IDDA, the weight decay is $5 \cdot 10^{-4}$ on Cityscapes, while it is not used on IDDA, 2 local epochs, the client optimizer is SGD with momentum 0.9. Differently from [6], we do not use mixed precision, thus the batch size is reduced from 16 to 8. A polynomial learning rate scheduler is applied locally, following [31]. The optimization is based on the Online Hard-Negative Mining [26], which selects the 25% of the pixels having the highest cross-entropy loss. The training is spanned across $1.5k$ rounds.

SAM/ASAM: The parameter ρ of `SAM` is searched in $\{0.01, 0.05, 0.1\}$. As for `ASAM`, the value of ρ is tuned in the set $\{0.05, 0.1, 0.5\}$ and $\eta \in \{0.0, 0.1, 0.2\}$.

SWA: Following the setup established for the CIFAR datasets, `SWA` starts at the 75% of training, *i.e.* the 1125*th* round. The learning rates (γ_1, γ_2) are respectively $(10^{-1}, 10^{-3})$ for IDDA and $(5 \cdot 10^{-2}, 5 \cdot 10^{-4})$ for Cityscapes. The cycle length is 5 for both datasets.

C.3 Plotting the Loss Landscapes

In the main text, we introduced both 2-D (Fig. 2 of the main text) and 3-D plots of the loss landscapes (Fig. 1 of the main text). Implementation details follow.

2D Loss Landscape Following the indications of [8,20]:

1. We choose three weight vectors $\theta_1, \theta_2, \theta_3$ and use them to obtain two basis vectors $\vec{u} = (\theta_2 - \theta_1)$ and $\vec{v} = (\theta_3 - \theta_1) - \frac{\langle \theta_3 - \theta_1, \theta_2 - \theta_1 \rangle}{||\theta_2 - \theta_1||^2} \cdot (\theta_2 - \theta_1)$.
2. Then, the normalized vectors $\hat{u} = {}^{u}/_{||u||}$ and $\hat{v} = {}^{v}/_{||v||}$ form an orthonormal basis in the plain containing $\theta_1, \theta_2, \theta_3$.
3. We now define a Cartesian grid of $N \times N$ points in the basis $\hat{u}, \hat{v}$. In our case, $N = 21$.
4. For each point of the grid, the corresponding weights are computed and the loss is consequently evaluated with the resulting network. For each point P of the grid having coordinates (x, y), the corresponding weights are computed as $P = \theta_1 + x \cdot \hat{u} + y \cdot \hat{v}$. As a consequence, θ_1 is the reference and can be found in the origin $(0, 0)$.

We adapted the code of [8][4] to our scenario.

3D Loss Landscape The plots in Fig. 1 in the main text are generated using the code of [18][5], modified to fit our datasets and models. Given a network architecture and its pre-trained parameters, the loss surface is computed along random directions near the optimal parameters.

C.4 Computing Hessian Eigenvalues

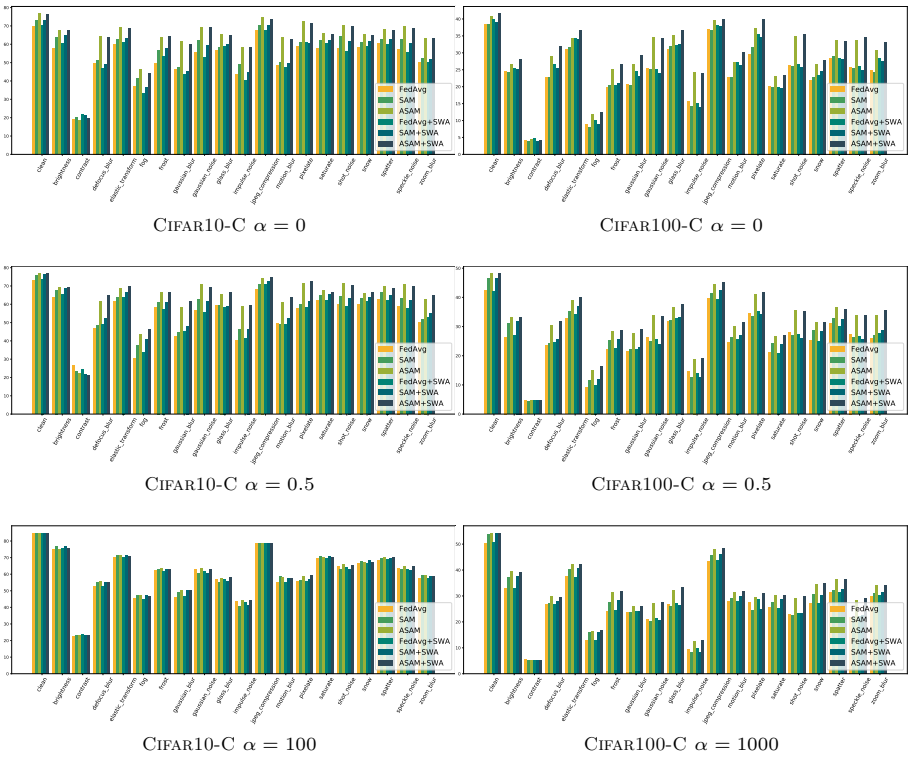
We refer to [21] for computing both the local and the top 50 Hessian eigenvalues (Figs. 4,5 in the main text) with Stochastic Power Iteration method [30] with maximum 20 iterations per run.

D Results on Corrupted CIFAR10 and CIFAR100

In Fig. 2, we compare the performance obtained by `FedAvg`, `FedSAM`, `FedASAM`, `FedAvg` + `SWA`, `FedSAM` + `SWA` and `FedASAM` + `SWA` on CIFAR10-C and CIFAR100-C as α varies. All results tell us that `ASAM` (alone or combined with `SWA`) is the algorithm with the best generalization capabilities, as already seen in Sec. 5.2 of the main text.

E Ablation Studies

In this Section, we present our ablation studies on server-side optimizers, `SAM`, `ASAM` and `SWA`, moved from the main text due to space constraints.



CIFAR10-C $\alpha = 0$ CIFAR100-C $\alpha = 0$

CIFAR10-C $\alpha = 0.5$ CIFAR100-C $\alpha = 0.5$

CIFAR10-C $\alpha = 100$ CIFAR100-C $\alpha = 1000$

Fig. 2: Domain generalization in FL. Results with 20 clients, severity level 5 on CIFAR10-C and CIFAR100-C.

E.1 Ablation Study on Server-Side Optimizers

To choose the best server-side optimizer, we test SGD, Adam and AdaGrad on the heterogeneous ($\alpha = 0$) and homogeneous ($\alpha = 1k$) versions of CIFAR100 with 5 clients per round. Following [23], we set $\beta_1 = \beta_2 = 0$ for AdaGrad and $\beta_1 = 0.9, \beta_2 = 0.99$ for Adam. As Table 5 shows, SGD with learning rate 1, *i.e.* `FedAvg`, is certainly the best choice to have acceptable performances both in the homogeneous scenario and above all in the heterogeneous one.

E.2 Ablation Study on `SAM` and `ASAM`

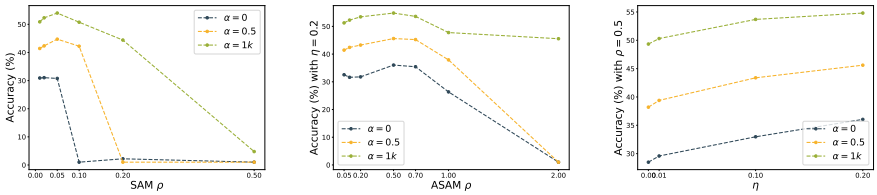
We present here an analysis on the sensitivity of the model to the hyper-parameters ρ and η in `ASAM` and ρ in `SAM` (Fig. 3), having as a reference the setting with 5% clients participation on CIFAR100. Regardless of the distribution, we can see that high values of `SAM`'s ρ lead to a fast decline in performance (Fig. 3a), meaning that the algorithm handles smaller neighborhoods better. On the other hand,

[4] https://github.com/timgaripov/dnn-mode-connectivity
[5] https://github.com/tomgoldstein/loss-landscape

Table 5: Final accuracy (%) using different server-side optimizers with varying learning rate (LR) on CIFAR100 @ $20k$ rounds. 5% clients participation. In bold the best results on both $\alpha = 0$ and $\alpha = 1k$.

Optimizer	LR	$\alpha = 0$	$\alpha = 1k$
`SGD`	1	**30.25**	**49.92**
	0.1	14.09	40.43
	0.01	2.67	11.35
	0.001	1.20	1.12
`Adam`	1	1.00	51.73
	0.1	29.75	51.62
	0.01	13.72	40.12
	0.001	2.60	11.31
`AdaGrad`	1	1.00	1.00
	0.1	1.77	46.74
	0.01	26.25	51.44
	0.001	9.70	32.01

`ASAM` allows us to have more freedom and expand the size of the neighborhood up to the value of $\rho = 0.5$ (Fig. 3b), index of the greater robustness of the method. In Fig. 3c, we notice that the performances improve linearly as η increases, where η is a hyper-parameter balancing the trade-off between stability and adaptivity.



(a) (b) (c)

Fig. 3: Results on CIFAR100, 5% clients participation. **(a)** Sensitivity to `SAM`'s parameter ρ. **(b)-(c)** Sensitivity to `ASAM`'s parameters ρ (with fixed $\eta = 0.2$) and η (with fixed $\rho = 0.5$) as α varies.

E.3 Ablation Study on `SWA`

`SWA` adds two new concepts to the standard federated training: the average of stochastic weights collected along the trajectory of SGD (Eq. 8) and the cyclical learning rate (Eq. 7), which decreases from γ_1 to γ_2 according to the cycle length c, transmitted as additional information to the clients of each round. Our ablation studies aim to understand which of these two components has the greatest impact on the achieved stability and increased model performance. We compare the results obtained by `SWA` with $c > 1$ with those reached when the learning rate is kept constant, *i.e.* $c = 1$, and when the server-side average of the collected

Table 6: `SWA` ablation study: comparison between cyclical ($c > 1$) and constant learning rate ($c = 1$) and contribution given by averaging stochastic weights. Highlighted in bold the best result for each combination (Algorithm, α, participating clients).

Dataset	Algorithm	WeightsAvg	c	$\alpha = 0$			$\alpha = 0.5/0.05$			$\alpha = 1k/100$		
				$5cl$	$10cl$	$20cl$	$5cl$	$10cl$	$20cl$	$5cl$	$10cl$	$20cl$
CIFAR100	FedAvg	✓	20	**39.34**	39.74	39.85	**43.90**	**44.02**	42.09	50.98	50.87	50.92
	FedSAM	✓	20	**39.30**	**39.51**	39.24	**47.96**	**46.76**	**46.47**	53.90	53.67	**54.36**
	FedASAM	✓	20	42.01	**42.64**	41.62	**49.17**	**48.72**	48.27	53.86	54.79	54.10
	FedAvg	✓	1	38.86	**39.82**	**40.19**	43.86	43.93	**42.67**	**51.33**	**51.05**	**51.11**
	FedSAM	✓	1	38.58	39.20	**39.37**	47.29	46.34	46.40	53.88	**53.70**	**54.36**
	FedASAM	✓	1	**42.50**	42.40	**41.76**	48.67	48.50	47.95	54.16	**55.07**	54.19
	FedAvg	✗	20	30.68	34.86	37.42	40.34	42.40	41.89	50.06	50.21	50.81
	FedSAM	✗	20	31.51	35.87	37.81	44.08	45.80	46.43	53.76	53.46	54.28
	FedASAM	✗	20	36.85	39.76	41.03	46.34	48.06	**48.38**	54.21	55.06	54.22
	FedAvg	✗	1	30.25	36.74	38.59	40.43	41.27	42.17	49.92	50.25	50.66
	FedSAM	✗	1	31.04	36.93	38.56	44.73	44.84	46.05	**54.01**	53.39	53.97
	FedASAM	✗	1	36.04	39.76	40.81	45.61	46.58	47.78	**54.81**	54.97	**54.50**
CIFAR10	FedAvg	✓	10	69.71	69.54	70.19	73.48	72.80	**73.81**	84.35	84.32	84.47
	FedSAM	✓	10	74.97	73.73	73.06	76.61	75.84	76.22	84.23	84.37	84.63
	FedASAM	✓	10	76.44	**75.51**	76.36	76.12	76.16	76.86	84.88	84.80	**84.79**
	FedAvg	✓	1	**69.88**	**69.83**	**70.72**	**73.91**	**73.12**	73.07	**84.90**	84.47	**84.67**
	FedSAM	✓	1	**75.17**	**74.00**	**73.53**	**76.93**	**76.06**	**76.55**	**84.53**	84.54	84.77
	FedASAM	✓	1	**76.80**	75.48	**76.84**	**76.87**	**76.30**	**77.55**	**85.09**	**85.06**	84.73
	FedAvg	✗	10	61.41	63.96	67.39	67.17	69.88	72.19	84.18	84.15	84.45
	FedSAM	✗	10	70.66	71.14	73.04	73.93	74.96	76.20	84.23	84.40	84.69
	FedASAM	✗	10	75.07	74.87	76.37	75.37	76.17	77.14	84.68	84.72	84.71
	FedAvg	✗	1	65.00	65.54	68.52	69.24	72.50	73.07	84.46	**84.50**	84.59
	FedSAM	✗	1	70.16	71.09	72.90	73.52	74.81	76.04	84.58	**84.67**	**84.82**
	FedASAM	✗	1	73.66	74.10	76.09	75.61	76.22	76.98	84.77	84.72	84.75

weights is not applied while maintaining $c > 1$, *i.e.* changing only the clients' learning rate cyclically (Table 6). We point out that using $c = 1$ and not applying the average brings us back to the standard federated setting. We discover that the server-side average gives the major contribution, which helps in stabilizing learning, while the cycle length does not particularly affect the results. Since the best results in the most difficult scenarios (*i.e.* low value of both α and number of participating clients on CIFAR100) are reached when $c > 1$, we prefer the cyclical learning rate to the constant one in further experiments.

In addition, in Table 7 we report the differences in results when applying `SWA` from $\{5\%, 25\%, 50\%, 75\%\}$ of the training onwards on `FedAvg` with 5 clients per round, showing that a longer pre-training of the network leads to the greater effectiveness of this algorithm.

F Tables Omitted in the Main Text

F.1 Heterogeneous FL Benefits Even More from Flat Minima - Additional Material

Table 8 completes the analysis introduced in Sec. 5.1 regarding the gains obtained in the federated scenario w.r.t. the centralized one. Here we report the results for $\alpha \in \{0, 1k\}$. As noted for $\alpha = 0$ (Table 5 in the main text), data augmentations

Table 7: `SWA` ablation study: comparison between `SWA` starting rounds when using `FedAvg` with 5 clients per round

Dataset	c	Start round	Test Accuracy (%)		
			$\alpha = 0$	$\alpha = 0.5/0.05$	$\alpha = 1k/100$
CIFAR100	20	1000	24.53	34.52	49.38
		5000	30.66	39.71	**51.52**
		10000	36.21	42.55	51.01
		15000	**39.34**	**43.90**	50.98
CIFAR10	10	500	55.57	60.50	79.09
		2500	60.34	65.72	81.49
		5000	66.22	70.55	83.79
		7500	**69.71**	**73.48**	**84.35**

fail in the federated heterogeneous scenarios ($\alpha \in \{0, 0.5\}$), but reasonably work in the homogeneous ones.

Table 8: Comparison of improvements (%) in centralized and federated scenarios ($\alpha \in \{0.5, 1k\}$, 5 clients) on CIFAR100, computed w.r.t. the reference at the bottom

Algorithm	Accuracy			Absolute Improvement			Relative Improvement		
	Centr.	$\alpha = 0.5$	$\alpha = 1k$	Centr.	$\alpha = 0.5$	$\alpha = 1k$	Centr.	$\alpha = 0.5$	$\alpha = 1k$
`SAM`	55.22	44.73	54.01	+3.02	+4.30	+4.01	+5.79	+10.64	+8.03
`ASAM`	55.66	45.61	54.81	+3.46	+5.18	**+4.89**	+6.63	+12.81	**+9.80**
`SWA`	52.72	43.90	50.98	+0.52	+3.47	+1.06	+1.00	+8.58	+2.12
`SAM + SWA`	55.75	47.96	53.90	+0.55	+7.53	+3.98	+1.06	+18.63	+7.97
`ASAM + SWA`	55.96	49.17	53.86	+3.76	**+8.74**	+3.94	+7.20	**+21.62**	+7.89
`Mixup`	58.01	35.10	55.34	+5.81	-5.33	+5.42	+11.13	-13.18	+10.86
`Cutout`	55.30	37.72	53.48	+3.10	-2.71	+3.56	+5.94	-6.70	+7.13
`Centralized`: **52.20** - `FedAvg` $\alpha = 0.5$: **40.43**, $\alpha = 1k$: **49.92**									

F.2 Data Augmentations with CIFAR10

Here we show the results obtained when applying Mixup and Cutout to CIFAR10 as the value of α, clients participation and algorithm change (Table 9). As demonstrated for CIFAR100 (Sec. 5.1), data augmentations do not improve generalization in a federated context, but on the contrary they seem to inhibit learning, leading to sometimes even worse results than `FedAvg`.

G Figures Omitted in the Main Text

All plots are best seen in colors.

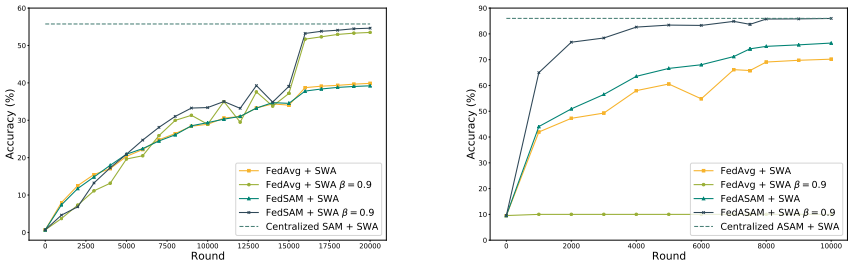
Convergence plots As shown in Sec. 5.1, once combined with `FedAvgM`- *i.e.* server-side momentum $\beta = 0.9$ - `SAM` and `ASAM` allow to reach convergence even in the most heterogeneous scenarios on both CIFAR10 and CIFAR100. Fig. 4 shows the convergence plots of those runs. In addition, Fig. 5 compares the behavior of `FedAvg`, `FedSAM`, `FedASAM` and their combination with `SWA` on the most difficult setting, *i.e.* $\alpha = 0$ and 5 clients per round on both CIFAR datasets, highlighting the stability and the positive gap in performance introduced by `SWA`.

Table 9: **FedAvg**, **SAM**, **ASAM** and **SWA** w/ strong data augmentations (**Mixup**, **Cutout**) on CIFAR10

	Algorithm	SWA	Aug	$\alpha = 0$			$\alpha = 0.5/0.05$			$\alpha = 1000/100$		
				$5cl$	$10cl$	$20cl$	$5cl$	$10cl$	$20cl$	$5cl$	$10cl$	$20cl$
CIFAR10	FedAvg	✗	None	65.00	65.54	68.52	69.24	72.50	73.07	84.46	84.50	84.59
	FedSAM	✗	None	70.16	71.09	72.90	73.52	74.81	76.04	84.58	84.67	**84.82**
	FedASAM	✗	None	73.66	74.10	76.09	75.61	76.22	76.98	84.77	84.72	84.75
	FedAvg	✓	None	69.71	69.54	70.19	73.48	72.80	73.81	84.35	84.32	84.47
	FedSAM	✓	None	74.97	73.73	73.06	76.61	75.84	76.22	84.23	84.37	84.63
	FedASAM	✓	None	**76.44**	**75.51**	**76.36**	**76.12**	**76.16**	**76.86**	**84.88**	**84.80**	**84.79**
	FedAvg	✗	Mixup	62.26	63.61	65.54	65.63	68.44	68.21	**82.38**	**84.46**	**83.58**
	FedSAM	✗	Mixup	67.35	69.32	69.78	70.34	72.98	72.54	81.88	82.24	82.25
	FedASAM	✗	Mixup	70.61	71.31	71.62	72.19	**72.84**	**72.72**	82.36	82.75	83.08
	FedAvg	✓	Mixup	66.31	66.89	66.26	69.79	69.12	68.80	82.27	82.88	82.67
	FedSAM	✓	Mixup	**72.42**	70.65	69.75	**73.36**	72.29	72.44	81.04	81.18	81.15
	FedASAM	✓	Mixup	72.37	**72.40**	**71.89**	72.54	72.36	72.32	81.86	81.70	81.92
	FedAvg	✗	Cutout	61.12	64.47	64.20	66.45	69.09	68.99	**83.77**	83.91	**84.31**
	FedSAM	✗	Cutout	63.69	66.30	67.25	67.66	71.39	70.67	83.03	83.84	83.49
	FedASAM	✗	Cutout	68.50	69.26	69.75	69.23	**71.91**	**71.28**	83.73	**84.10**	84.00
	FedAvg	✓	Cutout	65.54	65.60	65.79	69.94	69.55	69.63	83.35	83.39	83.64
	FedSAM	✓	Cutout	69.40	68.45	67.36	71.36	71.56	70.99	82.61	82.75	82.52
	FedASAM	✓	Cutout	**71.30**	**71.12**	**70.91**	**72.79**	71.76	71.09	83.06	83.31	83.11

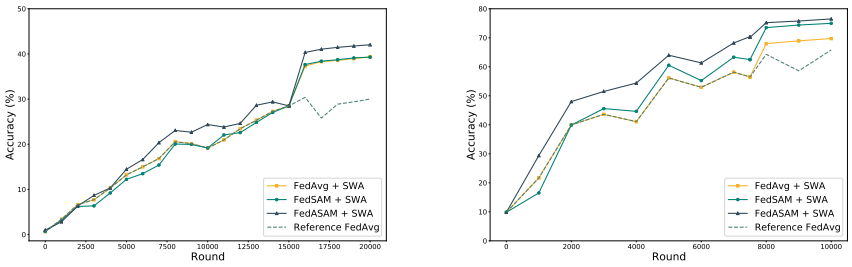
Loss Surfaces Fig. 6 shows the convergence points of three local models trained with $\alpha = 0.5$ on the corresponding test error surface, while Fig. 7 displays the train loss surfaces with $\alpha \in \{0, 0.5, 1000\}$. In addition, in Fig. 8 we compare the convergence points of **FedAvg**, **FedSAM** and **FedASAM** in the heterogeneous scenarios of CIFAR100, *i.e.* $\alpha \in \{0, 0.5\}$, proving that **ASAM** reaches the best local minimum.

Hessian Eigenvalues The top 50 eigenvalues of the global model trained with $\alpha = 0.5$ are showed in Fig. 9. Fig. 10 shows the complete comparison of the local Hessian eigenvalues partially shown in Sec. 3.2, introducing the values of λ^k_{max} $\forall k \in [K]$ resulting with **SAM** and $\alpha = 0.5$.



CIFAR100 CIFAR10

Fig. 4: Convergence plots with $\alpha = 0$, 20 clients. When combining `FedAvgM` or `FedSAM` (CIFAR100)/`FedASAM` (CIFAR10) with `SWA`, convergence is reached even in the most heterogeneous scenarios. `FedAvgM + SWA` applied to CIFAR10 fails to learn, while adding momentum to `FedASAM` significantly speeds up training.



CIFAR100 CIFAR10

Fig. 5: Convergence plots with $\alpha = 0$, 5 clients, highlighting the positive gap in performance and the stability introduced by `SWA` (both if applied on `FedAvg` but especially on `FedASAM`) in the most difficult setting.

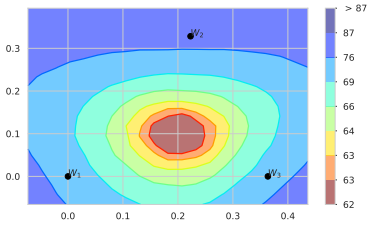
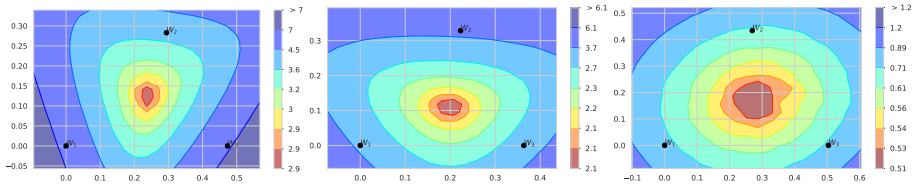
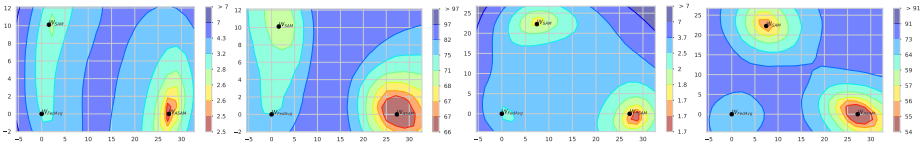


Fig. 6: Test error surface computed on CIFAR100 using three distinct local models trained with $\alpha = 0.5$ for $20k$ rounds.



(a) (b) (c)

Fig. 7: Train cross-entropy loss surfaces computed with three local models after $20k$ training rounds on CIFAR100. **(a)** $\alpha = 0$ **(b)** $\alpha = 0.5$ **(c)** $\alpha = 1000$.



(a) (b) (c) (d)

Fig. 8: Loss surfaces comparing the convergence points of `FedAvg`, `FedSAM` and `FedASAM` after $20k$ training rounds on CIFAR100. The minima reached by `SAM` and `ASAM` are found within low-loss neighborhoods. **(a)** Train loss surface $\alpha = 0$. **(b)** Test error surface $\alpha = 0$. **(c)** Train loss surface $\alpha = 0.5$. **(d)** Test error surface $\alpha = 0.5$.

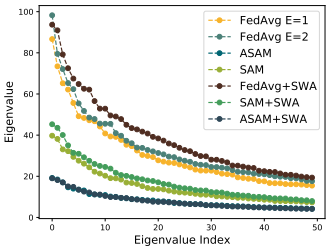
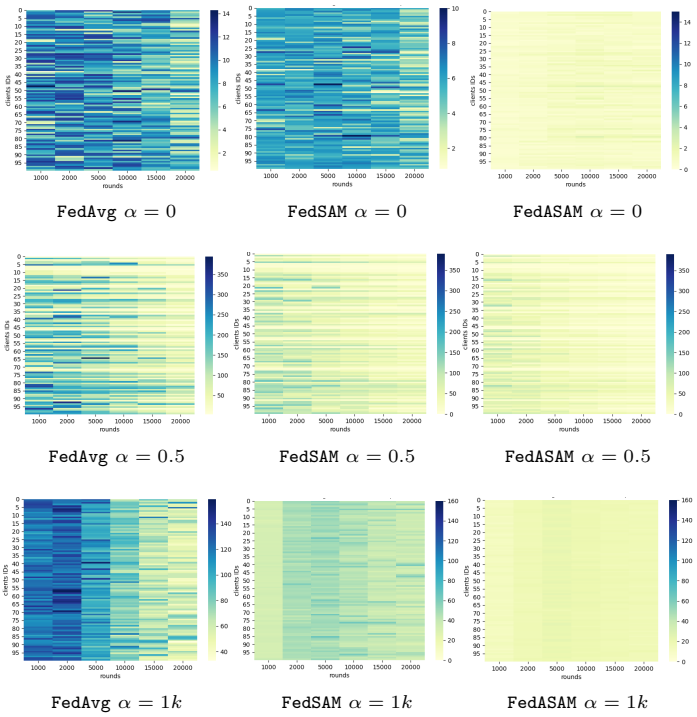


Fig. 9: Top 50 eigenvalues of the global model with $\alpha = 0.5$ on CIFAR100.



FedAvg $\alpha = 0$ FedSAM $\alpha = 0$ FedASAM $\alpha = 0$

FedAvg $\alpha = 0.5$ FedSAM $\alpha = 0.5$ FedASAM $\alpha = 0.5$

FedAvg $\alpha = 1k$ FedSAM $\alpha = 1k$ FedASAM $\alpha = 1k$

Fig. 10: Maximum Hessian eigenvalue computed for each client as rounds pass.

References

1. Alberti, E., Tavera, A., Masone, C., Caputo, B.: Idda: a large-scale multi-domain dataset for autonomous driving. IEEE Robotics and Automation Letters **5**(4), 5526–5533 (2020) 7
2. Bradbury, J., Frostig, R., Hawkins, P., Johnson, M.J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., Zhang, Q.: JAX: composable transformations of Python+NumPy programs (2018), http://github.com/google/jax 6
3. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3213–3223 (2016) 6
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009) 6
5. DeVries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552 (2017) 3, 8
6. Fantauzzo, L., Fani', E., Caldarola, D., Tavera, A., Cermelli, F., Ciccone, M., Caputo, B.: Feddrive: Generalizing federated learning to semantic segmentation in autonomous driving. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2022) 6, 7, 9
7. Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B.: Sharpness-aware minimization for efficiently improving generalization. International Conference on Learning Representations (2021) 1
8. Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D.P., Wilson, A.G.: Loss surfaces, mode connectivity, and fast ensembling of dnns. Advances in neural information processing systems **31** (2018) 10
9. Heek, J., Levskaya, A., Oliver, A., Ritter, M., Rondepierre, B., Steiner, A., van Zee, M.: Flax: A neural network library and ecosystem for JAX (2020), http://github.com/google/flax 6
10. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. International Conference on Learning Representations (2019) 5
11. Hsieh, K., Phanishayee, A., Mutlu, O., Gibbons, P.: The non-iid data quagmire of decentralized machine learning. In: International Conference on Machine Learning. pp. 4387–4398. PMLR (2020) 5
12. Hsu, T.M.H., Qi, H., Brown, M.: Measuring the effects of non-identical data distribution for federated visual classification. NeurIPS Workshop (2019) 4
13. Hsu, T.M.H., Qi, H., Brown, M.: Federated visual classification with real-world data distribution. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16. pp. 76–92. Springer (2020) 5, 6, 7, 9
14. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. pp. 448–456. PMLR (2015) 5
15. Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., Wilson, A.G.: Averaging weights leads to wider optima and better generalization. Uncertainty in Artificial Intelligence (UAI) (2018) 1, 2, 8

16. Kwon, J., Kim, J., Park, H., Choi, I.K.: Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. International Conference on Machine Learning (2021) 1, 2
17. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998) 5
18. Li, H., Xu, Z., Taylor, G., Studer, C., Goldstein, T.: Visualizing the loss landscape of neural nets. In: Neural Information Processing Systems (2018) 10
19. Li, W., McCallum, A.: Pachinko allocation: Dag-structured mixture models of topic correlations. In: Proceedings of the 23rd international conference on Machine learning. pp. 577–584 (2006) 5
20. Mirzadeh, S.I., Farajtabar, M., Gorur, D., Pascanu, R., Ghasemzadeh, H.: Linear mode connectivity in multitask and continual learning. NeurIPS (2018) 10
21. Noah, G., Zhewei, Y., Amir, G., Michael, M., Joseph, G.: pytorch-hessian-eigenthings: efficient pytorch hessian eigendecomposition (Oct 2018), https://github.com/noahgolmant/pytorch-hessian-eigenthings 10
22. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019) 4
23. Reddi, S., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečnỳ, J., Kumar, S., McMahan, H.B.: Adaptive federated optimization. International Conference on Learning Representations (2021) 5, 8, 11
24. Ro, J.H., Suresh, A.T., Wu, K.: Fedjax: Federated learning simulation with jax. arXiv preprint arXiv:2108.02117 (2021) 5, 6
25. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4510–4520 (2018) 6
26. Shrivastava, A., Gupta, A., Girshick, R.: Training region-based object detectors with online hard example mining. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 761–769 (2016) 9
27. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015) 6
28. Weyand, T., Araujo, A., Cao, B., Sim, J.: Google landmarks dataset v2-a large-scale benchmark for instance-level recognition and retrieval. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2575–2584 (2020) 5
29. Wu, Y., He, K.: Group normalization. In: Proceedings of the European conference on computer vision (ECCV). pp. 3–19 (2018) 5
30. Xu, P., He, B., De Sa, C., Mitliagkas, I., Re, C.: Accelerated stochastic power iteration. In: International Conference on Artificial Intelligence and Statistics. pp. 58–67. PMLR (2018) 10
31. Yu, C., Gao, C., Wang, J., Yu, G., Shen, C., Sang, N.: Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. International Journal of Computer Vision **129**(11), 3051–3068 (2021) 6, 9
32. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. International Conference on Learning Representations (2018) 3, 8