A CLOSER LOOK AT WEIGHT AND FEATURE REGULARIZATION IN FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated learning trains a shared model through local updates on distributed client data. Many methods address data heterogeneity by encouraging local model weights, features, or predictions to remain close to those of the global model. However, these distances alone do not fully describe how regularization changes the model's predictions or learning updates. In this paper, we study regularization on weight, feature, and prediction through the forward and backward computations of a neural network. We examine what each penalty directly preserves and how changes in features, weights, and prediction errors propagate to layer outputs, backward signals, and task gradients. This provides a common view of existing formulations and clarifies which changes are constrained and which can remain during local training. Our experiments show that regularizers can affect signals beyond their direct targets, while stronger agreement with the global model does not consistently correspond to higher accuracy. These results are consistent with our analysis, helping explain why a regularizer is useful in some settings but less effective in others.

1 INTRODUCTION

Federated learning (FL) allows clients to train a shared model without collecting their data in one place (McMahan et al., 2017). Clients start from the global model and train on their own data before the server combines their updates. Since each client sees only its own data, local models can move in different directions over several training steps.

To limit such local drift, many methods use the global model as a reference. They discourage changes in model parameters (Li et al., 2020), compare intermediate representations (Li et al., 2021; Kim et al., 2024), or keep predictions close to those of the global model (Yao et al., 2021; Lee et al., 2022). In each case, a penalty expresses what should remain similar and a strength controls its influence on training. Choosing the quantity to compare and tuning this strength are often treated as separate design decisions, but both determine which changes the local model is encouraged to avoid.

The phrase "remain close," however, does not fully describe what is preserved. Consider a feature passed to the next layer. Its distance from the global feature describes how much it has moved, but not how much that movement changes the next layer's output. The feature also enters the weight gradient, so keeping the feature close can restrict changes in the learning signal as well. Likewise, a weight participates in both forward computation and backpropagation. These shared computations mean that a penalty may preserve more than its stated target, while still leaving other changes unconstrained.

In this work, we study weight, feature, and prediction penalties by examining both what they directly compare and the computations that use those quantities. We derive how feature and weight changes affect layer outputs, error signals passed backward, and gradients of the task loss. We then measure these changes by evaluating local and global models on the same inputs during federated training. Our contributions are as follows:

- We develop a unified analysis of weight-, feature-, and prediction-level regularization in FL by examining not only the quantities they directly constrain, but also their effects on forward and backward computations.
- We characterize how changes in features, weights, and prediction errors propagate to layer outputs, backward signals, and task gradients. Our analysis shows that penalties imposed on

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

different quantities can induce fundamentally different optimization directions and cannot, in general, be made equivalent by simply adjusting their coefficients.

- We empirically study representative weight-, feature-, and prediction-based regularizers. The results reveal how these penalties affect gradient changes across layers and show that stronger similarity to the global model or across clients does not necessarily translate into higher predictive accuracy.

2 Related Work

Data heterogeneity has been addressed through both local training and aggregation. Server momentum accumulates aggregated updates across rounds to stabilize the update direction (Hsu et al., 2019), while SCAFFOLD uses control variates to correct client drift during local optimization (Karimireddy et al., 2020) and FedDyn introduces dynamically updated local regularization Acar et al. (2021). We focus on regularization during local training that compares parameters, features, or predictions with those of a reference model.

FedProx penalizes the distance between local and global parameters to limit how far the local model moves during training (Li et al., 2020). MOON encourages local representations to resemble global representations rather than those of the previous local model (Li et al., 2021). FedDr+ uses feature distillation to keep local features close to global features, together with dot regression and a fixed classifier (Kim et al., 2024). Prediction-level methods build on knowledge distillation Hinton et al. (2015). FedGKD uses global predictions as distillation targets to reduce differences between local and global predictions, with variants that use historical global eachers (Yao et al., 2021). FedNTD instead matches the conditional distribution over incorrect classes, retaining their relative probabilities while allowing the task loss to fit the true class without a direct distillation penalty on its logit (Lee et al., 2022).

Our position. Rather than introducing a new regularizer, we examine how these penalties affect layer outputs, propagated errors, and gradients of the task loss. We identify the changes each penalty limits and those that can remain when other quantities change during local training. For example, keeping a feature close to its reference does not prevent the weight gradient from changing when the incoming error changes. Across layers and regularization strengths, we measure how much these signals differ from those of the global model.

3 Preliminaries

3.1 Federated learning

FL is a decentralized framework in which multiple clients collaboratively train a shared model without centralizing their raw data. Suppose there are N clients and client i has a local dataset $\mathcal{D}_i$. The global objective is

$$\arg\min_{w} \mathcal{J}(w) = \sum_{i=1}^{N} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} J_i(w), \qquad J_i(w) = \mathbb{E}_{(x,y)\sim\mathcal{D}_i}[\ell(w; x, y)], \tag{1}$$

where $|\mathcal{D}| = \sum_i |\mathcal{D}_i|$. At communication round r, participating clients initialize from the same global model, perform local optimization, and return their updated models to the server for aggregation.

3.2 Network notation and backpropagation

We analyze an arbitrary affine layer rather than assuming that the layer is a classifier. For layer l,

$$z_l = W_l h_l + b_l, \qquad h_{l+1} = f_l(z_l), \tag{2}$$

where $h_l \in \mathbb{R}^{d_l}$ is the layer input, $W_l \in \mathbb{R}^{d_{l+1}\times d_l}$ is the weight, and z_l is the pre-activation. We denote the backpropagated error signal by $\delta_l = \partial\ell/\partial z_l$. A superscript 0 denotes a reference quantity, typically obtained from the round-start global model: W_l^0, h_l^0, z_l^0.

4 WHAT DOES A REGULARIZER PRESERVE?

We begin with simple examples that use squared distances to keep local features and weights close to those of the global model. We then extend this idea to forward outputs and backward signals, and examine how the choice of what to preserve changes the regularization gradient.

4.1 A SIMPLE EXAMPLE: PRESERVING A FEATURE

Feature proximity. Focusing on one layer and omitting the layer index for readability, let $h^0, h \in \mathbb{R}^d$ be the features from the global and current local models for the same input, respectively, and let $W^0 \in \mathbb{R}^{m\times d}$ be the global weight. If we preserve the feature itself, a natural penalty is

$$R_{\text{feat}} = \frac{1}{2}\|h - h^0\|^2, \qquad \nabla_h R_{\text{feat}} = h - h^0 = \Delta h. \tag{3}$$

This penalty pulls the local feature toward h^0. The feature-distillation term in FedDr+ uses the same squared distance, up to normalization by the feature dimension (Kim et al., 2024).

Feature-output matching. Now suppose that, rather than keeping the feature itself close, we want it to remain compatible with the global model's next layer. Here compatibility means that a layer produces a similar output from h and h^0. We therefore pass both features through W^0, so their output difference is

$$(W^0 h + b^0) - (W^0 h^0 + b^0) = W^0 \Delta h. \tag{4}$$

Preserving this output gives

$$R_{\text{out}} = \frac{1}{2}\|W^0\Delta h\|^2 = \frac{1}{2}\Delta h^\top (W^0)^\top W^0 \Delta h, \quad \nabla_h R_{\text{out}} = (W^0)^\top W^0 \Delta h. \tag{5}$$

Compared with equation 3, the gradient now includes $(W^0)^\top W^0$. This $d \times d$ matrix describes how the reference layer responds to changes in the d feature coordinates. In contrast, equation 3 treats all directions equally: equal-sized feature changes receive equal penalties. The output penalty distinguishes them by how much they change the reference output. The matrix $(W^0)^\top W^0$ therefore appears because of what we chose to preserve.

4.2 A SIMPLE EXAMPLE: PRESERVING A WEIGHT

Weight proximity. For the same layer, let $\Delta W = W - W^0$. Keeping the weight itself close gives $R_{\text{weight}} = \frac{1}{2}\|\Delta W\|_F^2$ with $\nabla_W R_{\text{weight}} = \Delta W$. FedProx uses this distance form across all model weights (Li et al., 2020).

Weight-output matching. Now suppose that, rather than keeping the weight itself close, we want it to process the global model's features in the same way as the global weight. We give the reference feature h^0 to W and W^0. The output difference $Wh^0 - W^0h^0 = \Delta W h^0$ is caused by the weight change. Averaging this effect over the chosen features gives

$$R_{W\to z} = \frac{1}{2}\mathbb{E}\left[\|\Delta W h^0\|^2\right], \qquad \nabla_W R_{W\to z} = \Delta W\, \mathbb{E}\left[h^0 (h^0)^\top\right]. \tag{6}$$

The matrix $\mathbb{E}[h^0(h^0)^\top] \in \mathbb{R}^{d\times d}$ summarizes their directions and magnitudes. Unlike ΔW alone, this correction weights a weight change by how much it changes the outputs on those reference inputs.

4.3 A COMMON VIEW: WHAT CHANGES AND WHAT IS COMPARED

The examples above differ in two choices. First, we choose the quantity that is allowed to change, denoted by q. Second, we choose what consequence of that change to compare with the reference, denoted by $r(q)$.

For feature proximity, $q = h$ and $r(q) = h$. For feature-output matching, the same feature h changes, but we compare the output of the reference layer, so $r(q) = W^0 h$. Likewise, weight proximity uses $q = W$ and $r(q) = W$, whereas weight-output matching uses $r(q) = Wh^0$.

We write these comparisons as

$$R(q) = \mathbb{E}\left[\mathcal{D}\big(r(q^0), r(q)\big)\right], \tag{7}$$

where $\mathcal{D}$ specifies how the two responses are compared. The feature and weight examples above use squared distance.

For the squared-distance comparisons above, when the other quantities are fixed at their reference values, the response change has the common form

$$r(q) - r(q^0) = A^0 \Delta q. \tag{8}$$

Here A^0 is the fixed transformation from the change in q to the response change being compared. For example, $A^0 = I$ for feature proximity and $A^0 = W^0$ for feature-output matching.

For a vector-valued q, using half the squared response difference gives

$$R(q) = \frac{1}{2}\|A^0 \Delta q\|^2 = \frac{1}{2}\Delta q^\top M \Delta q, \qquad M = (A^0)^\top A^0, \qquad \nabla_q R = M \Delta q. \tag{9}$$

Thus M is determined by the response being compared; it is not an additional parameter. Feature proximity gives $M = I$, whereas feature-output matching gives $M = (W^0)^\top W^0$.

Backward computations give the same choices. For example, setting $q = W$ and $r(q) = W^\top \delta^0$ keeps the incoming error fixed and compares how the weight transmits it. Fixing W^0 instead and changing $q = \delta$ compares the incoming error itself or the signal $(W^0)^\top \delta$ transmitted to the preceding layer.

Table 1 collects these squared-distance comparisons. The δ rows are used later to measure the incoming-error contribution to the backward and weight-gradient changes.

Table 1: Squared-distance comparisons for one input. All quantities other than q are fixed at their reference values. The third column shows the response change that is penalized.

Comparison	q	$r(q) - r(q^0)$	$\nabla_q R$
Feature proximity	h	Δh	Δh
Feature-output matching	h	$W^0 \Delta h$	$(W^0)^\top W^0 \Delta h$
Feature-gradient matching	h	$\delta^0 \Delta h^\top$	$\|\delta^0\|^2 \Delta h$
Weight proximity	W	ΔW	ΔW
Weight-output matching	W	$\Delta W h^0$	$\Delta W h^0 (h^0)^\top$
Weight-backward matching	W	$(\Delta W)^\top \delta^0$	$\delta^0 (\delta^0)^\top \Delta W$
Error proximity	δ	$\Delta \delta$	$\Delta \delta$
Error-backward matching	δ	$(W^0)^\top \Delta \delta$	$W^0 (W^0)^\top \Delta \delta$
Error-gradient matching	δ	$\Delta \delta (h^0)^\top$	$\|h^0\|^2 \Delta \delta$

Prediction comparisons. The same framework also covers prediction matching. Here $q = z$ is the classifier logit, while the compared quantity is a probability distribution. Let $p(z) = \mathrm{softmax}(z)$ and $\pi_c(z; y) = p_c(z)/(1 - p_y(z))$ for $c \neq y$. The reference distributions are $p^0 = p(z^0)$ and $\pi^0 = \pi(z^0; y)$.

The two rows correspond to GKD (Yao et al., 2021) and NTD (Lee et al., 2022), respectively. Both are instances of equation 7. The fixed $M\Delta q$ form in equation 9 is specific to a squared distance whose response change is linear in q; the softmax–KL objectives are differentiated in their original form.

In the next section, we examine how these output-level penalties affect the incoming errors and weight gradients of earlier layers. The two rows correspond to GKD (Yao et al., 2021) and NTD (Lee et al., 2022), respectively. For NTD, the true-class logit receives no correction from the distillation term, while each non-target logit receives $\pi_j - \pi_j^0$. This signal is propagated through the same preceding network. Both are instances of equation 7. The fixed $M\Delta q$ form in equation 9 is specific to a squared distance whose response change is linear in q; the softmax–KL objectives are differentiated in their original form.

Table 2: Prediction comparisons used in our analysis at temperature one.

Comparison	$r(z)$	Penalty R	$\nabla_z R$
Full-prediction matching (GKD)	$p(z)$	$D_{\mathrm{KL}}(p^0 \| p(z))$	$p - p^0$
Non-target prediction matching (NTD)	$\pi(z; y)$	$D_{\mathrm{KL}}(\pi^0 \| \pi(z; y))$	0 for y; $\pi_j - \pi_j^0$ for $j \neq y$

5 What Else Is Preserved, and What Can Still Change?

A feature, weight, error, or prediction participates in more than one calculation. We first examine what a simple distance also limits, then how several changes combine in the actual learning gradient. Finally, we follow feature and prediction penalties into earlier layers.

5.1 Proximity limits some, but not all, signal changes

The previous section described what each penalty compares directly. We now ask what other learning signals are affected by that change. For an affine layer, the task gradient of the weight is $\nabla_W \ell = \delta h^\top$, so a change in the feature also changes the weight gradient. If the incoming error is held at its reference value δ^0, the feature-induced change is $\delta^0 \Delta h^\top$. Its squared norm satisfies

$$\|\delta^0 \Delta h^\top\|_F^2 = \|\delta^0\|^2 \|\Delta h\|^2. \tag{10}$$

Thus feature proximity also limits the feature-induced weight-gradient change in Table 1, even though it does not compare that gradient directly. Conversely, penalizing this change gives the same feature correction scaled by $\|\delta^0\|^2$. The scale is sample-dependent and is averaged with the sample's penalty. Weight proximity similarly bounds the fixed-input and fixed-error effects:

$$\|\Delta W h^0\| \leq \|\Delta W\|_F \|h^0\|, \qquad \|(\Delta W)^\top \delta^0\| \leq \|\Delta W\|_F \|\delta^0\|. \tag{11}$$

FedProx's whole-parameter distance includes each weight block. Thus weight proximity already limits both fixed-reference effects, although it does not weight their directions in the same way as the response-specific penalties in Table 1. These bounds concern fixed reference inputs or errors, not every signal in the current model.

What strength adjustment can reproduce. A distance-only step $q^+ = q - \eta\lambda\Delta q$ gives

$$r(q^+) - r(q^0) = (1 - \eta\lambda)\big(r(q) - r(q^0)\big) \tag{12}$$

for a fixed linear response. Thus $0 < \eta\lambda < 1$ shrinks that response without a separate penalty. This is a direct update of q, not a full training step. We can also ask whether its correction approximates a scaled response gradient. For a vector-valued q and a fixed nonzero Δq, the closest multiple of Δq to $M\Delta q$ is $\alpha\Delta q$, where

$$\alpha = \frac{\Delta q^\top M \Delta q}{\|\Delta q\|^2}, \qquad t = M\Delta q - \alpha\Delta q. \tag{13}$$

For $\alpha > 0$, holding the coefficients fixed and scaling the response penalty by λ/α gives

$$\frac{\|\lambda\Delta q - (\lambda/\alpha) M\Delta q\|}{\|\lambda\Delta q\|} = \frac{\|t\|}{\alpha\|\Delta q\|}. \tag{14}$$

A small right-hand side makes the corrections close. Choosing λ alone does not ensure this, and the correspondence can change across states. Appendix A.3 gives the derivation. The fixed-coefficient experiments below do not fit α or test this approximation.

When the other quantities also change. The actual weight-gradient difference contains both feature and error changes. Adding and subtracting $\delta^0 h^\top$ gives

$$\|\delta h^\top - \delta^0 (h^0)^\top\|_F \leq \underbrace{\|\delta^0\|\|\Delta h\|}_{\text{feature effect}} + \underbrace{\|h\|\|\Delta\delta\|}_{\text{remaining error effect}}. \tag{15}$$

Feature proximity limits the first term, while the incoming error can still change. For the current forward and backward signals,

$$\begin{aligned} \|z - z^0\| &\le \|\Delta W\|_F \|h^0\| + \|W\|_2 \|\Delta h\| + \|\Delta b\|, \\ \|W^\top \delta - (W^0)^\top \delta^0\| &\le \|\Delta W\|_F \|\delta^0\| + \|W\|_2 \|\Delta \delta\|. \end{aligned} \tag{16}$$

Here $\Delta b = b - b^0$. Weight proximity bounds the first terms, but changes in the inputs, errors, and bias also matter. These finite-change bounds motivate measuring the actual signal alongside the conditional effects. Their proofs and convolutional form are in Appendix A.2.

5.2 How individual changes combine in the learning gradient

The bounds separate possible sources of change. To measure their combined result, we also need how the terms add. For the forward and backward calculations,

$$z - z^0 = W^0 \Delta h + \Delta W h^0 + \Delta W \Delta h + \Delta b, \tag{17}$$

$$W^\top \delta - (W^0)^\top \delta^0 = (\Delta W)^\top \delta^0 + (W^0)^\top \Delta \delta + (\Delta W)^\top \Delta \delta. \tag{18}$$

Each has two conditional effects and their interaction; the forward output also includes the bias change. The weight-gradient difference has the same structure:

$$\delta h^\top - \delta^0 (h^0)^\top = \underbrace{\delta^0 \Delta h^\top}_{a} + \underbrace{\Delta \delta (h^0)^\top}_{b} + \underbrace{\Delta \delta \Delta h^\top}_{c}. \tag{19}$$

Let $S = \|a\|_F^2 + \|b\|_F^2 + \|c\|_F^2$. Then

$$\|a + b + c\|_F^2 = S + 2\langle a, b \rangle_F + 2\langle a, c \rangle_F + 2\langle b, c \rangle_F. \tag{20}$$

S measures the separate changes; the left-hand side measures the resulting gradient. Negative cross terms allow partly compensating changes, while positive terms increase the combined discrepancy. Keeping each effect small and keeping their sum close are therefore different requirements. We use this distinction to analyze existing training runs, not to rank separate and joint training objectives.

5.3 How feature and prediction penalties reach earlier layers

A feature penalty is backpropagated through the layers that produce the feature. A weight-distance penalty instead adds a gradient directly to each penalized weight. Prediction penalties also reach earlier layers. Thus a final-layer loss need not have a final-layer-only effect.

At the softmax–cross-entropy output $z_L = W_L h_L + b_L$, the task error is $\delta_L = p - \mathrm{onehot}(y)$. For the same label and temperature one, $\Delta \delta_L = \Delta p = p - p^0$. Full KL gives

$$\nabla_{z_L} R_{\mathrm{KD}} = \Delta p, \qquad \nabla_{h_L} R_{\mathrm{KD}} = W_L^\top \Delta p. \tag{21}$$

The current weight transmits the penalty gradient into the feature extractor. But the actual backward task signal at the feature changes by

$$W_L^\top \delta_L - (W_L^0)^\top \delta_L^0 = W_L^\top \Delta p + (\Delta W_L)^\top \delta_L^0. \tag{22}$$

The second term comes from the changed classifier weight, not from the prediction difference. The same distinction holds further back: the prediction correction is propagated through the current network, while the reference-to-current task-signal difference also reflects changes in the network that transmits it. Appendix A.4 gives the chain-rule expressions. The experiments measure the actual layer signals rather than separate Jacobian-change terms.

For NTD, the true-class logit receives no gradient from the distillation term, while each non-target logit j receives $\pi_j - \pi_j^0$. This correction is also propagated through the preceding layers. To see what the conditional comparison leaves unspecified, full KL separates as (Zhao et al., 2022)

$$D_{\mathrm{KL}}(p^0 \| p) = \underbrace{D_{\mathrm{KL}}(\mathrm{Bern}(p_y^0) \| \mathrm{Bern}(p_y))}_{\text{true-class mass}} + (1 - p_y^0) \underbrace{D_{\mathrm{KL}}(\pi^0 \| \pi)}_{\text{non-target distribution}}. \tag{23}$$

NTD uses the unweighted conditional KL and therefore does not directly compare the true-class mass. This does not imply that the mass must move farther from the reference; it can also decrease during training. At the output, probability changes are exactly task-error changes; earlier-layer errors additionally depend on the layers transmitting them. We therefore report the non-target and true-class prediction changes separately from the actual layer gradients, without treating every remaining change as a quantity that should be additionally penalized.

Table 3: Global evaluation accuracy (%). CIFAR results average rounds 451–500; Tiny ImageNet results average rounds 81–100. **Bold** indicates the highest reported value in a column.

Method	Coefficient	CIFAR-10		CIFAR-100		Tiny ImageNet	
		IID	$\beta = 0.1$	IID	$\beta = 0.1$	IID	$\beta = 0.1$
FedAvg	—	90.30	63.91	63.73	58.82	42.38	34.08
FedProx	0.01	90.30	64.36	63.01	59.06	42.35	34.19
	0.1	90.00	64.78	62.72	58.24	39.40	33.21
	1	87.62	68.35	55.13	51.49	23.33	20.06
Feature L2	0.01	90.48	63.30	63.91	59.12	42.27	33.94
	0.1	**90.73**	62.95	63.60	58.90	42.67	34.54
	1	90.55	62.80	64.25	60.22	**43.07**	33.08
GKD	0.01	90.44	64.41	63.66	59.26	42.70	34.14
	0.1	90.42	63.55	64.41	59.92	42.33	34.24
	1	90.68	64.92	64.20	56.14	40.23	30.47
NTD	0.01	90.42	63.47	63.66	59.19	42.71	34.51
	0.1	90.41	69.03	64.13	60.56	42.40	35.25
	1	90.50	**76.65**	**64.94**	**60.66**	42.08	**35.80**

6 EXPERIMENTS

We examine three consequences of the analysis: a penalty can affect signals beyond its direct target; conditional changes can compensate or reinforce one another; and closer reference signals need not imply better accuracy.

6.1 EXPERIMENTAL SETUP

Datasets and training. We use CIFAR-10, CIFAR-100 (Krizhevsky, 2009), and Tiny ImageNet Le & Yang (2015) with ResNet-18 (He et al., 2016). Each dataset is divided among 100 clients, with either IID sampling or a classwise Dirichlet partition with $\beta = 0.1$. We sample 10 clients per round and train each for five local epochs. SGD uses batch size 64, momentum 0.9, weight decay 0.001, and learning rate 0.01, multiplied by 0.998 after each round. CIFAR experiments run for 500 rounds and Tiny ImageNet for 100. We report evaluation accuracy over the final 50 and 20 rounds.

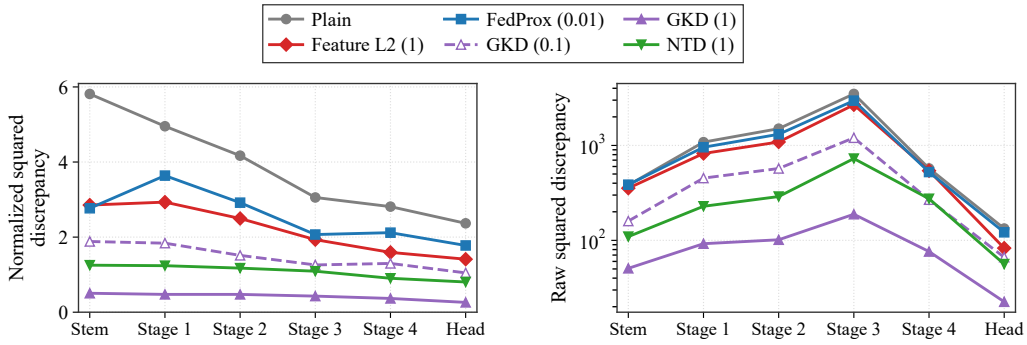
Compared methods. FedAvg (Plain) provides the reference without an added penalty. FedProx sums parameter distances. Feature L2 uses FedDr+'s feature distillation term with cross-entropy and a trainable classifier. GKD uses full prediction KL with the current global model as teacher; NTD excludes the true class before comparing distributions. Both use temperature one. Appendix B specifies preprocessing and the exact objectives.

Measurements. We evaluate the global model at start, selected local models, and the aggregated model. We record features, weights, task errors, and their forward, backward, and weight gradient effects at 21 affine layers. Each probe divides its mean squared change by its mean reference squared energy; averaging these layer ratios does not give a pooled network gradient norm. Unless stated otherwise, diagnostic summaries average rounds 400, 450, and 500 for CIFAR and rounds 80, 90, and 100 for Tiny.

6.2 PRESERVATION EFFECTS EXTEND BEYOND THE COMPARED FEATURE

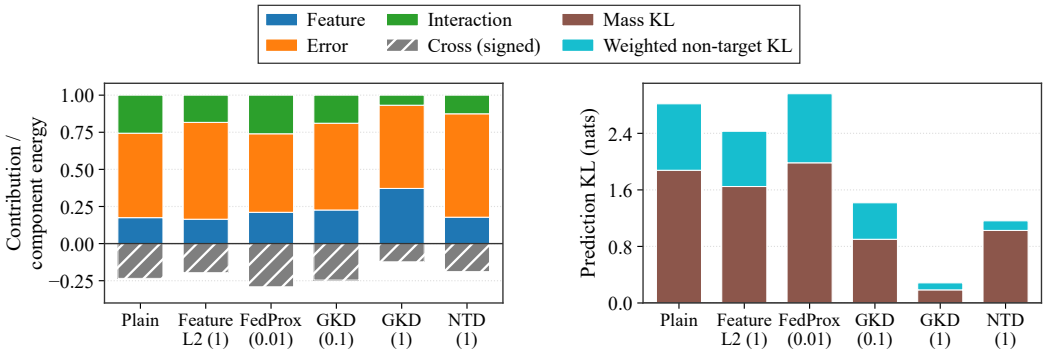
Table 3 first shows that a coefficient has different outcomes across settings. FedProx with $\mu = 1$ improves on Plain by 4.44pp on CIFAR-10 with $\beta = 0.1$, but decreases accuracy by 7.33 and 14.02pp on CIFAR-100 and Tiny ImageNet. NTD with $\lambda = 1$ improves CIFAR-10 by 12.74pp under $\beta = 0.1$, compared with 0.20pp under IID sampling. These contrasts motivate examining the signals themselves instead of attributing performance to proximity alone.

Figure 1 connects this observation to Section 5.3. Feature L2 with $\lambda = 1$ has a smaller relative weight gradient change than Plain from the stem (2.854 versus 5.813) to the classifier (1.411 versus 2.368). GKD and NTD also show changes throughout the network. The final feature enters the



(a) Relative change (b) Raw change

Figure 1: Weight-gradient changes at six affine layers of ResNet-18 on CIFAR-100 with $\beta = 0.1$. Curves compare late local models with their round-start references. The second panel reports the raw squared change; the first divides by each layer's reference energy. Feature L2 compares only the final feature, but the measured differences extend to earlier layers.



(a) Classifier gradient (b) Prediction divergence

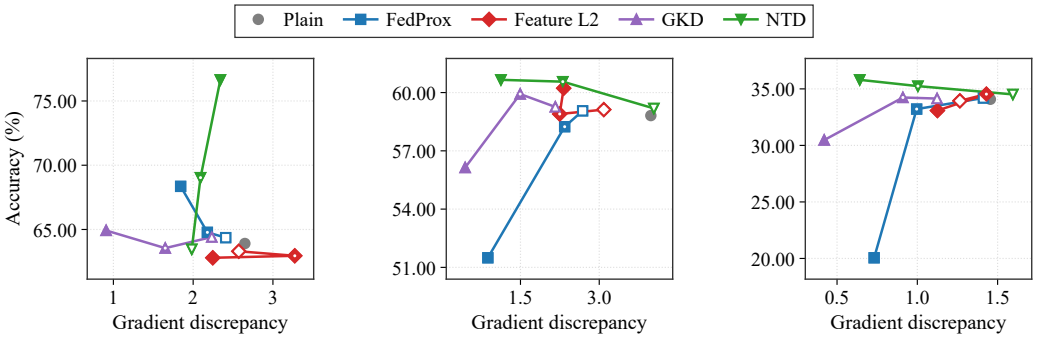
Figure 2: Classifier measurements on CIFAR-100 with $\beta = 0.1$. (a) The feature, error, and interaction energies in equation 19 are divided by their sum S; the signed cross terms are plotted below zero. (b) Full prediction KL is the sum of true-class mass KL and the weighted non-target term in equation 23.

classifier's task gradient, and penalties on the feature or prediction propagate into earlier layers. Thus the location of a comparison does not describe the full extent of its effects. This does not make a final feature penalty equivalent to FedProx, which directly penalizes all trainable parameters.

6.3 Individual effects differ from their combined result

Figure 2(a) measures the terms in equation 19. For Feature L2 with $\lambda = 1$, the feature, incoming-error, and interaction components contribute 0.164, 0.652, and 0.183 of S, respectively. Feature proximity therefore does not describe the complete gradient change. The cross terms further reduce its squared magnitude to 0.804 of S. For Plain, the corresponding ratio is 0.766. Measuring each component and adding its squared norm would miss this compensation. These ratios describe signal changes, not contributions to an accuracy gain.

The distinction also appears backward. At the classifier, the incoming-error component accounts for 0.982 of the backward component energy for Plain and 0.979 for NTD with $\lambda = 1$. A small conditional weight effect does not determine the actual backward change. Appendix C reports forward and backward decompositions and the finite-bound measurements.



(a) CIFAR-10 (b) CIFAR-100 (c) Tiny ImageNet

Figure 3: Accuracy and relative weight gradient discrepancy with $\beta = 0.1$. Open, white-centered, and filled markers denote coefficients 0.01, 0.1, and 1, respectively. Discrepancies average the 21 layer ratios and late probes. Accuracy uses the windows in Table 3.

6.4 Full and non-target matching retain different information

Figure 2(b) applies the KL decomposition in Section 5.3. Relative to Plain, NTD with $\lambda = 1$ has about 84% smaller conditional non-target KL, from 2.011 to 0.318. True-class mass KL also decreases, from 1.878 to 1.027, although NTD does not compare it directly. It remains about 88% of the reported mean full KL. Thus leaving a quantity out of the objective does not imply that it stays fixed or necessarily moves farther away.

GKD with $\lambda = 1$ gives smaller values in both prediction terms, but lower accuracy than NTD. The same distinction appears on CIFAR-10 and Tiny ImageNet: NTD retains a larger prediction discrepancy while achieving higher accuracy. The decomposition identifies what differs between the comparisons; it does not imply that adding a penalty on the remaining mass would improve NTD.

6.5 Closer signals do not give a consistent accuracy ranking

Figure 3 compares measured preservation with accuracy. On CIFAR-100, increasing GKD from 0.1 to 1 decreases the gradient discrepancy from 1.494 to 0.446 while accuracy decreases from 59.92% to 56.14%. FedProx similarly reduces the discrepancy from 2.684 to 0.879 between 0.01 and 1, with a 7.57pp accuracy decrease. On CIFAR-10, however, FedProx with 1 improves accuracy while reducing its discrepancy. Preservation is therefore neither uniformly helpful nor uniformly harmful.

For GKD $0.1 \to 1$, classifier task-gradient cosine on common inputs rises from 0.515 to 0.862, despite lower accuracy. The Stage-4 mean cosine instead falls from 0.346 to 0.325. These compare different quantities; neither determines accuracy. Further aggregation measurements appear in Appendix C.

7 Conclusion

We studied regularization on weight, feature, and prediction in federated learning through the forward and backward computations in which the constrained quantities participate. This view separates the quantity directly compared with a reference from the downstream signals that the comparison indirectly affects. Proximity penalties limit specific conditional changes, but the actual forward, backward, and task-gradient differences also depend on changes in other quantities, their interactions, and possible compensation between them. Our experiments reflect these distinctions across layers and objectives. Regularizers applied to a final feature or prediction can alter learning signals throughout the network, while stronger agreement with the global model does not consistently correspond to higher accuracy. These observations suggest evaluating a regularizer through three separate quantities: its direct target, the resulting learning signals, and predictive performance. Raw and normalized discrepancies should also be distinguished when their reference energies change over training. More broadly, the analysis provides a way to interpret what a regularization objective preserves and what remains free to change, rather than treating distance to the global model as a single notion of drift.

REFERENCES

Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N. Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. In *International Conference on Learning Representations*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, June 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

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. URL https://arxiv.org/abs/1909.06335.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank J. Reddi, Sebastian U. Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5132–5143. PMLR, 2020.

Seongyoon Kim, Minchan Jeong, Sungnyun Kim, Sungwoo Cho, Sumyeong Ahn, and Se-Young Yun. FedDr+: Stabilizing dot-regression with global feature distillation for federated learning. *arXiv preprint arXiv:2406.02355*, 2024.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 2015.

Gihun Lee, Minchan Jeong, Yongjin Shin, Sangmin Bae, and Se-Young Yun. Preservation of the global knowledge by not-true distillation in federated learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10713–10722, 2021.

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

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 2017.

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.

Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11953–11962, June 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Zhao_Decoupled_Knowledge_Distillation_CVPR_2022_paper.html.

AI USE STATEMENT

We used generative AI tools for language editing, literature search, and occasional research brainstorming. We also used them to assist with parts of the mathematical reasoning and proofs. All references, technical claims, and mathematical arguments were checked by the authors. We did not use generative AI to generate synthetic data or to draft complete sections of the paper. The authors are responsible for the final content of the paper.

A DERIVATIONS FOR THE COMPARISONS AND BOUNDS

A.1 REFERENCE COMPARISONS AND MATRIX DERIVATIVES

For a fixed reference, let $e_r = r(q) - r(q^0)$. Squared comparison gives $R = \frac{1}{2}\|e_r\|^2$ and $\nabla_q R = (\partial r/\partial q)^\top e_r$. For $r(q) = A^0 q + b^0$, this becomes $(A^0)^\top A^0 \Delta q$. The reference bias cancels. Finite sample averaging gives the stated M when Δq is common; input-dependent features instead require averaging the per-input losses before backpropagation.

For the weight-forward residual $u = \Delta W h^0$, differentiation yields

$$dR = u^\top (dW) h^0 = \langle u (h^0)^\top, dW \rangle_F, \qquad \nabla_W R = \Delta W h^0 (h^0)^\top.$$

For the backward residual $v = (\Delta W)^\top \delta^0$,

$$dR = v^\top (dW)^\top \delta^0 = \langle \delta^0 v^\top, dW \rangle_F, \qquad \nabla_W R = \delta^0 (\delta^0)^\top \Delta W.$$

The averaged second moments have shapes $d \times d$ and $m \times m$, respectively. With W^0 fixed, $\frac{1}{2}\|(W^0)^\top \Delta\delta\|^2$ gives $W^0 (W^0)^\top \Delta\delta$. These are derivatives with respect to the named variables. Optimizing a penalty on a computed error would also differentiate its dependence on the parameters; no such hidden-error penalty is trained in this suite.

A.2 FINITE CHANGES AND CONVOLUTIONAL WEIGHTS

The identity $\|uv^\top\|_F = \|u\|\|v\|$ follows by summing $u_j^2 v_k^2$. Applying the triangle inequality to

$$\delta h^\top - \delta^0 (h^0)^\top = \delta^0 \Delta h^\top + \Delta\delta h^\top$$

proves equation 15. Similarly, $\Delta z = \Delta W h^0 + W \Delta h + \Delta b$ and $\Delta(W^\top \delta) = (\Delta W)^\top \delta^0 + W^\top \Delta\delta$, with $\|Wx\| \le \|W\|_2 \|x\|$ and $\|W\|_2 \le \|W\|_F$, give equation 16. Averaging the per-example bounds preserves the inequalities but does not replace products by products of means.

For a convolutional weight, its per-input task gradient is $\sum_s \delta_s h_s^\top$, where h_s is the input patch and s indexes spatial positions. Its components are

$$a = \sum_s \delta_s^0 \Delta h_s^\top, \quad b = \sum_s \Delta\delta_s (h_s^0)^\top, \quad c = \sum_s \Delta\delta_s \Delta h_s^\top.$$

The actual change is $a + b + c$, and its squared norm includes spatial cross terms. The single outer-product norm identity is not applied to an entire convolutional kernel without the spatial sum.

A.3 SCALAR FITTING AND ITS SCOPE

A step using only the distance penalty gives $\Delta q^+ = (1 - \eta\lambda)\Delta q$. For a fixed linear response,

$$r(q^+) - r(q^0) = (1 - \eta\lambda)(r(q) - r(q^0)). \tag{24}$$

This contracts the response when $0 < \eta\lambda < 1$. It is a direct update of q, not a complete parameter training step. For a fixed positive-semidefinite M and nonzero Δq, differentiating $\|M\Delta q - a\Delta q\|^2$ with respect to a gives the stated α. With $t = M\Delta q - \alpha\Delta q$, $\Delta q^\top t = 0$ and

$$\|M\Delta q - a\Delta q\|^2 = \|t\|^2 + (a - \alpha)^2 \|\Delta q\|^2.$$

For $\alpha > 0$, $(\lambda/\alpha) M\Delta q = \lambda\Delta q + (\lambda/\alpha) t$. Hence $\|t\| \le \varepsilon\alpha\|\Delta q\|$ implies

$$\|(\lambda/\alpha) M\Delta q - \lambda\Delta q\| \le \varepsilon \|\lambda\Delta q\|.$$

This is a conditional error bound, not evidence that the residual is small. The coefficients are fixed during differentiation. At $\alpha = 0$, positive semidefiniteness gives $M\Delta q = 0$ and λ/α is undefined. Matrix variables use the Frobenius inner product and the corresponding linear operator. For $q = q(\theta)$, the parameter-gradient difference is multiplied by $(\partial q/\partial\theta)^\top$, so closeness in q is not automatically the same as relative closeness in parameter space. No matrix or fitted-scalar replacement is trained in the current experiments.

A.4 Feature and prediction gradients

For one input in evaluation mode, write $K_l = \partial h_L/\partial z_l$ and $J_l = \partial z_L/\partial z_l$. The half-squared final-feature penalty gives $K_l^\top \Delta h_L$, while full prediction KL gives $J_l^\top \Delta p$. The implemented feature mean-squared error changes the constant scale, not this chain-rule path. At an affine layer the added weight-gradient contribution is the arriving penalty gradient times $h_l^\top$; shared weights require summation.

For the CE task, $\delta_l = J_l^\top \delta_L$ and $\delta_l^0 = (J_l^0)^\top \delta_L^0$. Consequently,

$$\Delta\delta_l = J_l^\top \Delta p + (J_l - J_l^0)^\top \delta_L^0.$$

This derivation explains why prediction matching is not the same as matching every internal task error. The experiments report actual errors and gradients, not the two Jacobian terms separately. For batch-dependent training computations, the corresponding identities use the full batch Jacobian; the per-input version matches the evaluation-mode probes.

A.5 Full and non-target prediction matching

For $j \neq y$, substitute $p_j = (1 - p_y)\pi_j$ in full KL:

$$D_{\mathrm{KL}}(p^0 \| p) = p_y^0 \log \frac{p_y^0}{p_y} + (1 - p_y^0) \log \frac{1 - p_y^0}{1 - p_y} + (1 - p_y^0) \sum_{j \neq y} \pi_j^0 \log \frac{\pi_j^0}{\pi_j}.$$

This is equation 23. The teacher-mass weighting is applied per input before averaging; it is absent from the native NTD objective. Differentiation gives $\nabla_{z_L} R_{\mathrm{KD}} = \Delta p$ and the NTD signal v, with $v_y = 0$ and $v_j = \pi_j - \pi_j^0$ for $j \neq y$. To connect the probability split to the task error, embed π as $\bar{\pi}$ with a zero in position y. Then

$$\Delta\delta_L = \Delta p = (1 - p_y)v + \Delta p_y(\mathrm{onehot}(y) - \bar{\pi}^0).$$

The first term is a weighted non-target signal and the second changes the true-class probability. Passing either through a fixed linear map gives the corresponding transmitted component. Their squared norms also have a cross term. This identity does not equate the KL terms with squared backward-error terms, or prove that either term must harm learning.

B Experimental Protocol

Data and architecture. CIFAR-10 and CIFAR-100 each have 50,000 training images and 10,000 test images at 32×32 resolution. Tiny ImageNet has 100,000 training images over 200 classes; we evaluate its 10,000 labeled validation images at 64×64 resolution. No held-out data are used to adjust training coefficients. Training uses random horizontal flips and random crops with four pixels of padding. Evaluation uses the original image with dataset normalization. ResNet-18 uses a 3×3 stride-one stem and omits the ImageNet stem max-pooling layer. The feature comparison uses the final pooled representation.

Client data and optimization. IID partitioning randomly permutes the training set and splits it equally across 100 clients; it does not explicitly stratify each client's labels. The non-IID partition samples a Dirichlet allocation for each class with concentration $\beta = 0.1$, balances allocation against client capacity, and requires at least 64 examples per client. Ten clients participate per round. Each runs five epochs of SGD, with incomplete minibatches dropped. Batch size is 64, momentum is 0.9, weight decay is 0.001, and the initial learning rate is 0.01 with multiplicative round decay 0.998.

Exact losses. For reference parameters θ^0 and final features h_i^0, the additional penalties are

$$R_{\mathrm{Prox}} = \frac{\mu}{2} \sum_j (\theta_j - \theta_j^0)^2, \tag{25}$$

$$R_{\mathrm{Feature}} = \frac{\lambda}{Bd} \sum_{i=1}^{B} \|h_i - h_i^0\|^2, \tag{26}$$

$$R_{\mathrm{GKD}} = \frac{\lambda}{B} \sum_i D_{\mathrm{KL}}(p_i^0 \| p_i), \qquad R_{\mathrm{NTD}} = \frac{\lambda}{B} \sum_i D_{\mathrm{KL}}(\pi_i^0 \| \pi_i). \tag{27}$$

Prox includes all trainable parameters, including biases and BN affine parameters, but not the running buffers. The W diagnostic only covers the affine weight blocks. GKD and NTD use temperature one, a frozen current global teacher, and batch-mean KL. Feature L2 uses unnormalized feature vectors and mean squared error, including the factor $1/d$. Its penalty is exactly the minibatch average of the feature distillation term in FedDr+ (Kim et al., 2024), multiplied by λ. Its coefficient is therefore not numerically comparable to that of a sum over coordinates. The complete FedDr+ method instead combines this term with dot regression using weights $1 - \beta$ and β, respectively, and freezes a simplex ETF classifier. Our experiment adds the feature term to cross-entropy and trains the classifier; it tests that component rather than reproducing the complete method. Here β in FedDr+ is an objective weight, unrelated to the Dirichlet partition parameter.

Evaluation and diagnostics. The grid contains five methods, with three coefficients for each of the four penalties, on three datasets and two partitions. Plain has no coefficient sweep. Each setting uses seed 0. CIFAR accuracy averages rounds 451–500 and Tiny accuracy averages rounds 81–100. These windows are fixed and do not select the best test round. Method comparisons are descriptive; round fluctuations and layer samples are not independent training repetitions.

Diagnostics use the same eight training inputs for each compared model, four from each of two selected clients. They cover the stem, convolutional weights in the residual stages and their projection shortcuts, and the classifier, for 21 affine layers in total. We measure the round-start model, the two local models after training, and the aggregate of all ten participating clients. The probes do not control optimization. The CIFAR measurement rounds are 1, 10, 20, 30, 40, 50, 100, 150, 200, 250, 300, 350, 400, 450, and 500. The Tiny schedule is 1, 2, 4, 6, 8, 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100. Late summaries require all of the final three listed rounds.

At each layer and probe, the mean squared change over the eight inputs is divided by the mean reference squared energy, floored at 10^{-8}. We then average these ratios over local models and measurement rounds. The summary across layers also averages the 21 layer ratios; it is not the squared norm of a concatenated network gradient. Convolutional weight gradients sum spatial contributions before taking their squared norm. Native BN uses each model's own running buffers in evaluation mode. Common BN replaces these buffers by the same run's round-start values. This is an evaluation comparison, not a different training algorithm.

Reproducibility and scope. The accompanying tables and plotting files contain the scalar measurements used in the paper. They retain raw values, reference-normalized values, and the method, coefficient, dataset, and partition for each observation. Ten existing runs with matching training settings and the required FL diagnostics were reused. No response-budget controller, gradient-ratio adaptation, or one-step penalty replacement is part of these experiments. The scalar and matrix comparisons in Section 5.1 are mathematical comparisons, not training claims about this grid.

C Additional Measurements and Their Interpretation

Signals and conditional effects. The tables below extend the main figures to every coefficient. At each affine layer, H and E compare features and incoming task errors. The forward effects are $F_h = W^0 \Delta h$ and $F_W = \Delta W h^0$; the backward effects are $B_W = (\Delta W)^\top \delta^0$ and $B_\delta = (W^0)^\top \Delta \delta$. Actual G compares task gradients and includes both conditional effects, their interaction, and the signed cross terms. The displayed numbers are discrepancies of these quantities, rather than the quantities themselves. Different columns have different reference denominators.

Bounds and interactions. For the seven representative CIFAR-100 non-IID settings in the analysis, the actual classifier-gradient norm divided by its feature-plus-error triangle bound is 0.675–0.724. The bound has no positive measured violation, but is not an equality. The backward and forward decompositions also differ from the learning-gradient decomposition: for Plain at the classifier, the feature contribution is 0.933 of the forward component-energy sum, while the error contribution is 0.982 of the backward sum. Plain's signed forward cross ratio is -0.005 on CIFAR-100 and $+0.184$ on CIFAR-10. The negative learning-gradient cross terms in the main figure should therefore not be generalized to every signal.

Client comparisons and aggregation. The head-gradient cosine compares two local models' task gradients on the same inputs. The update cosine compares each of the ten client parameter updates with their mean weighted by each client's number of training minibatches; it is not a pairwise task-gradient cosine. On CIFAR-100, changing GKD from 0.1 to 1 decreases predictive JS from 0.393 to 0.137, while the CE of the selected two-client probability ensemble increases from 0.511 to 0.898. Agreement does not measure task quality. This ensemble also differs from the ten-client aggregate; their discrepancy is not a measurement of aggregation nonlinearity alone. Own-client and other-client CE changes separately describe local fitting on the selected training inputs, not population risk.

Raw and relative scales. For Plain on CIFAR-100, the classifier raw squared gradient change decreases from 216.815 at round 100 to 133.235 late in training, while its relative value increases from 1.326 to 2.368. The reference energy changes. A more extreme example is CIFAR-10 IID Feature L2 with coefficient 1: relative classifier discrepancy 1839.987, raw squared discrepancy 3.873, and accuracy 90.55%. A small reference gradient can create a large relative discrepancy without failed training. Raw and relative values are therefore reported together.

Training duration. The final accuracy windows cannot be replaced by early rankings. On CIFAR-10 non-IID, GKD with coefficient 1 changes from 43.87% over rounds 251–300 to 64.92% over rounds 451–500. FedProx with coefficient 1 changes from 64.14% to 68.35%. NTD with coefficient 0.1 is slightly below FedProx with coefficient 0.1 at the earlier window, but above it at the final window. All main accuracy comparisons retain the same full training horizon.

Table 4: CIFAR-10, $\beta = 0.1$: late mean reference-normalized squared changes over 21 affine layers. Different columns have different reference denominators.

Method	H	E	F_h	F_W	B_W	B_δ	G
Feature L2 0.01	0.275	2.208	0.271	0.016	0.008	2.146	2.572
Feature L2 0.1	0.282	2.540	0.265	0.016	0.010	2.479	3.274
Feature L2 1	0.279	1.868	0.262	0.016	0.010	1.839	2.246
GKD 0.01	0.304	1.995	0.293	0.018	0.011	1.912	2.234
GKD 0.1	0.368	1.482	0.330	0.022	0.012	1.403	1.650
GKD 1	0.272	0.984	0.230	0.015	0.008	0.873	0.909
NTD 0.01	0.299	1.784	0.301	0.016	0.008	1.735	1.983
NTD 0.1	0.251	1.712	0.242	0.014	0.007	1.629	2.092
NTD 1	0.139	1.922	0.116	0.008	0.004	1.814	2.341
Plain	0.298	2.067	0.299	0.016	0.008	1.946	2.648
FedProx 0.01	0.290	2.093	0.288	0.016	0.008	2.032	2.408
FedProx 0.1	0.268	2.020	0.286	0.014	0.006	1.956	2.178
FedProx 1	0.080	1.734	0.105	0.040	0.003	1.720	1.843

Table 5: CIFAR-100, $\beta = 0.1$: late mean reference-normalized squared changes over 21 affine layers. Different columns have different reference denominators.

Method	H	E	F_h	F_W	B_W	B_δ	G
Feature L2 0.01	0.135	2.744	0.161	0.012	0.005	2.779	3.087
Feature L2 0.1	0.135	1.999	0.163	0.012	0.005	2.022	2.247
Feature L2 1	0.097	2.024	0.102	0.010	0.004	2.021	2.323
GKD 0.01	0.134	1.877	0.161	0.012	0.006	1.870	2.164
GKD 0.1	0.113	1.372	0.114	0.009	0.004	1.325	1.494
GKD 1	0.049	0.554	0.036	0.005	0.002	0.496	0.446
NTD 0.01	0.134	3.295	0.156	0.012	0.005	3.392	4.040
NTD 0.1	0.109	1.954	0.115	0.010	0.004	1.940	2.306
NTD 1	0.052	1.173	0.043	0.006	0.003	1.151	1.128
Plain	0.136	3.425	0.159	0.012	0.006	3.508	3.983
FedProx 0.01	0.137	2.379	0.171	0.012	0.005	2.379	2.684
FedProx 0.1	0.124	2.135	0.151	0.011	0.005	2.126	2.341
FedProx 1	0.049	0.942	0.063	0.004	0.001	0.916	0.879

Table 6: Tiny ImageNet, $\beta = 0.1$: late mean reference-normalized squared changes over 21 affine layers. Different columns have different reference denominators.

Method	H	E	F_h	F_W	B_W	B_δ	G
Feature L2 0.01	0.112	1.295	0.103	0.035	0.011	1.238	1.265
Feature L2 0.1	0.110	1.477	0.101	0.035	0.011	1.417	1.430
Feature L2 1	0.089	1.194	0.070	0.031	0.010	1.117	1.125
GKD 0.01	0.112	1.162	0.101	0.034	0.011	1.102	1.123
GKD 0.1	0.097	0.979	0.084	0.026	0.009	0.914	0.910
GKD 1	0.062	0.567	0.043	0.013	0.006	0.496	0.420
NTD 0.01	0.110	1.530	0.103	0.035	0.011	1.482	1.595
NTD 0.1	0.101	1.103	0.086	0.025	0.010	1.046	1.004
NTD 1	0.078	0.760	0.055	0.017	0.009	0.693	0.641
Plain	0.114	1.451	0.105	0.036	0.011	1.406	1.455
FedProx 0.01	0.109	1.456	0.102	0.034	0.010	1.413	1.412
FedProx 0.1	0.090	1.083	0.090	0.024	0.006	1.029	0.996
FedProx 1	0.080	0.886	0.089	0.015	0.001	0.856	0.731

Table 7: CIFAR-100, $\beta = 0.1$: prediction changes in nats. Full KL equals Mass plus Weighted NT; NT is the unweighted conditional KL.

Method	Full KL	NT	Weighted NT	Mass
Feature L2 0.01	2.796	1.896	0.960	1.836
Feature L2 0.1	3.227	2.195	1.263	1.964
Feature L2 1	2.429	1.415	0.782	1.648
GKD 0.01	2.755	1.805	0.960	1.795
GKD 0.1	1.417	0.918	0.516	0.901
GKD 1	0.284	0.134	0.099	0.185
NTD 0.01	2.520	1.976	0.884	1.636
NTD 0.1	2.149	1.167	0.621	1.528
NTD 1	1.162	0.318	0.135	1.027
Plain	2.818	2.011	0.940	1.878
FedProx 0.01	2.960	1.967	0.977	1.983
FedProx 0.1	2.902	1.814	0.956	1.947
FedProx 1	1.440	0.820	0.532	0.908

Table 8: CIFAR-100, IID: prediction changes in nats. Full KL equals Mass plus Weighted NT; NT is the unweighted conditional KL.

Method	Full KL	NT	Weighted NT	Mass
Feature L2 0.01	1.526	1.420	0.531	0.995
Feature L2 0.1	1.088	1.234	0.313	0.775
Feature L2 1	0.880	0.679	0.182	0.698
GKD 0.01	0.915	1.185	0.274	0.641
GKD 0.1	0.719	0.872	0.225	0.493
GKD 1	0.210	0.166	0.066	0.144
NTD 0.01	0.780	0.924	0.269	0.511
NTD 0.1	0.708	0.520	0.130	0.578
NTD 1	0.503	0.217	0.053	0.450
Plain	1.347	1.364	0.451	0.896
FedProx 0.01	1.099	1.297	0.294	0.806
FedProx 0.1	0.997	1.200	0.313	0.684
FedProx 1	0.975	0.758	0.311	0.664

Table 9: Forward response Wh, excluding bias, at the CIFAR-100 classifier, $\beta = 0.1$. Components use fixed references; ratios are calculated per observation before averaging. S is the sum of the first three energies.

Method	First/S	Second/S	Interaction/S	Cross/S	Actual/S
Feature L2 0.01	0.936	0.063	0.001	-0.002	0.998
Feature L2 0.1	0.940	0.059	0.001	-0.009	0.991
Feature L2 1	0.918	0.082	0.001	-0.015	0.985
GKD 0.01	0.937	0.062	0.001	-0.018	0.982
GKD 0.1	0.915	0.083	0.001	-0.045	0.955
GKD 1	0.827	0.171	0.003	-0.208	0.792
NTD 0.01	0.936	0.063	0.001	-0.016	0.984
NTD 0.1	0.932	0.067	0.001	-0.007	0.993
NTD 1	0.882	0.117	0.001	-0.181	0.819
Plain	0.933	0.066	0.001	-0.005	0.995
FedProx 0.01	0.943	0.056	0.001	-0.015	0.985
FedProx 0.1	0.935	0.064	0.001	-0.016	0.984
FedProx 1	0.869	0.131	0.000	-0.120	0.880

Table 10: Backward signal at the CIFAR-100 classifier, $\beta = 0.1$. Components use fixed references; ratios are calculated per observation before averaging. S is the sum of the first three energies.

Method	First/S	Second/S	Interaction/S	Cross/S	Actual/S
Feature L2 0.01	0.006	0.983	0.011	0.004	1.004
Feature L2 0.1	0.009	0.978	0.013	-0.002	0.998
Feature L2 1	0.006	0.983	0.010	0.006	1.006
GKD 0.01	0.009	0.978	0.013	-0.006	0.994
GKD 0.1	0.010	0.977	0.013	-0.006	0.994
GKD 1	0.050	0.937	0.012	-0.067	0.933
NTD 0.01	0.006	0.984	0.010	0.008	1.008
NTD 0.1	0.006	0.983	0.011	0.007	1.007
NTD 1	0.010	0.979	0.011	-0.009	0.991
Plain	0.007	0.982	0.011	0.002	1.002
FedProx 0.01	0.007	0.983	0.010	-0.002	0.998
FedProx 0.1	0.007	0.982	0.011	-0.002	0.998
FedProx 1	0.007	0.987	0.006	-0.011	0.989

Table 11: Weight gradient at the CIFAR-100 classifier, $\beta = 0.1$. Components use fixed references; ratios are calculated per observation before averaging. S is the sum of the first three energies.

Method	First/S	Second/S	Interaction/S	Cross/S	Actual/S
Feature L2 0.01	0.192	0.551	0.257	-0.223	0.777
Feature L2 0.1	0.224	0.530	0.246	-0.303	0.697
Feature L2 1	0.164	0.652	0.183	-0.196	0.804
GKD 0.01	0.227	0.542	0.231	-0.269	0.731
GKD 0.1	0.226	0.585	0.189	-0.246	0.754
GKD 1	0.372	0.561	0.068	-0.122	0.878
NTD 0.01	0.183	0.565	0.253	-0.151	0.849
NTD 0.1	0.200	0.582	0.218	-0.241	0.759
NTD 1	0.178	0.697	0.126	-0.189	0.811
Plain	0.175	0.569	0.256	-0.234	0.766
FedProx 0.01	0.211	0.529	0.260	-0.291	0.709
FedProx 0.1	0.187	0.573	0.240	-0.225	0.775
FedProx 1	0.182	0.706	0.112	-0.183	0.817

Table 12: CIFAR-100, $\beta = 0.1$: actual gradient changes across model and BN conditions, averaged over 21 layers.

Method	Local native	Local common	Aggregate native	Aggregate common
Feature L2 0.01	3.087	10.494	0.380	0.597
Feature L2 0.1	2.247	7.448	0.480	0.655
Feature L2 1	2.323	7.530	0.298	0.460
GKD 0.01	2.164	6.462	0.489	0.819
GKD 0.1	1.494	4.824	0.300	0.597
GKD 1	0.446	1.272	0.077	0.133
NTD 0.01	4.040	9.464	0.469	0.726
NTD 0.1	2.306	11.121	0.299	0.530
NTD 1	1.128	2.710	0.110	0.167
Plain	3.983	10.734	0.722	1.601
FedProx 0.01	2.684	6.988	0.451	0.718
FedProx 0.1	2.341	5.025	0.383	0.538
FedProx 1	0.879	1.213	0.120	0.172

Table 13: CIFAR-100, $\beta = 0.1$: client agreement and fitting. Update cosine uses Stage 4 and the participating clients; head cosine uses two local models on the same inputs. Own/other columns are CE changes from the round-start model.

Method	Update cos.	Head cos.	Predictive JS	Own CE	Other CE
Feature L2 0.01	0.365	0.442	0.494	-0.788	2.605
Feature L2 0.1	0.361	0.284	0.504	-1.074	2.465
Feature L2 1	0.349	0.386	0.490	-0.890	2.127
GKD 0.01	0.360	0.364	0.489	-0.930	2.655
GKD 0.1	0.346	0.515	0.393	-0.861	1.270
GKD 1	0.325	0.862	0.137	-1.011	0.346
NTD 0.01	0.361	0.397	0.475	-0.678	2.546
NTD 0.1	0.347	0.306	0.391	-0.767	1.614
NTD 1	0.321	0.465	0.295	-0.811	0.510
Plain	0.362	0.288	0.512	-0.743	2.723
FedProx 0.01	0.364	0.383	0.476	-0.872	2.662
FedProx 0.1	0.358	0.344	0.493	-0.937	2.619
FedProx 1	0.335	0.633	0.363	-1.483	0.809

Table 14: CIFAR-10, IID: classifier changes. Relative values can be large when reference gradient energy is small. Raw G is the mean squared task-gradient change, without reference normalization.

Method	Relative H	Raw H	Relative G	Raw G
Feature L2 0.01	0.090	12.368	4092.520	4.716
Feature L2 0.1	0.068	9.169	169.558	4.918
Feature L2 1	0.045	8.910	1839.987	3.873
GKD 0.01	0.070	10.338	113.656	5.683
GKD 0.1	0.078	10.952	168.678	3.972
GKD 1	0.037	3.937	0.597	1.552
NTD 0.01	0.079	10.950	418.945	1.464
NTD 0.1	0.044	5.952	2.007	3.329
NTD 1	0.047	5.178	4.632	2.854
Plain	0.075	10.145	70.863	2.413
FedProx 0.01	0.072	10.272	12.430	3.449
FedProx 0.1	0.102	14.225	405.224	7.358
FedProx 1	0.050	10.865	133.929	13.261