Beyond Local Sharpness: Communication-Efficient Global Sharpness-aware Minimization for Federated Learning
Appendix

A. Algorithm

Alg. 1 summarizes FEDGLOSS, using as an example SGD or SAM as local optimizers, differently highlighted. The comment colors replicate those of Fig. 3.

Algorithm 1 FEDGLOSS with SAM or SGD as local optimizers

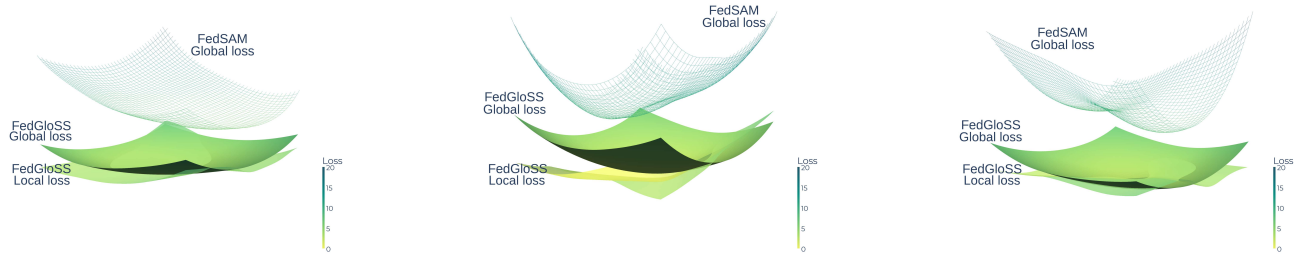
Input: Global model $\boldsymbol{w}$, clients $\mathcal{C}$, rounds T, local iterations E, clients' learning rate η, clients' SAM neighborhood size ρ_l, FEDGLOSS neighborhood size ρ, Lagrangian hyperparameter β.
Initialize: $\boldsymbol{w}^0$, $\sigma^0 = \sigma_k = 0$, $\Delta_{\boldsymbol{w}}^0 = 0$.
for each round $t \in [1, T]$ **do**
  $\tilde{\boldsymbol{\epsilon}}^t(\boldsymbol{w}) = \rho \frac{\Delta_{\boldsymbol{w}}^{t-1}}{\|\Delta_{\boldsymbol{w}}^{t-1}\|_2}$ ▷ Global perturbation with previous pseudo-grad
  $\tilde{\boldsymbol{w}}^t = \boldsymbol{w}^t + \tilde{\boldsymbol{\epsilon}}^t(\boldsymbol{w})$ ▷ Server-side approximated **FEDGLOSS ascent step**
  Randomly select a subset of clients $\mathcal{C}^t \subset \mathcal{C}$
  for each client $k \in \mathcal{C}^t$ in parallel **do**
    $\boldsymbol{w}_{k,0} = \tilde{\boldsymbol{w}}^t$ ▷ Initialize local model with *perturbed* global model $\tilde{\boldsymbol{w}}^t$
    Set iteration counter $i = 1$
    for each epoch $e \in [1, E]$ **do**
      for each batch $\mathcal{B} \in \mathcal{D}_k$ **do**
        $\boldsymbol{g}_{k,i} = \nabla f_{\mathcal{B}}(\boldsymbol{w}_{k,i-1})$ ▷ Local SGD gradient
        $\hat{\boldsymbol{\epsilon}}_{k,i} = \rho_l \frac{\boldsymbol{g}_{k,i}}{\|\boldsymbol{g}_{k,i}\|_2}$ ▷ SAM local perturbation
        $\boldsymbol{g}_{k,i} = \nabla f_{\mathcal{B}}(\boldsymbol{w}_{k,i-1} + \hat{\boldsymbol{\epsilon}}_{k,i})$ ▷ Local sharpness-aware gradient
        $\boldsymbol{w}_{k,i} \leftarrow \boldsymbol{w}_{k,i-1} - \eta[\boldsymbol{g}_{k,i} - \sigma_k + (\boldsymbol{w}_{k,i-1} - \boldsymbol{w}_{k,0})/\beta]$ ▷ Local update with ADMM
        $i \leftarrow i + 1$
      end for
    end for
    $\sigma_k \leftarrow \sigma_k - (\boldsymbol{w}_{k,E} - \tilde{\boldsymbol{w}}^t)/\beta$ ▷ Update local dual variable
    Send back to the server the local updated model $\boldsymbol{w}_k^t = \boldsymbol{w}_{k,E}$
  end for
  $\sigma^{t+1} = \sigma^t - \frac{1}{\beta|\mathcal{C}|} \sum_{k \in \mathcal{C}} (\boldsymbol{w}_k^t - \tilde{\boldsymbol{w}}^t)$ ▷ Update global dual variable
  $\tilde{\Delta}_{\boldsymbol{w}}^t = \sum_{k \in \mathcal{C}^t} \frac{N_k}{N} (\tilde{\boldsymbol{w}}^t - \boldsymbol{w}_k^t)$ ▷ **Global pseudo-gradient**
  $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \tilde{\Delta}_{\boldsymbol{w}}^t - \beta\sigma^{t+1}$ ▷ FEDGLOSS descent step w/ pseudo-grad using ADMM
end for

B. The Benefits on FEDGLOSS

B.1. Achieving Consistency on Local and Global Sharpness with FedGloSS

This section completes the analyses presented in Secs. 4 and 6.2, showing how FEDGLOSS guides towards consistent local and global flat loss landscapes. Fig. 10 extends Figs. 4a and 4b from the main paper with the 3 remaining clients (out of 5) selected in the last round, comparing the behavior of clients' models trained with FEDGLOSS from the local and global perspectives. These results highlight the effectiveness of FEDGLOSS in achieving local models with consistent local-global behavior. Indeed, these models end up in flat regions both in local and global loss landscapes. The comparison with FEDSAM (*net* surface) further demonstrates the effectiveness of using our global flatness-aware approach.

Fig. 11 instead extends Figs. 4c and 4d, offering a comparative analysis of FEDSMOO's clients' models w.r.t. FEDGLOSS (*net* landscape). The local solutions found by FEDGLOSS achieve flatter, better (*i.e.*, lower loss) and more consistent convergence points in the *global* loss landscape w.r.t. our main competitor FEDSMOO.

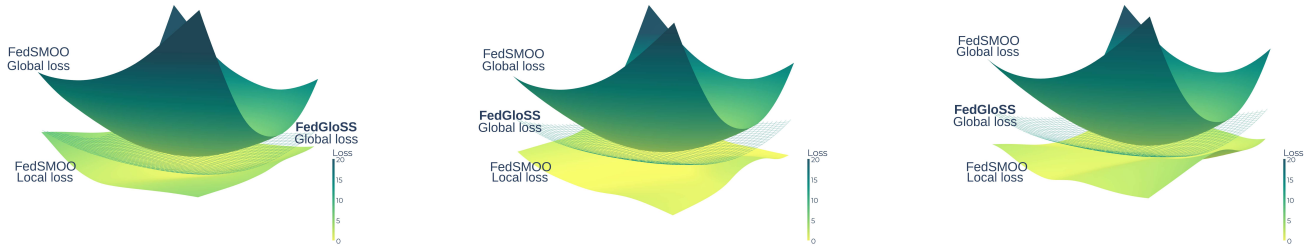


(a) Local model trained on class `bear` with FEDGLOSS

(b) Local model trained on class `skyscraper` with FEDGLOSS

(c) Local model trained on class `possum` with FEDGLOSS

Figure 10. **Global *vs.* local perspective on FEDGLOSS**. CIFAR100 $\alpha = 0$ with SAM as local optimizer @ $20k$ rounds on CNN. **(a) - (c):** Local models trained on one `class`, tested on the local ("*Local loss*") or global dataset ("*Global loss*"). Corresponding global perspective of local model trained with FEDSAM (*net*) added as reference.

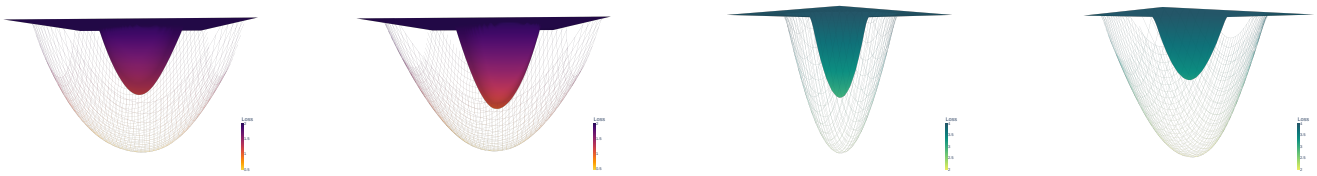


(a) Local model trained on class `bear` with FEDGLOSS

(b) Local model trained on class `skyscraper` with FEDGLOSS

(c) Local model trained on class `possum` with FEDGLOSS

Figure 11. **Global *vs.* local perspective on FEDSMOO**. CIFAR100 $\alpha = 0$ with SAM as local optimizer @ $20k$ rounds on CNN. **(a) - (c):** Local models trained on one `class`, tested on the local ("*Local loss*") or global dataset ("*Global loss*"). Corresponding global perspective of local model trained with FEDGLOSS (*net*) added as reference.

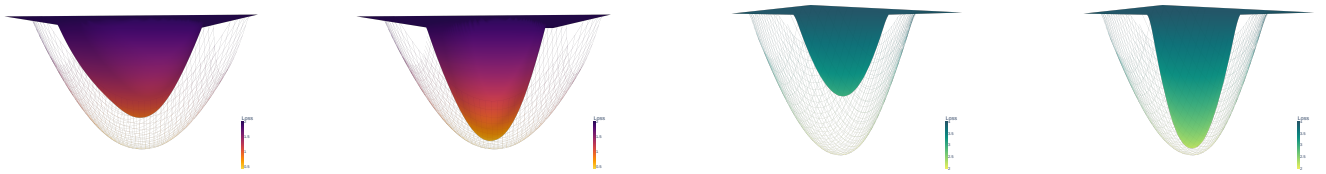


(a) CIFAR10 $\alpha = 0$ (b) CIFAR10 $\alpha = 0.05$ (c) CIFAR100 $\alpha = 0$ (d) CIFAR100 $\alpha = 0.5$

Figure 12. Visualization of the loss landscapes of the CNN trained with **FEDGLOSS** (*net*) and the de-facto standard optimization algorithm in FL **FEDAVG** (*solid*). Comparison with varying degrees of heterogeneity on CIFAR10 (left) and CIFAR100 (right). **FEDGLOSS consistently achieves flatter minima and lower loss values in the *global* loss landscape.**

B.2. Achieving Flatter Global Minima

FEDGLOSS *vs.* FEDAVG, FEDSAM, FEDSMOO. Fig. 12 compares the loss landscapes of global models trained with FEDGLOSS and FEDAVG, showing how the former consistently achieves flatter minima and lower loss values in the *global* loss landscapes. This confirms the behaviors already appreciated in Fig. 6. In addition, Fig. 13 shows the global loss surfaces of FEDGLOSS' solutions against models trained with FEDSAM and FEDSMOO. These plots extend Fig. 6 with the less heterogeneous scenarios $\alpha = 0.05$ for CIFAR10 and $\alpha = 0.5$ on CIFAR100. They confirm FEDGLOSS' effectiveness in reaching flatter and lower-loss solutions with respect to its main direct competitors.



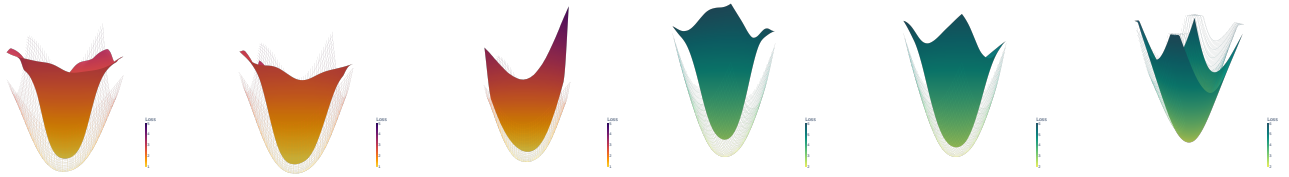
(a) CIFAR10 $\alpha = 0.05$ FEDGLOSS *vs.* FEDSAM

(b) CIFAR10 $\alpha = 0.05$ FEDGLOSS *vs.* FEDSMOO

(c) CIFAR100 $\alpha = 0.5$ FEDGLOSS *vs.* FEDSAM

(d) CIFAR100 $\alpha = 0.5$ FEDGLOSS *vs.* FEDSMOO

Figure 13. Visualization of the loss landscapes of the CNN trained with **FEDGLOSS** (*net*) **and FEDSAM or FEDSMOO** (*solid*). Comparison with $\alpha = 0.05$ CIFAR10 (left) and $\alpha = 0.5$ CIFAR100 (right) extending Fig. 6. **FEDGLOSS consistently achieves flatter minima and lower loss values in the *global* loss landscape.**



(a) CIFAR10 $\alpha = 0.05$ FEDGLOSS (*net*) *vs.* FEDAVG (*solid*)

(b) CIFAR10 $\alpha = 0.05$ FEDGLOSS (*net*) *vs.* FEDSAM (*solid*)

(c) CIFAR10 $\alpha = 0.05$ FEDGLOSS (*net*) *vs.* FEDSMOO (*solid*)

(d) CIFAR100 $\alpha = 0.5$ FEDGLOSS (*net*) *vs.* FEDAVG (*solid*)

(e) CIFAR100 $\alpha = 0.5$ FEDGLOSS (*net*) *vs.* FEDSAM (*solid*)

(f) CIFAR100 $\alpha = 0.5$ FEDGLOSS (*net*) *vs.* FEDSMOO (*solid*)

Figure 14. Visualization of loss landscapes of the ResNet18 trained with **FEDGLOSS** (*net*) **and FEDAVG, or FEDSAM or FEDSMOO** (*solid*). Comparison with CIFAR10 $\alpha = 0.05$ (*a-c*) and CIFAR100 $\alpha = 0.5$ (*d-f*). **FEDGLOSS achieves flatter and lower-loss regions in the global landscape**.

FEDGLOSS on ResNet18. Fig. 14 extends Fig. 6 from the main paper and shows the flatter loss landscapes reached by FEDGLOSS when using ResNet18 on CIFAR10 and CIFAR100.

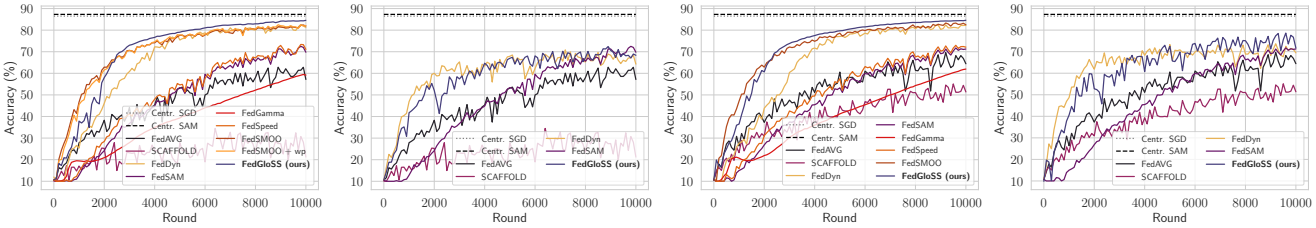
Hessian Eigenvalues. Tab. 9 reports the values of the maximum Hessian eigenvalues, as already shown in Fig. 7 in the main paper. First, as expected, we note that SAM-based methods achieve flatter minima w.r.t. the counterpart. Notably, our main competitor FEDSMOO presents higher sharpness than FEDSAM in the simpler CIFAR10, regardless of the data distribution. In addition, FEDGLOSS with local SAM achieves the lowest sharpness (*i.e.*, lowest λ_1) on *all* configurations, outperforming the state of the art and, specifically, *all* sharpness-aware methods.

Table 9. **FEDGLOSS *vs.* the state of the art**, distinguished by local optimizer - SGD (*top*) and SAM (*bottom*). Comparison in terms of communication cost and maximum Hessian eigenvalue λ_1. Best results in **bold**. Model: CNN.

	Method	Comm. Cost	CIFAR10 $\alpha = 0$ $\lambda_1(\downarrow)$	CIFAR10 $\alpha = 0.05$ $\lambda_1(\downarrow)$	CIFAR100 $\alpha = 0$ $\lambda_1(\downarrow)$	CIFAR100 $\alpha = 0.5$ $\lambda_1(\downarrow)$
Client SGD	FEDAVG	1×	66.23 ± 0.50	71.14 ± 4.07	$\mathbf{66.30 \pm 3.08}$	68.77 ± 0.96
	FEDPROX	1×	66.19 ± 0.52	71.41 ± 4.40	66.34 ± 3.75	$\mathbf{68.63 \pm 1.37}$
	FEDDYN	1×	63.94 ± 4.41	71.44 ± 8.73	-	-
	SCAFFOLD	2×	166.54 ± 6.93	180.51 ± 30.08	-	120.01 ± 0.76
	FEDGLOSS	1×	$\mathbf{58.26 \pm 3.49}$	$\mathbf{56.28 \pm 4.19}$	96.01 ± 9.00	107.35 ± 7.5
Client SAM	FEDSAM	1×	10.35 ± 0.07	9.43 ± 0.28	58.38 ± 2.93	57.54 ± 1.21
	FEDDYN	1×	10.04 ± 5.38	6.58 ± 0.20	-	-
	FEDSPEED	1×	10.92 ± 0.17	9.97 ± 0.12	58.23 ± 3.18	58.00 ± 1.86
	FEDGAMMA	2×	4.79 ± 0.20	4.55 ± 0.20	-	99.86 ± 6.74
	FEDSMOO	2×	15.37 ± 1.67	12.57 ± 0.56	28.43 ± 1.97	29.23 ± 0.17
	FEDGLOSS	1×	$\mathbf{2.03 \pm 0.05}$	$\mathbf{1.93 \pm 0.03}$	$\mathbf{17.18 \pm 0.97}$	$\mathbf{16.22 \pm 0.35}$

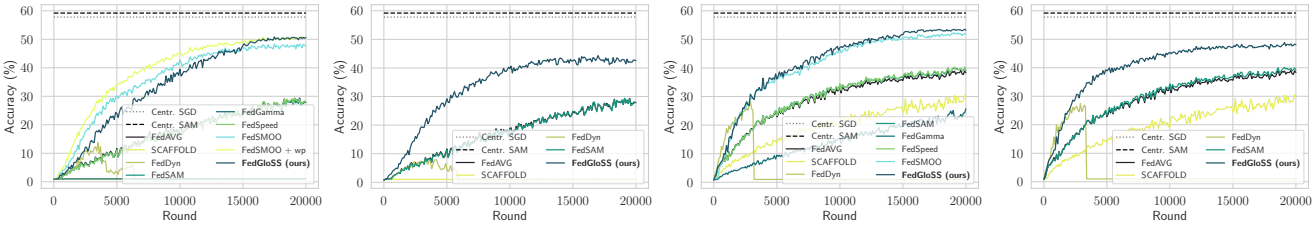
B.3. Increasing Convergence Speed

Figs. 15 and 16 show the accuracy trends of FEDGLOSS compared to state-of-the-art methods for heterogeneous FL on CIFAR10 and CIFAR100 respectively. For a clearer understanding, we distinguish between SAM-based and SGD-based



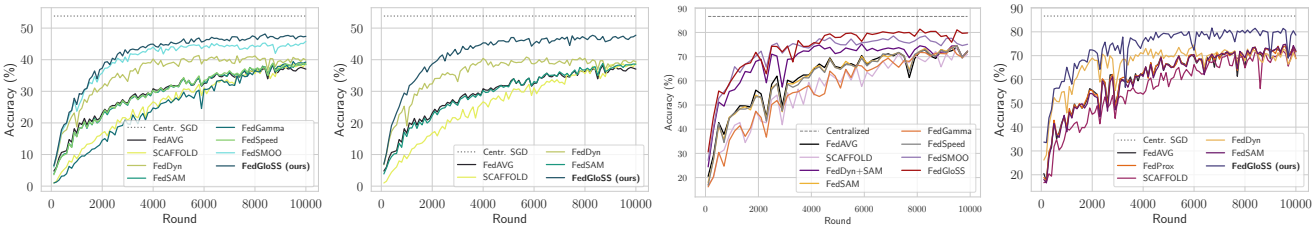
(a) $\alpha = 0$ SAM (b) $\alpha = 0$ SGD (c) $\alpha = 0.05$ SAM (d) $\alpha = 0.05$ SGD

Figure 15. CIFAR10 with varying degrees of heterogeneity ($\alpha \in \{0, 0.05\}$). Results of centralized runs (*dashed lines*) added as reference. Comparison of FEDGLOSS with state-of-the-art approaches, distinguished in SAM-based methods (**a**, **c**) and SGD-based ones (**b**, **d**). FEDGLOSS consistently achieves the best performance. Model: CNN.



(a) $\alpha = 0$ SAM (b) $\alpha = 0$ SGD (c) $\alpha = 0.5$ SAM (d) $\alpha = 0.5$ SGD

Figure 16. CIFAR100 with varying degrees of heterogeneity ($\alpha \in \{0, 0.5\}$) with **CNN**. Results of centralized runs (*dashed lines*) added as reference. Comparison of FEDGLOSS with state-of-the-art approaches, distinguished in SAM-based methods (**a**, **c**) and SGD-based ones (**b**, **d**). FEDGLOSS consistently achieves the best performance.



(a) CIFAR100 $\alpha = 0.5$ SAM (b) CIFAR100 $\alpha = 0.5$ SGD (c) CIFAR10 $\alpha = 0.05$ SAM (d) CIFAR10 $\alpha = 0.05$ SGD

Figure 17. Accuracy trends with **ResNet18** on CIFAR100 (*left*) and CIFAR10 (*right*). Comparison of FEDGLOSS with state-of-the-art approaches, distinguished in SAM-based methods (**a**, **c**) and SGD-based ones (**b**, **d**). FEDGLOSS consistently achieves the best performance, both in terms of final accuracy and convergence speed.

methods depending on the used local optimizer. For a fair comparison, we report FEDSMOO with and without the scheduling of ρ (+*wp* in the figure). For additional details on the scheduling, refer to App. C. FEDGLOSS consistently achieves the best performances and convergence speedup in each group. In addition, we remind that FEDGLOSS communicates *half* the number of bits at each round w.r.t. FEDSMOO. Fig. 17 reports the results obtained with ResNet18 on both datasets: FEDGLOSS consistently achieves the best speed of convergence and final accuracy, with both SAM and SGD as CLIENTOPT.

B.4. Effectiveness in Multiple Scenarios and with Several Model Architectures

As already shown in Sec. 6, FEDGLOSS outperforms the state of the art in terms of speed of convergence, final performance in accuracy, flatness of the solution and communication efficiency. This remains true across multiple datasets (CIFAR100, CIFAR10, LANDMARKS-USER-160K) and model architectures (CNN, ResNet18, MobileNetv2). The main paper focuses on results in more challenging heterogeneous FL scenarios. The next paragraphs discuss the behavior of FEDGLOSS in increasingly more homogeneous settings.

Table 10. FEDGLOSS against SOTA algorithms in increasingly more homogeneous settings on CIFAR10, compared in terms of accuracy (%) and maximum Hessian eigenvalue λ_1. Best results in **bold**. Model: CNN.

	Method	$\alpha = 1$		$\alpha = 5$		$\alpha = 10$	
		Accuracy	λ_1	Accuracy	λ_1	Accuracy	λ_1
SGD-based	FEDAVG	76.4 ± 0.2	68.2 ± 2.7	83.7 ± 0.4	63.3 ± 2.9	82.4 ± 0.1	70.6 ± 0.3
	FEDPROX	76.3 ± 0.2	69.5 ± 0.9	83.8 ± 0.4	63.0 ± 3.6	82.4 ± 0.2	71.8 ± 2.0
	FEDDYN	77.1 ± 0.1	64.5 ± 4.9	82.3 ± 0.3	48.3 ± 1.6	82.2 ± 0.1	58.1 ± 3.5
	SCAFFOLD	79.6 ± 0.1	62.7 ± 2.7	$\mathbf{84.5 \pm 0.2}$	47.4 ± 3.0	83.8 ± 0.2	58.4 ± 2.9
	FEDGLOSS	$\mathbf{80.31 \pm 0.4}$	$\mathbf{40.0 \pm 5.2}$	83.5 ± 0.2	$\mathbf{8.5 \pm 0.3}$	$\mathbf{84.8 \pm 0.3}$	$\mathbf{41.4 \pm 2.2}$
SAM-based	FEDSAM	77.7 ± 0.2	25.6 ± 0.02	83.5 ± 0.1	19.7 ± 0.1	83.2 ± 0.2	22.1 ± 1.4
	FEDDYN	80.5 ± 0.2	21.0 ± 0.8	82.9 ± 0.4	16.3 ± 0.5	84.1 ± 0.1	17.3 ± 0.3
	FEDSPEED	76.2 ± 0.5	8.3 ± 0.1	83.6 ± 0.1	20.3 ± 0.5	78.2 ± 0.2	8.6 ± 0.2
	FEDGAMMA	70.3 ± 1.0	4.8 ± 0.2	83.5 ± 0.1	16.0 ± 0.6	76.4 ± 0.3	6.4 ± 0.1
	FEDSMOO	85.2 ± 0.2	9.8 ± 0.2	86.2 ± 0.2	6.0 ± 0.2	86.8 ± 0.3	6.2 ± 0.2
	FEDGLOSS	$\mathbf{85.6 \pm 0.2}$	$\mathbf{2.0 \pm 0.04}$	$\mathbf{86.9 \pm 0.2}$	$\mathbf{4.5 \pm 0.02}$	$\mathbf{87.1 \pm 0.1}$	$\mathbf{4.1 \pm 0.03}$

FEDGLOSS in moderately heterogeneous settings. The choice of the Dirichlet's concentration parameter α in the main text aligns with the FL literature [1, 4, 20, 53, 54], where $\alpha < 1$ is commonly used to evaluate challenging settings. Moreover, for $\alpha > 0.6$, differences between methods become less pronounced [61]. To further analyze the impact of varying heterogeneity levels on FEDGLOSS, we consider $\alpha \in \{1, 5, 10\}$ on CIFAR10. Tab. 10 presents a comparison of FEDGLOSS against several state-of-the-art FL algorithms. Consistently, FEDGLOSS achieves the highest accuracy across all tested scenarios, except for $\alpha = 5$ with SGD as the local optimizer, where it is outperformed by SCAFFOLD. Notably, even in this case, FEDGLOSS exhibits the lowest maximum Hessian eigenvalue, highlighting its stability.

FEDGLOSS in homogeneous settings. Tab. 11 evaluates FEDGLOSS against the main methods FEDAVG, FEDSAM, FEDDYN and FEDSMOO in homogeneous FL settings. Here, client gradients are naturally more aligned due to reduced client drift [26]. We thus test FEDGLOSS with and without ADMM for global consistency. As expected, FEDGLOSS achieves similar accuracy with or without ADMM, particularly when using SAM as the local optimizer. However, ADMM facilitates convergence to flatter minima (evidenced by lower λ_1 values) by aligning local and global convergence points. Notably, FEDGLOSS achieves the flattest minima (lowest λ_1) across both datasets, and the best accuracy on the more complex CIFAR100. While FEDSMOO achieves slightly higher accuracy on CIFAR10, FEDGLOSS finds a flatter minimum and achieves competitive accuracy with significantly lower communication costs (halved).

Table 11. FEDGLOSS against SOTA FL methods on homogeneous CIFAR settings, compared in terms of communication costs, accuracy (%) and maximum Hessian eigenvalue λ_1. Best result in **bold** and second best underlined. Model: CNN.

Method	Comm. Cost	ADMM	CIFAR10 $\alpha = 100$		CIFAR100 $\alpha = 1000$	
			Accuracy	λ_1	Accuracy	λ_1
FEDAVG	$1\times$	✗	84.0	68.4	50.1	49.4
FEDSAM	$1\times$	✗	84.7	36.2	53.4	32.6
FEDDYN (SGD)	$1\times$	✓	83.8	47.8	51.9	91.7
FEDDYN (SAM)	$1\times$	✓	84.5	27.9	52.5	46.0
FEDSMOO	$2\times$	✓	**85.1**	<u>6.4</u>	53.9	24.6
FEDGLOSS (SGD)	$1\times$	✗	84.0	67.7	50.5	50.8
	$1\times$	✓	83.1	7.1	51.7	47.9
FEDGLOSS (SAM)	$1\times$	✗	<u>84.8</u>	36.2	**55.8**	<u>13.9</u>
	$1\times$	✓	<u>84.8</u>	**2.8**	<u>55.7</u>	**11.8**

B.5. Reducing Communication Costs

Analysis on communication cost. Tab. 12 extends the analysis presented in Sec. 6.3.4 by evaluating the communication costs across all scenarios considered in this work. Notably, since FEDGLOSS maintains the per-round communication cost of FEDAVG, it remains advantageous even when matching the convergence speed of the best-performing algorithm (FEDSMOO), due to its *halved communication costs*.

Table 12. Communication costs comparison w.r.t. FEDAVG. "-" for not reached accuracy, "✗" for non-convergence.

	Method	CNN								ResNet18				MobileNetv2	
		CIFAR-10				CIFAR-100				CIFAR-10		CIFAR-100		LANDMARKS-USER-160K	
		$\alpha = 0$		$\alpha = 0.05$		$\alpha = 0$		$\alpha = 0.5$		$\alpha = 0.05$		$\alpha = 0.5$			
		Rounds	*Cost*	*Rounds*	*Cost*	*Rounds*	*Cost*	*Rounds*	*Cost*	*Rounds*	*Cost*	*Rounds*	*Cost*	*Rounds*	*Cost*
SGD-based	FEDAVG	$10k$ (1×)	$1B$	$10k$ (1×)	$1B$	$20k$ (1×)	$1B$	$20k$ (1×)	$1B$	$10k$ (1×)	$1B$	$10k$ (1×)	$1B$	$1.3k$ (1×)	$1B$
	FEDPROX	$7.6k$ (1.3×)	$0.8B$	$7.9k$ (1.3×)	$0.8B$	$18.7k$ (1.1×)	$0.9B$	$18.6k$ (1.1×)	$0.9B$	$8.8k$ (1.1×)	$0.9B$	$8.3k$(1.2×)	$0.8B$	-	-
	FEDDYN	$2k$ (5×)	$\mathbf{0.2B}$	$1.9k$ (5×)	$\mathbf{0.2B}$	✗		✗		-	-	$3.5k$ (2.9×)	$0.4B$	-	-
	SCAFFOLD	-	-	-	-	-	-	-	-	-	-	$8.9k$ (1.1×)	$1.8B$	✗	
	FEDGLOSS	$3.4k$ (2.9×)	$0.3B$	$3.8k$ (2.6×)	$0.4B$	$5k$ (4×)	$\mathbf{0.3B}$	$4.7k$ (4.3×)	$\mathbf{0.2B}$	$2.4k$ (4.2×)	$\mathbf{0.2B}$	$1.9k$ (5.3×)	$\mathbf{0.2B}$	-	-
SAM-based	FEDSAM	$6.3k$ (1.6×)	$0.6B$	$7.8k$ (1.3×)	$0.8B$	$18.3k$ (1.1×)	$0.9B$	$16.3k$ (1.2×)	$0.8B$	$9.2k$ (1.1×)	$0.9B$	$7.8k$ (1.3×)	$0.8B$	$1.3k$ (1×)	$1B$
	FEDDYN	$3k$ (3.3×)	$0.3B$	$4.2k$ (2.4×)	$0.4B$	✗		✗		$4.1k$ (2.4×)	$0.4B$	$3.5k$ (2.9×)	$0.4B$	-	-
	FEDSPEED	$6.3k$ (1.6×)	$0.6B$	$6.9k$ (1.4×)	$0.7B$	$18.3k$ (1.1×)	$0.9B$	$15.7k$ (1.3×)	0.8B	$8.3k$ (1.2×)	$0.8B$	$8.3k$ (1.2×)	$0.8B$	$1.3k$ (1×)	$1B$
	FEDGAMMA	-	-	-	-	-	-	-	-	$9.3k$ (1.1×)	$1.9B$	$8.1k$ (1.2×)	$1.6B$	✗	
	FEDSMOO	$1.9k$ (5.3×)	$0.4B$	$2.2k$ (4.5×)	$0.4B$	$4.5k$ (4.4×)	$0.5B$	$6.5k$ (3.1×)	$0.7B$	$2.4k$ (4.2×)	$0.5B$	$2.3k$ (4.3×)	$0.5B$	200 (6.5×)	$0.3B$
	FEDGLOSS	$2.2k$ (4.5×)	$\mathbf{0.2B}$	$2.2k$ (4.5×)	$\mathbf{0.2B}$	$6.3k$ (3.2×)	$\mathbf{0.3B}$	$5.2k$ (3.8×)	$\mathbf{0.3B}$	$2.4k$ (4.2×)	$\mathbf{0.2B}$	$1.9k$ (5.3×)	$\mathbf{0.2B}$	200 (6.5×)	$\mathbf{0.2B}$

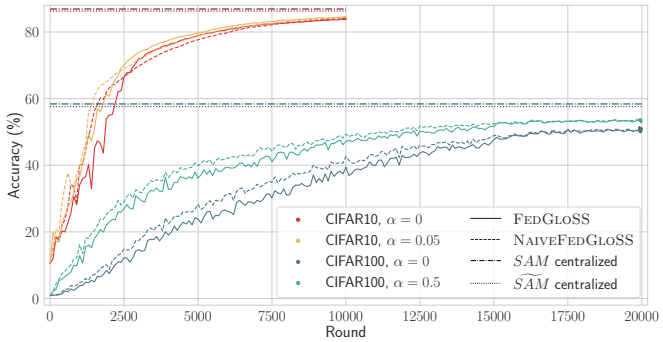


Figure 18. **Accuracy trends of FEDGLOSS *vs.* NAIVEFEDGLOSS.** The comparison includes the centralized upper bounds of SAM and $\widetilde{\text{SAM}}$ (the adaptation of FEDGLOSS' strategy to the centralized scenario). CNN on CIFAR10 and CIFAR100 with varying heterogeneity degree (α). NAIVEFEDGLOSS is $\approx 1.1\times$ faster than its efficient alternative FEDGLOSS in CIFAR100, while FEDGLOSS shows increased convergence speed after $\approx 25\%$ of training rounds in CIFAR10. However, both methods reach the same accuracy at the of training. These results motivate the choice of FEDGLOSS over NAIVEFEDGLOSS.

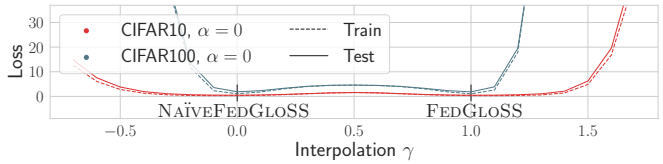


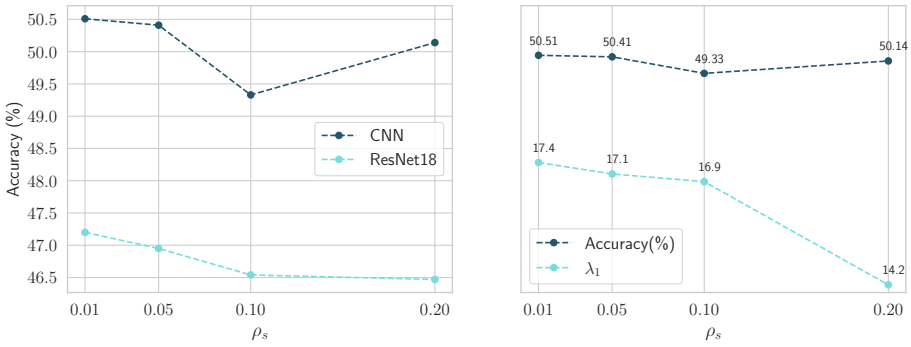
Figure 19. **Loss barriers** resulting from interpolating NAIVEFEDGLOSS and FEDGLOSS' models, which are found in the same basin. CIFAR datasets, CNN. Details on the computation in App. D.2.

FEDGLOSS *vs.* NAIVEFEDGLOSS. Fig. 18 deepens the comparison between FEDGLOSS and its baseline NAIVEFEDGLOSS, discussed in Sec. 6.4. In particular, this plot reports the accuracy trends of the two methods, showing that NAIVEFEDGLOSS is slightly faster ($\approx 1.1\times$) than FEDGLOSS in CIFAR100, while FEDGLOSS surpasses the speed of the baseline after $\approx 25\%$ of training in CIFAR10. However, both methods reach the same accuracy at the of training. In addition, it is to be noted that FEDGLOSS *halves* the communication cost w.r.t. NAIVEFEDGLOSS by transmitting half the number of bits at each round, as it eliminates the need to invoke the clients twice to compute the updates. Lastly, our sharpness approximation does not steer the optimization path: models trained with FEDGLOSS and NAIVEFEDGLOSS end up in the same basin (no loss barrier), with similar flatness (Fig. 19). These insights further support our choice of the efficient strategy of FEDGLOSS over NAIVEFEDGLOSS.

Convergence speed. Since all methods are compared over a fixed amount of communication rounds, **higher final accuracy implies faster convergence**. Since FEDGLOSS consistently outperforms the other state-of-the-art algorithms taken into account, it is guaranteed to converge faster, as also shown in the accuracy trends (Figs. 15 to 17).

B.6. The Impact of Server-side ρ_s

Fig. 20 analyzes the impact of ρ_s on the performance of the global model, both in terms of accuracy (Fig. 20a) and flatness of the solution (Fig. 20b). In details, Fig. 20a compares the accuracy of the global model trained on CIFAR100 when varying the model architecture (CNN *vs.* ResNet18) and the data heterogeneity ($\alpha = 0$ *vs.* $\alpha = 0.5$). In all the configurations, we note that a smaller value of ρ_s usually leads to the best results. Fig. 20b instead focuses on the CNN in the most heterogeneous setting ($\alpha = 0$) and compares the reached accuracy with the corresponding maximum Hessian eigenvalue λ_1 when varying ρ_s. A larger server-side ρ_s corresponds to a smaller λ_1, *i.e.*, a flatter region in the global loss landscape.



(a) FEDGLOSS ρ_s with different architectures

(b) FEDGLOSS ρ_s *vs.* λ_1

Figure 20. CIFAR100. **(a):** Accuracy (%) of FEDGLOSS when varying the server-side SAM ρ_s, with different heterogeneity and architecture (CNN with $\alpha = 0$ and ResNet18 with $\alpha = 0.5$). Smaller values of ρ_s lead to better performances. **(b):** FEDGLOSS ρ_s *vs.* maximum Hessian eigenvalue λ_1 on CNN with $\alpha = 0$. Larger values of ρ_s lead to lower eigenvalues with minimum loss in accuracy.

C. Implementation Details

This section delves into a comprehensive description of the datasets and models utilized throughout this paper, specifying the Deep Learning framework and the hardware employed (App. C.1). Additionally, we present the area of the hyper-parameters' space explored during the fine-tuning process in order to yield optimal results (App. C.2).

C.1. Datasets and Models

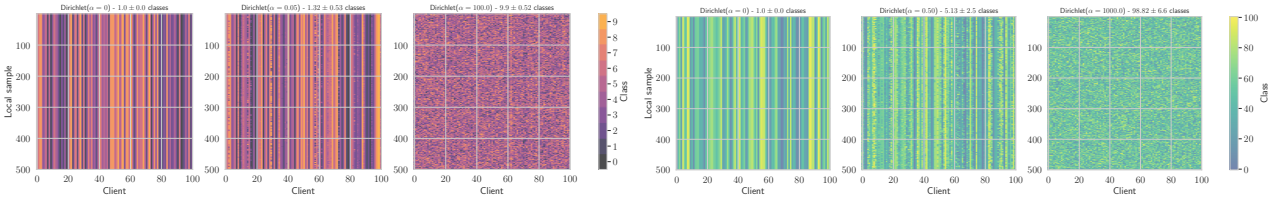
Tab. 13 provides a comprehensive overview of each dataset's general statistics. This includes the number of training clients participating in the process and the total number of samples used to construct both the training and test sets.

C.1.1. CIFAR10 and CIFAR100

We adapted these two well-known image classification datasets to the FL scenario by replicating the splits among clients proposed by [19]. Both datasets are split evenly among 100 clients, thus each of them has access to 500 data samples. This partitioning is performed according to a Latent Dirichlet Allocation (LDA) on the labels. In practice, each local dataset follows a multinomial distribution drawn according to a symmetric Dirichlet distribution with concentration parameter α. The higher the value of this parameter is, the more the local datasets resemble a homogeneous scenario, in the limit case $\alpha = 0$ each client has access to one only class of images. In our experiments we tested $\alpha \in \{0, 0.05, 1, 5, 10, 100\}$ and $\alpha \in \{0, 0.5, 1000\}$ for CIFAR10 and CIFAR100, respectively. Our choice of α aligns with FL literature [1, 4, 20, 53, 54], where $\alpha < 1$ is standard for testing challenging settings. Also, for $\alpha > 0.6$, method differences are less apparent [61]. Figs. 21a and 21b show how data is distributed across clients in all the experimental settings for these two datasets. Both datasets are pre-processed by applying random crops and random horizontal flips.

Table 13. Datasets' description with their general statistics on the size and number of clients.

Dataset	Train clients	Train samples	Test samples
CIFAR10	100	50,000	10,000
CIFAR100	100	50,000	10,000
LANDMARKS-USER-160K	1262	164,172	19,526



(a) CIFAR10 (b) CIFAR100

Figure 21. CIFAR10 (*left*) and CIFAR100 (*right*) data distribution across clients with the heterogeneity levels tested in the experiments. On top of each chart we report the average number of classes seen by each client.

Models. We trained a Convolutional Neural Network (CNN) inspired by the LeNet-5 architecture [30], as proposed by [20]. The network comprises two 64-channels convolutional layers, both using 5×5 kernels and followed by 2×2 max-pooling layers. This is succeeded by two fully-connected layers with 384 and 192 units, respectively. The final output layer is adapted to the specific number of classes in the dataset.

To explore deeper and more expressive architectures, we also trained a ResNet18 [16] on CIFAR100 with $\alpha = 0.5$ and CIFAR10 with $\alpha = 0.05$. We replaced the standard Batch Normalization layers [21] with Group Normalization layers [59] due to their demonstrated effectiveness in handling skewed data distributions in FL [18]. We carried out the experiments with the CNN model using PyTorch [43] and the ResNet18 ones using FedJAX [48].

C.1.2. LANDMARKS-USER-160K

To achieve a comprehensive understanding of the efficacy of the proposed method, a thorough analysis was undertaken utilizing large-scale real-world datasets. Specifically, we use the LANDMARKS-USER-160K dataset [20], a repository encompassing 164,172 images depicting 2028 distinct landmarks, distributed among 1262 clients.

Models. The model employed for training is a MobileNetV2 [20, 50], replacing batch normalization with group normalization layers and pre-trained on ImageNet [8]. The tests on this dataset were carried out on a cluster of NVIDIA A100 40GB, using our FedJAX codebase.

C.2. Hyperparameters

Grid search. In Tab. 14 we report the training hyperparameters associated to each dataset and model pairing, while Tab. 15 summarizes the hyper-parameters search grid tested for each method (in bold the chosen ones). All runs are averaged over 3 seeds. In addition, following previous works [1, 4, 5], we report the averaged accuracy of the last 100 rounds to reduce the noise typical of heterogeneous FL settings.

Local hyperparameters. As for local hyperparameters, we tested FEDGLOSS' robustness to $E = 2$ and $\eta_l = 0.001$ on CIFAR10 with $\alpha = 0$. A smaller η_l had the greatest impact, with FEDAVG and FEDSAM losing 45% and 77% of their performance (more local steps degrade performance in heterogeneous settings), respectively. Notably, FEDGLOSS loses only 29%. We attribute this to the pseudo-gradient consistently pointing toward higher-loss regions, ensuring a reliable ascent step approximation regardless of local hyperparameters.

Scheduling of ρ. While running our experiments on FEDGLOSS, we noticed that a larger value of local ρ allowed to reach the best final accuracy, while a smaller ρ achieved faster convergence in the initial training stages. Following this insight, we schedule the value of ρ for T_s rounds as

$$\rho(t) = \begin{cases} \rho_0 + {}^{(\rho-\rho_0)}/_{T_s} \cdot t & \text{if } t \leq T_s \\ \rho & \text{otherwise,} \end{cases}$$

starting from the value $\rho_0 = 0.001$.

D. Flatness Analysis

This section describes the procedure to compute the visualization of the loss landscapes and the Hessian eigenvalues.

Table 14. General training hyper-parameters common to all methods, distinguished by dataset and model architecture. Symbols: local epochs E, local learning rate η, weight decay wd, client-side momentum β_l, batch size B.

Dataset	Model	Rounds	Clients per round	Client optimization				
				E	η	wd	β_l	B
CIFAR10	CNN	10000	5	1	10^{-2}	$4 \cdot 10^{-4}$	0	64
CIFAR100	CNN	20000	5	1	10^{-2}	$4 \cdot 10^{-4}$	0	64
	ResNet18-GN	10000	10	1	10^{-2}	10^{-5}	0.7	64
LANDMARKS-USER-160K	MobileNetv2	1300	50	5	0.1	$4 \cdot 10^{-5}$	0	64

Table 15. Search grid used to find optimal hyper-parameters for each combination of method, dataset and model. We highlight the best performing values in **bold**.

Method	HParam	CIFAR10		CIFAR100		LANDMARKS-USER-160K
		CNN	ResNet18	CNN	ResNet18	
FEDSAM	ρ	[0.05, 0.1, **0.15**, 0.2]	[**0.01**, 0.02, 0.05]	[0.005, **0.01**, 0.02, 0.05]	[**0.01**, 0.02, 0.05]	[**0.05**]
FEDPROX	μ	[0.001, 0.01, **0.1**]	[0.001, **0.01**, 0.1]	[0.001, 0.01, **0.1**]	[0.001, 0.01, **0.1**]	[0.001, 0.01, **0.1**]
FEDDYN	α	[0.001, **0.01**, 0.1]	[0.001, **0.01**, 0.1]	[0.001, **0.01**, 0.1]	[**0.01**]	[**0.001**, 0.01]
	ρ (SAM-based only)	[**0.15**]	[**0.01**]	[0.01, **0.02**]	[**0.01**]	[**0.05**]
FEDSPEED	ρ	[0.05, 0.1, **0.15**, 0.2]	[**0.01**]	[0.005, **0.01**, 0.02, 0.05]	[**0.01**]	[**0.05**]
	α	[0.9, **0.95**, 0.99]	[0.9, **0.95**]	[0.9, **0.95**, 0.99]	[0.9, **0.95**]	[**0.95**]
	λ	[10, **100**, 1000]	[10, 100, **1000**]	[10, 100, **1000**]	[10, 100, **1000**]	[100, **1000**]
FEDGAMMA	ρ	[**0.15**]	[**0.01**]	[**0.01**]	[**0.01**]	[**0.05**]
FEDSMOO	ρ	[0.05, 0.1, **0.15**, 0.2]	[**0.01**]	[0.005, 0.01, 0.05, **0.1**, 0.2]	[**0.01**]	[0.05, **0.1**, 0.2, 0.3]
	β	[5, **10**, 100]	[1, 2, **5**, 10]	[10, **100**]	[5, **10**, 100]	[10, **50**, 100, 1000]
FEDGLOSS (ours)	ρ_s	[0.01, 0.1, **0.15**]	[**0.01**, 0.05, 0.1, 0.5]	[**0.01**, 0.05, 0.1, 0.2]	[**0.01**, 0.05, 0.1, 0.5]	[**0.005**, 0.01, 0.02]
	ρ	[0.05, 0.1, **0.15**, 0.2]	[**0.01**]	[0.05, 0.1, **0.2**]	[**0.01**]	[0.05, 0.1, 0.2, **0.3**]
	β	[5, **10**, 100]	[**1**, 2, 5, 10]	[10, **100**]	[**5**, 10, 100]	[10, **50**, 100]
	T_s	[1000, **2000**, 4000]	[**0**]	[1000, 2000, 5000, 10000, **15000**]	[**0**]	[**0**]

D.1. Visualizing 3D Loss Landscapes

We leverage techniques from [31] to visualize the loss landscapes of our models. We adapt their code to work with our specific datasets and network architectures. The process involves calculating the loss function along random directions in the parameter space. This is achieved by perturbing the model's parameters within a defined range. In our visualizations, we constrain these perturbations to occur within the range of $[-1, 1]$ for both the x and y axes. To ensure consistent comparisons across models (*e.g.*, as seen in Fig. 6), we utilize the same set of random directions for all models.

D.2. Visualizing 1D Loss Landscapes

Fig. 19 shows the interpolation of FEDGLOSS and NAIVEFEDGLOSS's models. Given their respective weights $\boldsymbol{w}_{\text{FEDGLOSS}}$ and $\boldsymbol{w}_{\text{NAIVEFEDGLOSS}}$, the interpolation is computed using a coefficient γ as

$$\boldsymbol{w} = \gamma \cdot \boldsymbol{w}_{\text{FEDGLOSS}} + (1 - \gamma) \cdot \boldsymbol{w}_{\text{NAIVEFEDGLOSS}} \,. \tag{8}$$

For each $\gamma \in [-1, 2]$, $\boldsymbol{w}$ is tested on the training or test sets, and the plot reports the computed loss. The resulting interpolation indicates that there is no loss barrier between the two models, suggesting they lie within the same basin. Additionally, the 1D geometry of the emerging loss landscape reveals that both models converge to a flat minimum when evaluated on both CIFAR100 and CIFAR10.

D.3. Hessian Eigenvalues for Flatness Measure

Following prior work [4, 12, 14, 31], we use the spectrum of the Hessian matrix to quantify the *flatness* of the loss landscape. Here, lower maximum eigenvalues correspond to flatter landscapes, implying less sharpness. To compute these eigenvalues (denoted by λ_1 in the main paper), we employ the stochastic power iteration method [60] with a maximum of 20 iterations, referring to the code of Golmant et al. [15].