SCALE-CALIBRATED PRECONDITIONING FOR LAYER-WISE CLIPPING UNDER DIFFERENTIAL PRIVACY

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances have improved the efficiency and utility of differentially private training for large vision and language models. Per-layer gradient clipping is an important approach to reducing memory overhead. However, clipping each layer independently with its own threshold can produce different scaling factors across layers, distorting the full-gradient direction and degrading utility. We propose SCP-Clip, which performs per-layer clipping in a scale-calibrated preconditioned space to jointly mitigate directional distortion caused by per-layer clipping. The method incurs no additional privacy cost while retaining the efficiency advantages of per-layer clipping. Theoretically, we establish a utility bound on the private gradient estimation error under stable feedback. Experiments on pretrained language and vision models show SCP-Clip outperforms flat clipping in most settings, with accuracy gains of up to 4.28% over adaptive per-layer clipping.

1 INTRODUCTION

Differential privacy (DP) is a security measure that formalizes and quantifies information leakage (Dwork & Roth, 2014). In deep learning, differentially private stochastic gradient descent (DP-SGD) (Abadi et al., 2016) is the most widely used DP method. It provides rigorous example-level privacy through per-example gradient clipping and Gaussian noise injection, at the cost of additional computation and substantial utility degradation. To make DP training feasible for large neural networks, prior work has reduced the computational or memory overhead of DP-SGD through partial-parameter fine-tuning (Yu et al., 2022; Bu et al., 2024; Hong et al., 2025) and implementation-level optimizations (Lee & Kifer, 2021; Subramani et al., 2021; Li et al., 2022b; Bu et al., 2023c).

Although these methods improve the scalability of DP training, gradient clipping in standard DP-SGD still incurs substantial memory overhead. Group clipping offers an important route to efficient DP-SGD implementations, with per-layer clipping as its most common form. Per-layer clipping integrates with backpropagation to clip and aggregate per-example gradients layer by layer and release intermediate results (He et al., 2023). Recent work has brought the memory and computational costs of per-layer DP training for large models substantially closer to those of non-private training. These implementations also accommodate model parallelism and large-scale architectures such as Transformers more naturally (Wang et al., 2025).

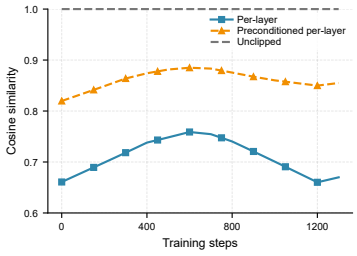


Figure 1: Preconditioning mitigates the gradient direction distortion caused by per-layer clipping.

However, fine-grained clipping typically trades some utility for lower memory overhead (Bu et al., 2023a). He et al. (2023) adaptively adjust each layer's clipping threshold to better accommodate changes in layer gradients. This approach improves the utility of training with per-layer clipping. Nevertheless, flat clipping scales the full gradient by a single coefficient, whereas per-layer clipping applies different scaling coefficients to different layers. This changes the original gradient direction and introduces additional clipping bias (Figure 1).

To address these limitations, we introduce SCP-Clip, a scale-calibrated preconditioning method for per-layer clipping. At its core, SCP-Clip performs per-layer clipping in a preconditioned space constructed by rescaling gradient coordinates using inverse square roots of the optimizer's second-moment estimates from previously privatized gradients. This coordinate-wise transformation is fur-

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.

ther calibrated by a layer-wise scalar updated from private gradient norm feedback, adapting to evolving gradient scales while keeping clipping thresholds fixed. Gradient clipping and noise addition occur in this space, the result of which, i.e., the noisy aggregate, is mapped back to the original space. This design mitigates clipping-induced directional distortion without additional privacy cost while preserving layer-by-layer execution during backpropagation. Our contributions are threefold:

1. We propose preconditioned per-layer gradient clipping to mitigate the additional clipping bias introduced by layer-specific clipping coefficients.
2. We introduce layer-wise scale calibration of the preconditioner, enabling clipping to adapt to changes in gradient scale while keeping clipping thresholds fixed.
3. We evaluate SCP-Clip across multiple pretrained models on sentence classification, image classification, and table-to-text generation, demonstrating substantial utility improvements over existing per-layer clipping methods.

2 PRELIMINARIES

Definition 1 ((ϵ, δ)-DP) *A randomized algorithm $\mathcal{M}$ is (ϵ, δ)-differentially private if, for all neighboring datasets $\mathbb{D}, \mathbb{D}'$ differing in only one element and every possible output subset $\mathbb{O}$,*

$$\Pr(\mathcal{M}(\mathbb{D}) \in \mathbb{O}) \leq e^{\epsilon} \Pr(\mathcal{M}(\mathbb{D}') \in \mathbb{O}) + \delta.$$

Gradient-based differentially private training typically clips per-example gradients to bound individual contributions and adds Gaussian noise to their aggregate. Flat and group clipping differ in the granularity of parameter grouping.

Flat clipping. For model parameters $\boldsymbol{\theta} \in \mathbb{R}^d$ and an example x_i, let $\boldsymbol{g}^{(i)} = \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; x_i) \in \mathbb{R}^d$ denote the loss gradient. During backpropagation, the gradient norms of individual modules are computed in reverse order and combined to obtain the full-gradient norm $\|\boldsymbol{g}^{(i)}\|_2$. Once backpropagation is complete, we use this norm and a single threshold $C > 0$ to clip the full gradient:

$$\overline{\boldsymbol{g}}^{(i)} = \mathrm{Clip}\Big(\boldsymbol{g}^{(i)}, C\Big) = \alpha^{(i)} \boldsymbol{g}^{(i)}, \qquad \alpha^{(i)} = \min\left\{1, C/\|\boldsymbol{g}^{(i)}\|_2\right\}.$$

Thus, flat clipping uses the same scaling coefficient for all parameters of a given example.

Group clipping. Partition the parameters into L disjoint clipping groups with d_l parameters in group l. Then

$$\boldsymbol{\theta} = [\boldsymbol{\theta}_1; \ldots; \boldsymbol{\theta}_L], \qquad \boldsymbol{\theta}_l \in \mathbb{R}^{d_l}, \qquad d = \sum_{l=1}^{L} d_l.$$

For example x_i, let $\boldsymbol{g}_l^{(i)} = \nabla_{\boldsymbol{\theta}_l} \ell(\boldsymbol{\theta}; x_i) \in \mathbb{R}^{d_l}$ denote the loss gradient with respect to parameter group l. Once group l completes its backward computations, we immediately clip and aggregate its gradients and release intermediate states before continuing backpropagation:

$$\overline{\boldsymbol{g}}_l = \sum_{i \in \mathbb{B}} \mathrm{Clip}\Big(\boldsymbol{g}_l^{(i)}, C_l\Big) = \sum_{i \in \mathbb{B}} \alpha_l^{(i)} \boldsymbol{g}_l^{(i)}, \qquad \alpha_l^{(i)} = \min\left\{1, C_l/\|\boldsymbol{g}_l^{(i)}\|_2\right\}.$$

In group clipping, each parameter group independently determines its scaling coefficient from its own gradient norm and clipping threshold. This allows clipping, aggregation, and noise injection to be integrated directly into backpropagation without waiting for the full gradient to be computed. The execution order and intermediate-state lifetimes therefore more closely resemble those of non-private training. In particular, per-layer clipping is the grouping scheme that most naturally matches the backpropagation structure of neural networks.

Gradient preconditioning. Gradient preconditioning encompasses methods that improve the geometry of optimization problems. The parameter update can be written as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \boldsymbol{P}_t \boldsymbol{g},$$

where $\boldsymbol{P}_t \in \mathbb{R}^{d \times d}$ is a preconditioning matrix typically based on gradient second-moment estimates or local curvature information. Preconditioning typically attenuates updates along high-curvature or large-scale directions while relatively amplifying those along low-curvature or small-scale directions. This mitigates scale imbalances across directions and adapts to gradient geometry (Duchi et al., 2011; Kingma & Ba, 2017).

3 SCALE-CALIBRATED PRECONDITIONING FOR PER-LAYER CLIPPING

We introduce SCP-Clip, a layer-wise clipping method. We first examine the additional clipping bias introduced by per-layer clipping and apply gradient preconditioning to mitigate this bias in Section 3.1. We then incorporate scale calibration into preconditioning to match gradient distributions in Section 3.2.

3.1 REDUCING CLIPPING BIAS VIA PRECONDITIONING

Per-layer clipping bias. Per-layer clipping determines a separate clipping coefficient for each layer. For an individual example x_i, the gradient with per-layer clipping is

$$\overline{\boldsymbol{g}}^{(i)} = [\alpha_1^{(i)}\boldsymbol{g}_1^{(i)}; \ldots; \alpha_L^{(i)}\boldsymbol{g}_L^{(i)}],$$

where the clipping coefficient for layer l is $\alpha_l^{(i)} = \min\{1, C_l/\|\boldsymbol{g}_l^{(i)}\|_2\}$. Per-layer clipping introduces biases that deviate from the original batch gradient. Even under flat clipping, where the clipping threshold C_l is uniform across layers, the clipped gradient fails to preserve the original gradient direction. During mini-batch aggregation, clipping scales down a subset of per-example gradients while preserving others. This disparate treatment induces a clipping bias relative to the original batch gradient (Chen et al., 2020; Xiao et al., 2023). In per-layer clipping, such bias is further amplified as each layer employs distinct clipping coefficients, resulting in heterogeneous scaling across the full gradient. Consequently, the additional directional distortion introduced by per-layer clipping leads to greater utility degradation.

The clipping coefficient $\alpha_l^{(i)}$ depends jointly on the clipping threshold C_l and the gradient norm $\|\boldsymbol{g}_l^{(i)}\|_2$. The threshold C_l is shared across all examples, whereas $\|\boldsymbol{g}_l^{(i)}\|_2$ varies from example to example. Existing per-layer clipping methods determine clipping coefficients separately within each layer and do not account for differences in the inter-layer structure of individual example gradients. For instance, how far the norm of the clipped example gradients exceeds/falls short of the clipping threshold may vary significantly across examples. Hence per-layer clipping severely deviates from the original optimization direction with each layer clipped individually. Therefore, before computing each layer's clipping coefficient, we consider recalibrating coordinate scales using historical gradient second moments. This recalibration would adjust clipping behavior on the basis of the aggregated batch gradients.

Preconditioning. Let $\mathbb{B}_t$ denote the batch at step t, and $\boldsymbol{g}_l^{(i)}$ be the gradient of example x_i at layer l. We reuse the second-moment estimate $\hat{\boldsymbol{v}}_{l,t-1} \in [0,\infty)^{d_l}$ maintained by the optimizer from previously privatized batch gradients. We define the base diagonal preconditioner $\mathbf{D}_{l,t} \in \mathbb{R}^{d_l \times d_l}$ and the complete preconditioner $\boldsymbol{P}_{l,t}$ as

$$\mathbf{D}_{l,t} = \operatorname{diag}\Big(1/(\sqrt{\hat{\boldsymbol{v}}_{l,t-1}} + \epsilon_{\text{num}})\Big), \qquad \boldsymbol{P}_{l,t} = \rho_{l,t}\mathbf{D}_{l,t}, \qquad \rho_{l,t} > 0.$$

Here, ϵ_{num} is a numerical stabilizer. Vector square roots and divisions in the moment and preconditioner formulas are applied element-wise. The layer-wise scalar $\rho_{l,t}$ is maintained using previously released private feedback. Both $\mathbf{D}_{l,t}$ and $\rho_{l,t}$ remain fixed throughout the current iteration, and all examples undergo the transformation:

$$\hat{\boldsymbol{g}}_l^{(i)} = \boldsymbol{P}_{l,t}\boldsymbol{g}_l^{(i)} = \rho_{l,t}\mathbf{D}_{l,t}\boldsymbol{g}_l^{(i)}.$$

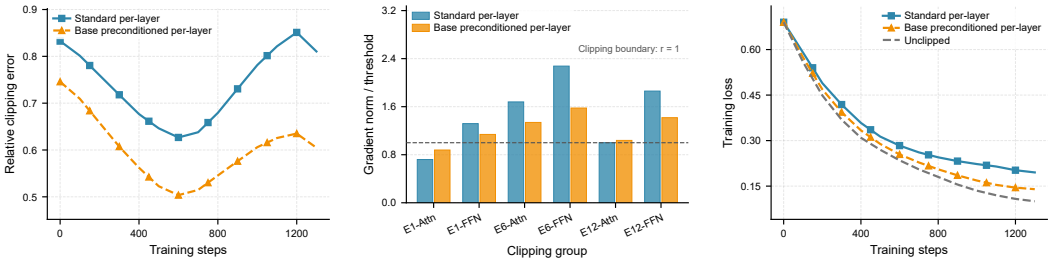
Gradient privatization in the preconditioned space. In the transformed space, we clip per-example gradients using the threshold C_l for that layer:

$$\overline{\boldsymbol{g}}_l^{(i)} = \operatorname{Clip}\Big(\hat{\boldsymbol{g}}_l^{(i)}, C_l\Big).$$

Or equivalently, the clipping coefficient in the preconditioned space is

$$\alpha_{l,t}^{(i)} = \min\Big\{1,\, C_l/\Big(\rho_{l,t}\|\mathbf{D}_{l,t}\boldsymbol{g}_l^{(i)}\|_2\Big)\Big\}.$$

Without causing confusion, we refer to the special case where $\rho_{l,t} = 1$ for all l and t as **base preconditioning**, and we will discuss the more complicated case in Section 3.2.



(a) Relative clipping error. (b) Norm-to-threshold ratio. (c) Training loss.

Figure 2: **Clipping in the preconditioned space reduces the clipping bias introduced by per-layer clipping.** (a) Relative clipping error $\|\boldsymbol{g}_c - \boldsymbol{g}\|_2 / \|\boldsymbol{g}\|_2$, where $\boldsymbol{g}$ is the unclipped gradient and $\boldsymbol{g}_c$ is the clipped gradient. (b) Mean ratio of the per-example gradient norm to the threshold in six representative clipping groups. The preconditioned method computes this ratio using $\|\mathbf{D}_{l,t}\boldsymbol{g}_l^{(i)}\|_2 / C_l$. The dashed line $r = 1$ marks the boundary at which an individual example begins to be clipped. (c) Training loss without privacy noise.

We then aggregate the clipped gradients and add Gaussian noise. Applying the inverse transformation yields the noisy aggregated gradient in the original space:

$$\tilde{\boldsymbol{g}}_l = \frac{1}{B}\boldsymbol{P}_{l,t}^{-1}\left(\sum_{i\in\mathbb{B}_t}\bar{\boldsymbol{g}}_l^{(i)} + \mathcal{N}\left(\mathbf{0}, \sigma^2 C^2 \boldsymbol{I}_{d_l}\right)\right).$$

Here, σ is the noise multiplier, and $C = \sqrt{\sum_{l=1}^{L} C_l^2}$ is the equal of the global clipping threshold. The resulting $\tilde{\boldsymbol{g}}_l$ is passed to the optimizer to update its state and the model parameters.

Evidence of bias reduction for base preconditioning. By changing layer gradient norms, preconditioning changes the coefficients that determine the clipping value in each layer. Constructed from historical second moments, $\mathbf{D}_{l,t}$ contracts coordinates with persistently large scales more strongly. The resulting norm $\|\mathbf{D}_{l,t}\boldsymbol{g}_l\|_2$ better reflects the current gradient's magnitude relative to its historical scale. When historical scale differences dominate the differences in layer gradient norms, preconditioning can make layer scales more comparable relative to their clipping thresholds. This reduces the dispersion of clipping coefficients across layers and, consequently, the deviation from the original gradient direction.

Figure 1 illustrates differences in directional alignment between the clipping methods. To examine this mechanism further, we compare standard and base preconditioned per-layer clipping. The detailed setting is given in Appendix F. Figure 2(a) shows that preconditioned per-layer clipping maintains a lower relative clipping error during training. Its clipped gradients are therefore closer overall to the unclipped gradients. Figure 2(b) shows each group's position relative to the clipping boundary. Across the six representative clipping groups, the base preconditioner produces gradients of norm more concentrated to the clipping threshold. This alleviates scale imbalances between groups (layers) relative to their clipping thresholds. Figure 2(c) shows that preconditioned clipping achieves lower training loss in the middle and later stages of training, which indicates a smaller training error and better convergence performance.

3.2 Preconditioner Scaling via Gradient Distribution

Although historical second moments can calibrate gradient coordinate scales, the gradient norms after base preconditioning $\|\mathbf{D}_{l,t}\boldsymbol{g}_l^{(i)}\|_2$ still vary for each example. Hence the per-example gradient would still have its direction mismatched to the unclipped gradient. We are motivated to design an adaptive clipping by leveraging the per-example gradient norm statistics. Such statistics may include the fraction of unclipped examples given the clipping threshold in the previous iteration (He et al., 2023).

Specifically, we incorporate the adaptive threshold adjustment in the preconditioner. We keep the per-layer clipping thresholds C_l fixed throughout training. Private norm feedback cumulatively updates the layer-wise scalar $\rho_{l,t}$ in the preconditioner $\boldsymbol{P}_{l,t} = \rho_{l,t}\mathbf{D}_{l,t}$. The preconditioning trans-

Algorithm 1 Preconditioned adaptive per-layer clipping

Input: dataset $\mathbb{D}$; initial parameters $\boldsymbol{\theta}_0$; iterations T; normalization constant B; clipping thresholds $\{C_l\}_{l=1}^L$; initial preconditioners $\{\boldsymbol{P}_{l,1}\}_{l=1}^L$; initial scales $\{\rho_{l,1}\}_{l=1}^L$; noise multiplier σ; slack dimension K; scale step size η_ρ; Adam parameters $\mathcal{H} = (\eta_1, \beta_1, \beta_2, \epsilon_{\text{num}})$.
Initialize $\mathcal{S}_0 = (\boldsymbol{\mu}_0, \boldsymbol{v}_0) \leftarrow (\mathbf{0}, \mathbf{0})$.
for $t = 1$ to T **do**
  Sample $\mathbb{B}_t$ and perform the forward pass at $\boldsymbol{\theta}_{t-1}$.
  for $l = L$ to 1 **do**
    Compute $\{\boldsymbol{g}_l^{(i)}\}_{i\in\mathbb{B}_t}$ from activations and output gradients.
    Transform $\hat{\boldsymbol{g}}_l^{(i)} \leftarrow \boldsymbol{P}_{l,t}\boldsymbol{g}_l^{(i)}$.
    Clip $\overline{\boldsymbol{g}}_l^{(i)} \leftarrow \hat{\boldsymbol{g}}_l^{(i)} \min\{1, C_l/\|\hat{\boldsymbol{g}}_l^{(i)}\|_2\}$.
    Set $\lambda_l \leftarrow C_l/\sqrt{K}$ and encode $\boldsymbol{s}_l^{(i)}$ using equation 1.
    Release $\left[\tilde{\boldsymbol{h}}_l,\ \tilde{\boldsymbol{s}}_l\right] \leftarrow \frac{1}{B}\left(\sum_{i\in\mathbb{B}_t}\left[\overline{\boldsymbol{g}}_l^{(i)},\ \boldsymbol{s}_l^{(i)}\right] + \mathcal{N}(\mathbf{0}, \sigma^2 C^2 \boldsymbol{I}_{d_l+K})\right)$ and $C = \sqrt{\sum_{l=1}^L C_l^2}$.
    Map back $\tilde{\boldsymbol{g}}_l \leftarrow \boldsymbol{P}_{l,t}^{-1}\tilde{\boldsymbol{h}}_l$.
  end for
  $(\boldsymbol{\theta}_t, \mathcal{S}_t) \leftarrow \text{AdamStep}(\boldsymbol{\theta}_{t-1}, \mathcal{S}_{t-1}, \tilde{\boldsymbol{g}}, t, \mathcal{H})$.
  $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t/(1-\beta_2^t)$, $\quad \mathbf{D}_{l,t+1} \leftarrow \text{diag}\big(1/(\sqrt{\hat{\boldsymbol{v}}_{l,t}} + \epsilon_{\text{num}})\big)$ for all l.
  for $l = 1$ to L **do**
    Normalize $\hat{\boldsymbol{s}}_l \leftarrow \tilde{\boldsymbol{s}}_l/\lambda_l$ and compute $\gamma_{l,t} \leftarrow \Pi_{[0,1]}(1 - (1 - \hat{s}_{l,K}/C_l)/2)$.
    Update $\rho_{l,t+1} \leftarrow \rho_{l,t}\exp(\eta_\rho(\hat{s}_{l,1} - \gamma_{l,t}))$ and $\boldsymbol{P}_{l,t+1} \leftarrow \rho_{l,t+1}\mathbf{D}_{l,t+1}$.
  end for
end for
return $\boldsymbol{\theta}_T$.

formation thus combines coordinate-scale calibration based on historical second moments with feedback-driven layer-wise scale adjustment.

Private estimation of gradient norm distributions. We obtain private feedback using the slack encoding mechanism of Zou et al. (2026). For each example and layer in the current iteration, we compute the nonnegative slack between the preconditioned gradient norm and the threshold, $\max\Big\{C_l - \|\rho_{l,t}\mathbf{D}_{l,t}\boldsymbol{g}_l^{(i)}\|_2,\ 0\Big\}$. We encode this slack in a K-dimensional vector of the form

$$\boldsymbol{s}_l^{(i)} \leftarrow [\lambda_l \mathbf{1}_a,\ b,\ \mathbf{0}]^{(i)}. \tag{1}$$

Here, $\lambda_l = C_l/\sqrt{K}$ is the upper bound on each coordinate. The parameters $a \in \mathbb{N}$ and $b \in [0, \lambda_l)$ are uniquely determined by $\sqrt{K}\max\Big\{C_l - \|\rho_{l,t}\mathbf{D}_{l,t}\boldsymbol{g}_l^{(i)}\|_2,\ 0\Big\} = a\lambda_l + b$. Here, the vector $\mathbf{1}_a$ denotes the a-dimensional vector of all ones. For example, with $K = 4$, a scaled slack of $2.5\lambda_l$ gives $\boldsymbol{s}_l^{(i)} = [\lambda_l, \lambda_l, 0.5\lambda_l, 0]$. The aggregated and normalized private vector $\hat{\boldsymbol{s}}_l$ approximately characterizes the empirical cumulative distribution of the preconditioned gradient norms. Its first coordinate $\hat{s}_{l,1}$ approximates the fraction of unclipped examples. We jointly privatize and release the encoded slack with the gradients, incurring no additional privacy cost. A detailed privacy analysis of the mechanism is provided in Section 4.1.

Updating layer-wise preconditioner scales. The preconditioner for the current iteration is determined by historical private outputs and remains fixed throughout that iteration. After the current joint release, private feedback updates the layer-wise scale for the next iteration:

$$\rho_{l,t+1} = \rho_{l,t}\exp\left[\eta_\rho\big(\hat{s}_{l,1} - \gamma_{l,t}\big)\right], \qquad \boldsymbol{P}_{l,t+1} = \rho_{l,t+1}\mathbf{D}_{l,t+1},$$

where $\eta_\rho > 0$ is the step size for the scale update. The target fraction of unclipped examples $\gamma_{l,t}$ is computed according to Algorithm 1. For a given base preconditioner, when the feedback falls below the target fraction, the update decreases $\rho_{l,t}$. This reduces the transformed gradient norms and thereby alleviates clipping. When feedback exceeds the target fraction, the update increases $\rho_{l,t}$, making clipping more restrictive. Each update acts on the existing scale, preserving the state accumulated from previous feedback.

Effective clipping scale adaptation. Although the clipping thresholds remain fixed in the transformed space, the layer-wise scalar adjusts the clipping region in the original gradient space:

$$\{\boldsymbol{g}_l : \|\rho_{l,t}\mathbf{D}_{l,t}\boldsymbol{g}_l\|_2 \le C_l\} = \{\boldsymbol{g}_l : \|\mathbf{D}_{l,t}\boldsymbol{g}_l\|_2 \le C_l/\rho_{l,t}\}\,.$$

Thus, relative to the base preconditioner, the layer-wise scalar adjusts the effective clipping scale through $C_l/\rho_{l,t}$. For a given $\mathbf{D}_{l,t}$, this adjustment changes the size of the clipping ellipsoid while preserving the ratios of its axis lengths.

4 Analysis

4.1 Privacy Analysis

At each iteration, Algorithm 1 jointly releases the private average of the extended gradients, with layer-wise outputs $[\tilde{\boldsymbol{h}}_l,\ \tilde{\boldsymbol{s}}_l]$. We show that preconditioned adaptive per-layer clipping requires no additional privacy budget beyond standard DP-SGD.

Theorem 4.1 (Privacy guarantee of Algorithm 1) *Assume fixed per-layer clipping thresholds $\{C_l\}_{l=1}^L$ and normalize each release by a public fixed constant $B > 0$. Let the equivalent global clipping threshold be $C = \sqrt{\sum_{l=1}^L C_l^2}$. With the same sampling rule and rate, noise multiplier, and privacy accountant, Algorithm 1 admits the same per-step privacy upper bound as standard DP-SGD with clipping threshold C. (The full proof and supporting lemmas are in Appendix D.)*

Proof sketch. Define the extended gradient for each example and layer as $\boldsymbol{g}_l^{+(i)} = \big[\overline{\boldsymbol{g}}_l^{(i)},\ \boldsymbol{s}_l^{(i)}\big]$. If $\|\boldsymbol{P}_{l,t}\boldsymbol{g}_l^{(i)}\|_2 \ge C_l$, the slack vector is $\boldsymbol{s}_l^{(i)} = \mathbf{0}$, and the clipped gradient has norm C_l. Hence, $\|\boldsymbol{g}_l^{+(i)}\|_2 = C_l$ in this case. If $\|\boldsymbol{P}_{l,t}\boldsymbol{g}_l^{(i)}\|_2 < C_l$, clipping leaves the preconditioned gradient unchanged. The slack encoding, with $\lambda_l = C_l/\sqrt{K}$, satisfies

$$\|\boldsymbol{s}_l^{(i)}\|_2^2 \le \lambda_l\|\boldsymbol{s}_l^{(i)}\|_1 = C_l\big(C_l - \|\boldsymbol{P}_{l,t}\boldsymbol{g}_l^{(i)}\|_2\big).$$

It follows that

$$\|\boldsymbol{g}_l^{+(i)}\|_2^2 = \|\boldsymbol{P}_{l,t}\boldsymbol{g}_l^{(i)}\|_2^2 + \|\boldsymbol{s}_l^{(i)}\|_2^2 \le \|\boldsymbol{P}_{l,t}\boldsymbol{g}_l^{(i)}\|_2^2 + C_l\big(C_l - \|\boldsymbol{P}_{l,t}\boldsymbol{g}_l^{(i)}\|_2\big) \le C_l^2.$$

Thus, $\|\boldsymbol{g}_l^{+(i)}\|_2 \le C_l$ holds in both cases. Concatenating the extended gradients across all layers gives the full extended gradient $\boldsymbol{g}^{+(i)}$, which satisfies

$$\|\boldsymbol{g}^{+(i)}\|_2^2 = \sum_{l=1}^L \|\boldsymbol{g}_l^{+(i)}\|_2^2 \le \sum_{l=1}^L C_l^2 = C^2. \tag{2}$$

Each example's contribution to the joint query therefore remains bounded by C, even after incorporating slack feedback.

Consider the averaged extended gradient query on a sampled batch $\mathbb{B}$, $f_t^+(\mathbb{B}) = \frac{1}{B}\sum_{i\in\mathbb{B}}\boldsymbol{g}^{+(i)}$, which is evaluated at $\mathbb{B} = \mathbb{B}_t$ during iteration t. Under add/remove adjacency and the public fixed normalization constant B, equation 2 gives the ℓ_2 sensitivity bound

$$\Delta_2(f_t^+) = \sup_{\mathbb{B}\sim\mathbb{B}'} \big\|f_t^+(\mathbb{B}) - f_t^+(\mathbb{B}')\big\|_2 \le C/B. \tag{3}$$

This matches the sensitivity bound for standard DP-SGD with clipping threshold C and the same normalization.

The base preconditioner $\mathbf{D}_{l,t}$ is constructed from optimizer states maintained using previously privatized gradients, while $\rho_{l,t}$ is updated using previously released private slack feedback. Conditioned on these historical outputs, the preconditioning transformation is deterministic, and its application to current raw gradients is protected by the joint release. The inverse transformation $\boldsymbol{P}_{l,t}^{-1}\tilde{\boldsymbol{h}}_l$ and the updates to model parameters, optimizer states, layer-wise scales, and next preconditioners are post-processing of private outputs. These operations therefore do not incur additional privacy loss.

The independent Gaussian noise vectors added to the layer-wise sums all have standard deviation σC per coordinate. By equation 3, the mechanism has the same sensitivity calibration as standard DP-SGD, and the Gaussian privacy bound does not depend on the query dimension (Mironov et al., 2019). With the same sampling rule and rate and noise multiplier, the same privacy accountant therefore yields the same per-step privacy upper bound.

4.2 Utility Analysis

We analyze the private gradient supplied to the optimizer by the complete mechanism. Write $\boldsymbol{g}_t = \nabla \ell(\boldsymbol{\theta}_{t-1}; x)$ for a fresh example, $\mathbb{E}_t$ for expectation conditional on previous private outputs, and $u_{l,t} = \log \rho_{l,t}$.

Condition (stable feedback). Assume $\mathbb{E}_t \boldsymbol{g}_t = \nabla F(\boldsymbol{\theta}_{t-1})$ and $\mathbb{E}_t \|\boldsymbol{g}_{l,t}\|_2^2 \le G_l^2$. Use B independent samples. With the current model and $\mathbf{D}_{l,t}$ fixed, let $h_{l,t}(u) = \mathbb{E}_t[\hat{s}_{l,1}(u) - \gamma_{l,t}(u)]$ be the mean of the algorithm's private feedback. Assume a history-measurable root $u_{l,t}^\star$ of $h_{l,t}$ and $\mu > 0$ satisfy

$$(u - u_{l,t}^\star) h_{l,t}(u) \le -\mu \min\left\{(u - u_{l,t}^\star)^2,\, |u - u_{l,t}^\star|\right\}, \quad u \in \mathbb{R}. \tag{4}$$

The initial log-scale errors have finite first moments, and the expectations below are finite. The feedback pushes the scale toward equilibrium on average. The equilibrium can change with the model and historical moments.

Theorem 4.2 (Utility of scale-calibrated preconditioning) *Under the condition above, for $T \ge 1$ and $\eta_\rho > 0$, Algorithm 1 satisfies*

$$\begin{aligned} \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\|\tilde{\boldsymbol{g}}_t - \nabla F(\boldsymbol{\theta}_{t-1})\|_2^2 \le \frac{\sum_l G_l^2}{B} + \frac{\sigma^2 C^2}{B^2 T} \sum_{t=1}^{T}\sum_{l=1}^{L} \mathbb{E}\|\boldsymbol{P}_{l,t}^{-1}\|_F^2 \\ + \underbrace{\frac{2}{T}\sum_{t=1}^{T}\sum_{l=1}^{L} \mathbb{E}\left[(1-\alpha_{l,t}^\star)^2 \|\boldsymbol{g}_{l,t}\|_2^2\right]}_{\textit{calibrated clipping error}} + \underbrace{c\left(\frac{1+V_T}{\eta_\rho T} + \eta_\rho\right)}_{\textit{scale-tracking error}}. \end{aligned} \tag{5}$$

where

$$\alpha_{l,t}^\star = \min\left\{1, C_l / \left(e^{u_{l,t}^\star} \|\mathbf{D}_{l,t}\boldsymbol{g}_{l,t}\|_2\right)\right\}, \qquad V_T = \sum_{t=1}^{T-1}\sum_{l} \mathbb{E}|u_{l,t+1}^\star - u_{l,t}^\star|.$$

Set $\alpha_{l,t}^\star = 1$ for zero layer gradients. The constant c depends on the moment bounds, μ, initialization, and fixed mechanism parameters, but not on T or η_ρ. The full proof is provided in Appendix E.

Explanation of the theorem. The third term measures clipping error at the equilibrium scales. Its direction-changing component is the gradient-energy-weighted dispersion of $\alpha_{l,t}^\star$ across layers. Thus, when historical scale calibration makes these coefficients more consistent, it reduces this additional directional distortion; common magnitude attenuation may remain. The fourth term bounds the additional error from tracking changing scales, including changes induced by $\mathbf{D}_{l,t}$. For bounded V_T and fixed remaining parameters, $\eta_\rho \asymp T^{-1/2}$ makes this contribution $O(T^{-1/2})$; a fixed step size leaves an $O(\eta_\rho)$ term. The first two terms retain sampling error and inverse-transformed privacy noise, respectively, showing that alleviating clipping must be balanced against noise amplification.

5 Experiments

We evaluate SCP-Clip on sentence classification, image classification, and table-to-text generation to assess its effectiveness. We compare with Flat clipping (Abadi et al., 2016), Fixed per-layer (McMahan et al., 2018), Auto per-layer (Bu et al., 2023a), and Adaptive per-layer (He et al., 2023). Details of the baselines are provided in Appendix C. We also provide additional comparisons with the per-layer clipping variants of closely related clipping methods in Appendix I. We report all accuracies as averages over three random seeds. Additional settings are given in Appendix B.

We use Book-Keeping (BK) (Bu et al., 2023c) to implement per-layer clipping in our experiments. Algorithm 1 describes the procedure using explicit per-example gradients for clarity. We show that our method is compatible with efficient implicit implementations, such as BK and ghost clipping (Li et al., 2022b) (Appendix G), and incurs negligible computational overhead (Appendix H).

5.1 Performance Across Tasks

Sentence Classification. We fully fine-tune RoBERTa-base and RoBERTa-large (Liu et al., 2019) on SST-2, QNLI, QQP, and MNLI from GLUE (Wang et al., 2018) at $\epsilon \in \{3, 8\}$. We train for 20

epochs with Adam using the standard classification objective without text infilling. Further details are provided in Appendix B.1.

Table 1 shows that SCP-Clip outperforms existing per-layer clipping methods in most settings. For RoBERTa-large at $\epsilon = 3$, SCP-Clip achieves 91.34% (QNLI), exceeding Adaptive per-layer by 1.54%. Meanwhile, SCP-Clip also shows an overall advantage over Flat clipping.

Table 1: Test accuracy (%) of RoBERTa-base and RoBERTa-large, averaged over three seeds. MNLI reports matched/mismatched accuracy.

	RoBERTa-base				**RoBERTa-large**			
Method	**MNLI**	**QQP**	**QNLI**	**SST-2**	**MNLI**	**QQP**	**QNLI**	**SST-2**
$\epsilon = 3$								
Flat clipping	83.02/83.30	85.76	**87.11**	92.20	87.17/87.22	86.66	90.48	94.02
Fixed per-layer	80.19/81.04	81.75	83.98	90.17	84.87/84.78	83.13	88.25	92.34
Auto per-layer	82.02/82.56	83.26	85.85	91.40	86.37/86.66	84.78	89.60	**94.38**
Adaptive per-layer	82.83/83.27	85.67	86.13	92.03	87.10/87.20	86.80	89.80	93.87
SCP-Clip	**83.30/83.45**	**85.89**	87.06	**92.65**	**87.36/87.25**	**86.83**	**91.34**	93.95
$\epsilon = 8$								
Flat clipping	**83.76**/83.45	86.42	**87.73**	92.53	87.05/87.26	**87.27**	91.11	94.43
Fixed per-layer	82.07/81.21	82.34	84.17	90.47	85.91/85.61	83.21	89.15	93.09
Auto per-layer	82.24/82.84	83.49	86.42	92.09	86.53/86.93	85.22	90.10	94.50
Adaptive per-layer	83.70/83.97	86.23	87.13	92.40	87.67/87.57	87.20	90.77	94.03
SCP-Clip	83.74/**84.11**	**86.54**	87.61	**92.78**	**87.78/87.64**	87.16	**91.47**	**94.69**

Image Classification. We fully fine-tune ViT-small and ViT-large (Dosovitskiy et al., 2021) on CIFAR-10 and CIFAR-100. We set $\epsilon \in \{2, 4, 8\}$ and $\delta = 1/n^{1.1}$, where n is the training set size. All methods are trained for 5 epochs using DP-AdamW (Loshchilov & Hutter, 2019) with an effective batch size of 1,000 and a constant learning rate of 5×10^{-4}, without data augmentation. Additional details can be found in Appendix B.2.

Table 2 summarizes the results, where SCP-Clip consistently outperforms all existing per-layer clipping methods across datasets and privacy levels. Even at $\epsilon = 2$, SCP-Clip reaches 78.56% for ViT-small (CIFAR-100), outperforming Auto per-layer by 6.29%. Moreover, SCP-Clip maintains its advantage over Flat clipping.

Table 2: Top-1 test accuracy (%) of ViT-small and ViT-large, averaged over three seeds.

	ViT-small						**ViT-large**					
	CIFAR-10			**CIFAR-100**			**CIFAR-10**			**CIFAR-100**		
Method	$\epsilon = 2$	$\epsilon = 4$	$\epsilon = 8$	$\epsilon = 2$	$\epsilon = 4$	$\epsilon = 8$	$\epsilon = 2$	$\epsilon = 4$	$\epsilon = 8$	$\epsilon = 2$	$\epsilon = 4$	$\epsilon = 8$
Flat clipping	96.89	96.95	97.09	**81.88**	**83.01**	84.22	98.48	98.60	**98.87**	**89.47**	89.79	90.41
Fixed per-layer	96.23	96.31	96.68	68.98	70.84	77.36	98.20	98.29	98.46	80.54	81.30	87.81
Auto per-layer	96.51	96.59	96.79	72.27	74.83	80.63	98.29	98.41	98.52	83.56	85.15	88.24
Adaptive per-layer	96.59	96.64	96.77	74.61	76.03	81.12	98.23	98.37	98.55	85.68	85.92	88.54
SCP-Clip	**97.05**	**97.08**	**97.13**	78.56	80.31	**84.61**	**98.55**	**98.61**	98.70	89.07	**89.89**	**90.85**

Table-to-Text Generation. We compare SCP-Clip with Flat clipping and Adaptive per-layer for full fine-tuning of GPT-2 and GPT-2-medium on E2E (Novikova et al., 2017). We evaluate test BLEU and ROUGE-L. We follow the setup of Bu et al. (2023a) and use their reported Flat clipping results. See more details in Appendix B.3.

Table 3 confirms the trend extends to language generation. For GPT-2-medium, SCP-Clip reaches 63.14 (BLEU) at $\epsilon = 3$ compared to 61.69 for Adaptive per-layer.

5.2 Ablation Studies

To better understand the individual contributions of the base preconditioning and the scale calibration, we conduct ablation studies using RoBERTa-base and ViT-small. See Appendix B.4 for details.

Table 3: Test BLEU and ROUGE-L scores of GPT-2 and GPT-2-medium.

	GPT-2				GPT-2-medium			
	$\epsilon = 3$		$\epsilon = 8$		$\epsilon = 3$		$\epsilon = 8$	
Method	**BLEU**	**ROUGE-L**	**BLEU**	**ROUGE-L**	**BLEU**	**ROUGE-L**	**BLEU**	**ROUGE-L**
Flat clipping	61.34	65.87	63.60	67.07	**63.85**	**67.07**	64.22	67.53
Adaptive per-layer	61.10	65.12	63.42	66.69	61.69	66.68	63.80	67.32
SCP-Clip	**61.43**	**66.04**	**63.68**	**67.54**	63.14	67.05	**64.51**	**67.73**

Effect of base preconditioning. We compare Fixed per-layer with base preconditioning only to assess the contribution of preconditioning without scale calibration. Base preconditioning consistently improves performance over Fixed per-layer. For example, Table 4 shows that CIFAR-100 accuracy increases from 70.84% to 78.01% and a similar trend holds on GLUE tasks. These results support that base preconditioning before per-layer clipping improves model utility.

Effect of scale calibration. We assess scale calibration through two comparisons: Fixed per-layer versus scale calibration only, and base preconditioning only versus SCP-Clip. Both comparisons show improvements across all metrics in Table 4. On CIFAR-100 with ViT-small, scale calibration alone raises accuracy from 70.84% to 76.27%, while adding it to base preconditioning increases accuracy from 78.01% to 80.31%. Clipping scale calibration effectively tracks changes in gradient scale, helping achieve a better trade-off between clipping bias and noise perturbation.

Table 4: Effects of scale calibration and base preconditioning on accuracy (%, mean $\pm$ standard deviation over three seeds). MNLI-m denotes accuracy on the matched split of MNLI.

	RoBERTa-base, $\epsilon = 3$				**ViT-small**, $\epsilon = 4$	
Method	**MNLI-m**	**QQP**	**QNLI**	**SST-2**	**CIFAR-10**	**CIFAR-100**
Fixed per-layer	$80.19_{\pm 0.62}$	$81.75_{\pm 0.46}$	$83.98_{\pm 0.60}$	$90.17_{\pm 0.33}$	$96.31_{\pm 0.67}$	$70.84_{\pm 0.55}$
Scale calibration only	$81.92_{\pm 0.38}$	$83.77_{\pm 0.40}$	$85.09_{\pm 0.39}$	$91.24_{\pm 0.45}$	$96.52_{\pm 0.49}$	$76.27_{\pm 0.50}$
Base preconditioning only	$82.23_{\pm 0.28}$	$84.30_{\pm 0.46}$	$85.78_{\pm 0.38}$	$91.51_{\pm 0.35}$	$96.65_{\pm 0.33}$	$78.01_{\pm 0.72}$
SCP-Clip	$\mathbf{83.30}_{\pm 0.39}$	$\mathbf{85.89}_{\pm 0.41}$	$\mathbf{87.06}_{\pm 0.56}$	$\mathbf{92.65}_{\pm 0.59}$	$\mathbf{97.08}_{\pm 0.32}$	$\mathbf{80.31}_{\pm 0.44}$

6 Related Work

Layer-wise clipping. Early work on layer-wise clipping accommodates differences in gradient or update scale across layers (McMahan et al., 2018; Dupuy et al., 2022). He et al. (2023) and Bu et al. (2023a) subsequently explore the efficiency benefits of layer-wise clipping while highlighting its potential utility loss. More recently, Wang et al. (2025) further improve efficiency through GPU kernel fusion and reduced memory traffic. Complementary to prior work, we enhance the utility of layer-wise clipping by addressing its clipping bias.

Adaptive clipping. Existing adaptive clipping methods primarily adjust clipping thresholds or transform gradient representations. One line of work updates C_t using private estimates of gradient norm distributions while retaining the underlying clipping geometry (Andrew et al., 2021; Wei et al., 2025; Zou et al., 2026). Beyond threshold adjustment, AdaCliP (Pichapati et al., 2019) and GeoClip (Gilani et al., 2025) transform gradients before clipping, but their reliance on accumulated gradient statistics may limit the ability to track changes in gradient scale. In contrast, SCP-Clip jointly adapts the direction and scale of clipping. We defer a full discussion of related work to Appendix A.

7 Conclusion

In this work, we propose SCP-Clip, a per-layer clipping method for improving the efficiency–utility trade-off in differentially private training. By performing clipping in a scale-calibrated preconditioned space, SCP-Clip mitigates gradient direction distortion and adapts to evolving gradient scales. The method incurs no additional privacy cost and remains compatible with memory-efficient layer-wise execution. We demonstrate the effectiveness of SCP-Clip both theoretically and empirically. Future work will explore richer preconditioning structures and tighter characterizations of the trade-off between clipping bias and noise amplification.

AI USE STATEMENT

Large Language Models (LLMs) were used to assist with the writing and polishing of the manuscript. Specifically, they supported sentence rephrasing, grammar correction, and improvements in clarity, readability, and the overall flow of the text.

LLMs were not involved in research ideation, methodological development, experimental design, or data analysis. All research ideas, methods, experiments, and scientific interpretations were developed and carried out by the authors. The use of LLMs was limited to improving the presentation and linguistic quality of the manuscript.

The authors take full responsibility for the content of the manuscript, including any text generated or revised with LLM assistance, and for ensuring its accuracy, originality, and adherence to ethical research standards.

ETHICS STATEMENT

This work follows the principles of the ICLR Code of Ethics. No human subjects or animal experimentation were involved in this study. All datasets used, including CIFAR-10, CIFAR-100, SST-2, QNLI, QQP, MNLI, and E2E, are existing research benchmarks. We are committed to respecting applicable dataset usage guidelines and protecting privacy throughout the research process. No personally identifiable information was used, and no experiments were conducted that could raise privacy or security concerns. We remain committed to transparency, integrity, and responsible research practices.

REPRODUCIBILITY STATEMENT

We have made every effort to ensure the reproducibility of the results presented in this paper. The source code for SCP-Clip is provided as a ZIP archive in the supplementary material for anonymous review, facilitating replication and verification. The experimental setup, including datasets, model configurations, training procedures, hyperparameters, and baseline methods, is described in Section 5 and Appendices B and C. Together, these materials support independent reproduction and verification of our results.

REFERENCES

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pp. 308–318, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. doi: 10.1145/2976749.2978318. URL https://doi.org/10.1145/2976749.2978318.

Galen Andrew, Om Thakkar, Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 17455–17466. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/91cff01af640a24e7f9f7a5ab407889f-Paper.pdf.

Hilal Asi, John Duchi, Alireza Fallah, Omid Javidbakht, and Kunal Talwar. Private adaptive gradient methods for convex optimization. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 383–392. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/asi21a.html.

Zhiqi Bu, Jialin Mao, and Shiyun Xu. Scalable and efficient training of large convolutional neural networks with differential privacy. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 38305–38318. Curran Associates, Inc., 2022. doi: 10.52202/068431-2775. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/fa5617c176e76fee83f3f9947fdf9f3f-Paper-Conference.pdf.

Zhiqi Bu, Ruixuan Liu, Yu-Xiang Wang, Sheng Zha, and George Karypis. On the accuracy and efficiency of group-wise clipping in differentially private optimization, 2023a. URL https://arxiv.org/abs/2310.19215.

Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. Automatic clipping: Differentially private deep learning made easier and stronger. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 41727–41764. Curran Associates, Inc., 2023b. doi: 10.52202/075280-1808. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8249b30d877c91611fd8c7aa6ac2b5fe-Paper-Conference.pdf.

Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. Differentially private optimization on large model at small cost. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 3192–3218. PMLR, 23–29 Jul 2023c. URL https://proceedings.mlr.press/v202/bu23a.html.

Zhiqi Bu, Yu-Xiang Wang, Sheng Zha, and George Karypis. Differentially private bias-term fine-tuning of foundation models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4730–4751. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/bu24c.html.

Xiangyi Chen, Steven Wu, and Mingyi Hong. Understanding gradient clipping in private sgd: A geometric perspective. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 13773–13782. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/9ecff5455677b38d19f49ce658ef0608-Paper.pdf.

Marc Molina Van den bosch, Riccardo Taiello, Albert Sund Aillet, Andrea Protani, Miguel Angel Gonzalez Ballester, and Luigi Serio. DP-KFC: Data-free preconditioning for privacy-preserving deep learning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Z6HxJqAzbp.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011. URL http://jmlr.org/papers/v12/duchi11a.html.

Christophe Dupuy, Radhika Arava, Rahul Gupta, and Anna Rumshisky. An efficient dp-sgd mechanism for large scale nlu models. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4118–4122. IEEE, 2022.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and trends® in theoretical computer science*, 9(3-4):211–487, 2014.

Arun Ganesh, Brendan McMahan, and Abhradeep Thakurta. On design principles for private adaptive optimizers, 2025. URL https://arxiv.org/abs/2507.01129.

Atefeh Gilani, Naima Tasnim, Lalitha Sankar, and Oliver Kosut. Geoclip: Geometry-aware clipping for differentially private sgd. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 6233–6257. Curran Associates, Inc., 2025. doi: 10.52202/085713-0218. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/093dd0011ada459922e38b86317aa8f3-Paper-Conference.pdf.

Jiyan He, Xuechen Li, Da Yu, Huishuai Zhang, Janardhan Kulkarni, Yin Tat Lee, Arturs Backurs, Nenghai Yu, and Jiang Bian. Exploring the limits of differentially private deep learning with group-wise clipping. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=oze0clVGPeX.

Daeyoung Hong, Woohwan Jung, and Kyuseok Shim. DP-FROST: Differentially private fine-tuning of pre-trained models with freezing model parameters. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 6966–6984, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.465/.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Jaewoo Lee and Daniel Kifer. Scaling up differentially private deep learning with fast per-example gradient clipping. *Proceedings on Privacy Enhancing Technologies*, 2021.

Tian Li, Manzil Zaheer, Sashank Reddi, and Virginia Smith. Private adaptive optimization with side information. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 13086–13105. PMLR, 17–23 Jul 2022a. URL https://proceedings.mlr.press/v162/li22x.html.

Tian Li, Manzil Zaheer, Ken Liu, Sashank J. Reddi, Hugh Brendan McMahan, and Virginia Smith. Differentially private adaptive optimization with delayed preconditioners. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=j1zQGmQQOX1.

Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=bVuP3ltATMz.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models, 2018. URL https://arxiv.org/abs/1710.06963.

Ilya Mironov, Kunal Talwar, and Li Zhang. Rényi differential privacy of the sampled gaussian mechanism, 2019. URL https://arxiv.org/abs/1908.10530.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. The E2E dataset: New challenges for end-to-end generation. In Kristiina Jokinen, Manfred Stede, David DeVault, and Annie Louis (eds.), *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pp. 201–206, Saarbrücken, Germany, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5525. URL https://aclanthology.org/W17-5525/.

Venkatadheeraj Pichapati, Ananda Theertha Suresh, Felix X. Yu, Sashank J. Reddi, and Sanjiv Kumar. Adaclip: Adaptive clipping for private sgd, 2019. URL https://arxiv.org/abs/1908.07643.

Pranav Subramani, Nicholas Vadivelu, and Gautam Kamath. Enabling fast differentially private sgd via just-in-time compilation and vectorization. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 26409–26421. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/ddf9029977a61241841edeae15e9b53f-Paper.pdf.

Naima Tasnim, Lalitha Sankar, and Oliver Kosut. Dp-macadam: Differentially private mechanism with adaptive clipping and adaptive momentum, 2026. URL https://arxiv.org/abs/2606.05435.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi (eds.), *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446/.

Liangyu Wang, Junxiao Wang, Jie Ren, Zihang Xiang, David Keyes, and Di Wang. Private training large-scale models with efficient dp-sgd. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 27408–27436. Curran Associates, Inc., 2025. doi: 10.52202/085713-0922. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/27aa3a0e6d63db269977bb2df5607cb8-Paper-Conference.pdf.

Chengkun Wei, Weixian Li, Chen Gong, and Wenzhi Chen. Dc-sgd: Differentially private sgd with dynamic clipping through gradient norm distribution estimation. *IEEE Transactions on Information Forensics and Security*, 20:4498–4511, 2025. doi: 10.1109/TIFS.2025.3557755.

Hanshen Xiao, Zihang Xiang, Di Wang, and Srinivas Devadas. A theory to instruct differentially-private learning via clipping bias reduction. In *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 2170–2189, 2023. doi: 10.1109/SP46215.2023.10179409.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A. Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. Differentially private fine-tuning of language models, 2022. URL https://arxiv.org/abs/2110.06500.

Shuyan Zou, Shaowei Wang, Zhanxing Zhu, Jin Li, Changyu Dong, Vladimiro Sassone, and Han Wu. Slaclip: Gradient norm slacks can be indicator for adaptive clipping in DP-SGD. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=48suUeYKdb.

A Extended Related Work

Efficient private training and group-wise clipping. Efficient implementations of DP-SGD reduce the computational and memory overhead of per-example gradient clipping (Lee & Kifer, 2021; Subramani et al., 2021; Bu et al., 2022). Ghost clipping computes per-example gradient norms without explicitly materializing full per-example gradients (Li et al., 2022b). Book-Keeping reuses computations from the backward pass to reduce redundant work (Bu et al., 2023c). Group-wise clipping allows gradients to be clipped and aggregated during backpropagation. He et al. (2023) integrate adaptive per-layer clipping into this process, while Bu et al. (2023a) analyze the accuracy–memory trade-off across grouping strategies. FlashDP further improves execution efficiency with a cache-friendly implementation that fuses per-layer gradient computation, clipping, aggregation, and noise injection (Wang et al., 2025). These advances make private training more scalable, but independently clipping different layers can alter the direction of an example's full gradient. We retain the layer-wise organization and calibrate coordinate scales before clipping to mitigate this additional directional distortion.

Adaptive clipping and private threshold estimation. Adaptive clipping reduces the need to select a fixed threshold that remains suitable throughout training. One approach updates thresholds using private estimates of gradient norm statistics. Andrew et al. (2021) track a target quantile of user-update norms in federated learning. DC-SGD uses private histograms to select thresholds based on percentiles or error (Wei et al., 2025). SlaClip obtains threshold feedback by jointly releasing clipped gradients and encoded norm slack, without separately releasing norm statistics or increasing the sensitivity bound (Zou et al., 2026). Automatic Clipping instead uses normalization-style clipping rules to avoid tuning a separate clipping scale (Bu et al., 2023b).

Geometry-aware clipping and private adaptive optimization. Beyond choosing a scalar clipping threshold, geometry-aware methods adapt the gradient representation before privatization. AdaCliP centers and scales coordinates using historical private statistics (Pichapati et al., 2019), whereas GeoClip uses covariance-aware transformations (Gilani et al., 2025). This perspective also links gradient clipping to adaptive optimization methods (Asi et al., 2021). DP-MacAdam combines coordinate-wise adaptive clipping with Adam-style moment updates (Tasnim et al., 2026). Ganesh et al. (2025) analyze the scale-then-privatize ordering, in which rescaling precedes clipping and noise injection. Another distinction concerns the source of information used for preconditioning. AdaDPS uses non-sensitive side information (Li et al., 2022a), DP^2 constructs delayed estimates from previously privatized gradients (Li et al., 2023), and DP-KFC uses data-independent synthetic probes (den bosch et al., 2026).

B Experimental Details

Preconditioner initialization. For every layer in SCP-Clip, we initialize $\rho_{l,1} = 1$ and $\mathbf{D}_{l,1} = \boldsymbol{P}_{l,1} = \boldsymbol{I}_{d_l}$. The first iteration therefore clips gradients in their original coordinates. This initialization applies to the clipping preconditioner; the optimizer's first- and second-moment estimates remain initialized to zero. After the first joint private release, the optimizer updates its moments using the privatized gradient, and the updated second moments determine $\mathbf{D}_{l,2}$. The private slack feedback determines $\rho_{l,2}$ through the scale update in Algorithm 1. The second-moment-based definition of $\mathbf{D}_{l,t}$ applies from $t = 2$ onward.

Slack dimension and scale step size. Following the criterion of Zou et al. (2026), adapted to per-layer clipping, we choose

$$K = \left\lfloor \left(\frac{B}{2z_{0.995}\sigma\sqrt{L}} \right)^{2/3} \right\rfloor, \qquad z_{0.995} \approx 2.576,$$

where B is the effective batch size, σ is the calibrated noise multiplier, and L is the number of clipping groups. SCP-Clip uses fixed thresholds $C_l = C_0/\sqrt{L}$. We take the largest admissible integer to retain feedback resolution and keep K fixed within each run. The scale step size is $\eta_\rho = 0.2$ for all tasks.

Clipping groups. Each trainable linear, convolutional, embedding, or normalization module forms one clipping group, with its weight and bias clipped jointly. Fused QKV projections remain a single

group. RoBERTa uses the standard classification head; ViT's class token and positional embedding each form an additional group; and GPT-2 uses an output head untied from its input embedding. Table 5 summarizes the resulting partitions.

Table 5: Clipping groups for each model. Other groups lie outside the Transformer blocks.

Model	**Blocks**	**Groups/block**	**Other groups**	**Total L**
RoBERTa-base	12	8	6	102
RoBERTa-large	24	8	6	198
ViT-small	12	6	5	77
ViT-large	24	6	5	149
GPT-2	12	6	4	76
GPT-2-medium	24	6	4	148

B.1 Sentence Classification

We fully fine-tune RoBERTa-base and RoBERTa-large on SST-2, QNLI, QQP, and MNLI from the GLUE benchmark using the standard classification objective without text infilling. We use $\epsilon \in \{3, 8\}$ and $\delta = n^{-1.1}$, where n is the training set size. We report accuracy averaged over three random seeds, with MNLI matched and mismatched results reported separately.

We train for 20 epochs using Adam with $(\beta_1, \beta_2) = (0.9, 0.98)$, $\epsilon_{\text{num}} = 10^{-6}$, and no weight decay. The learning rate follows a 6% warm-up followed by linear decay; dropout and attention dropout are both 0.1. We select the peak learning rate from $\{1, 2, 4\} \times 10^{-4}$ and the SST-2 batch size from $\{512, 1024, 2048, 4096\}$ based on validation performance under an 80/20 split of its training set. The selected hyperparameters are transferred to the other tasks, with batch sizes scaled to preserve the sampling rate.

All methods share the initial global clipping threshold search space $C_0 \in \{0.1, 0.5, 1, 5\}$. For per-layer clipping, each layer is initialized with $C_{l,1} = C_0/\sqrt{L}$, where L is the number of layers. Noise multipliers are calibrated using Rényi differential privacy (RDP) accounting (Mironov et al., 2019).

B.2 Image Classification

We fully fine-tune ViT-small and ViT-large on CIFAR-10 and CIFAR-100, using the official splits of 50,000 training and 10,000 test images for each dataset. We use pretrained models from `timm` with 16×16 patches and replace the classification head with a 10-way or 100-way linear classifier. Images are resized to 224×224, converted to tensors with pixel values in $[0, 1]$, and normalized using a mean and standard deviation of 0.5 for each channel. We use the cross-entropy loss without data augmentation.

All methods are fine-tuned for five epochs using DP-AdamW with an effective batch size of 1,000, a constant learning rate of 5×10^{-4}, and weight decay of 0.01. We use the default moment coefficients $(\beta_1, \beta_2) = (0.9, 0.999)$ and optimizer stabilizer $\epsilon_{\text{num}} = 10^{-8}$. We evaluate at $\epsilon \in \{2, 4, 8\}$ with $\delta = n^{-1.1}$, where n is the training set size. We report top-1 test accuracy averaged over three seeds.

B.3 Table-to-Text Generation

We fully fine-tune pretrained GPT-2 and GPT-2-medium on the E2E dataset using its official training, validation, and test splits. We use the corresponding pretrained tokenizer and set the maximum training sequence length to 100 tokens. The serialized input record serves as context, and we optimize an autoregressive cross-entropy loss over the target description.

Both models are fine-tuned for 10 epochs using DP-AdamW with an effective batch size of 1,000, a constant learning rate of 10^{-3}, and weight decay of 0.01. We set $(\beta_1, \beta_2) = (0.9, 0.999)$ and the optimizer stabilizer $\epsilon_{\text{num}} = 10^{-8}$. We evaluate at $\epsilon \in \{3, 8\}$ and set $\delta = 8 \times 10^{-6}$, following the released E2E configuration.

For generation, we use beam search with a beam width of 5 and limit the generated continuation to 100 tokens, excluding the input context, as in the released implementation. We report BLEU and ROUGE-L scores on the test set using the official E2E evaluation script.

B.4 Ablation Studies

We follow the main experimental settings in Appendices B.1 and B.2. The ablations in Table 4 use RoBERTa-base at $\epsilon = 3$ and ViT-small at $\epsilon = 4$. We report accuracy as the mean $\pm$ standard deviation over three random seeds, with only the matched split reported for MNLI.

Scale calibration only. We remove coordinate-wise preconditioning by setting $\mathbf{D}_{l,t} = \boldsymbol{I}_{d_l}$ for all layers and iterations, yielding $\boldsymbol{P}_{l,t} = \rho_{l,t}\boldsymbol{I}_{d_l}$. We initialize $\rho_{l,1} = 1$, so that $\boldsymbol{P}_{l,1} = \boldsymbol{I}_{d_l}$. The joint gradient–slack release and the feedback-driven scale update are retained as in Algorithm 1, with the same slack dimension K and scale step size η_ρ as SCP-Clip. Both clipping and slack encoding therefore use the norm $\rho_{l,t}\|\boldsymbol{g}_l^{(i)}\|_2$, without coordinate-wise rescaling by historical second moments.

Base preconditioning only. We disable feedback-driven scale calibration by fixing $\rho_{l,t} = 1$ throughout training. We initialize $\mathbf{D}_{l,1} = \boldsymbol{I}_{d_l}$ and, for $t \geq 2$, construct the base preconditioner from the optimizer's second-moment estimates of previously privatized gradients. Thus, $\boldsymbol{P}_{l,t} = \mathbf{D}_{l,t}$ at every iteration. This variant does not use slack feedback to adjust the transformation.

Both variants retain gradient privatization in the transformed space and the corresponding inverse mapping before the optimizer update. The downstream optimizer is unchanged, and its first- and second-moment estimates are initialized to zero. In particular, setting $\mathbf{D}_{l,t} = \boldsymbol{I}_{d_l}$ does not disable the optimizer's moment updates. All four configurations are compared at the same total privacy budget, and retaining the joint slack release requires no separate privacy allocation (Section 4.1).

C Baseline Methods

Flat clipping. We use the standard gradient clipping scheme of Abadi et al. (2016). For each example, the full gradient $\boldsymbol{g}^{(i)}$ is clipped at a fixed global threshold C:

$$\overline{\boldsymbol{g}}^{(i)} = \alpha^{(i)}\boldsymbol{g}^{(i)}, \qquad \alpha^{(i)} = \min\left\{1, \frac{C}{\|\boldsymbol{g}^{(i)}\|_2}\right\}.$$

The clipped gradients are aggregated over the mini-batch, and Gaussian noise is added before the parameter update. All layers of a given example share the same clipping coefficient, preserving the direction of that example's full gradient.

Fixed per-layer. We adapt the fixed per-layer clipping strategy of McMahan et al. (2018), originally applied to user updates, to per-example gradients. Each layer is assigned a fixed threshold C_l, and its gradients are clipped independently using

$$\alpha_l^{(i)} = \min\left\{1, \frac{C_l}{\|\boldsymbol{g}_l^{(i)}\|_2}\right\}.$$

The clipped gradients are then aggregated and perturbed with Gaussian noise. In our experiments, all layers use $C_l = C_0/\sqrt{L}$ throughout training, where L is the number of layers.

Auto per-layer. We use the layer-wise automatic normalization scheme of Bu et al. (2023a). This method replaces hard clipping with the normalization rule $\boldsymbol{g}_l^{(i)}/(\|\boldsymbol{g}_l^{(i)}\|_2 + \gamma)$, where $\gamma > 0$ is a numerical stability constant, set to 0.01 in the original work. The normalized gradients are uniformly scaled according to the number of layers, then aggregated and perturbed with Gaussian noise. This method controls each example's contribution through normalization, without separately estimating gradient norm statistics to update layer-wise thresholds.

Adaptive per-layer. We follow the adaptive threshold strategy of He et al. (2023) and maintain a separate clipping threshold $C_{l,t}$ for each layer. At each iteration, the method estimates the fraction of examples whose layer gradient norms do not exceed $C_{l,t}$ and adds Gaussian noise to this estimate. The threshold is then updated multiplicatively based on the difference between the noisy fraction and a target quantile. This update allows each layer's threshold to track changes in its gradient

norm distribution. Gradients are clipped using the current layer-wise thresholds, aggregated, and perturbed with Gaussian noise. The additional privacy cost of threshold estimation is included in the overall privacy accounting.

D Proof of the Privacy Guarantee

We use the notation and assumptions of Theorem 4.1. Four supporting lemmas establish the dependence of the algorithm state on private history, the norm bound preserved by slack encoding, the joint Gaussian release, and the dimension independence of the Gaussian mechanism. We then reduce each iteration to the norm-bounded Gaussian sum mechanism analyzed by standard DP-SGD privacy accountants, establishing the same per-step privacy upper bound.

Let the joint private output of all layers at iteration t be

$$\mathbf{Y}_t = \left[\tilde{\boldsymbol{h}}_1; \tilde{\boldsymbol{s}}_1; \ldots; \tilde{\boldsymbol{h}}_L; \tilde{\boldsymbol{s}}_L\right], \qquad \mathsf{H}_{t-1} = (\mathbf{Y}_1, \ldots, \mathbf{Y}_{t-1}),$$

where semicolons denote vector concatenation. The algorithm state before the current query is

$$\mathcal{X}_t = \left(\boldsymbol{\theta}_{t-1}, \mathcal{S}_{t-1}, \{\boldsymbol{P}_{l,t}, \rho_{l,t}\}_{l=1}^{L}\right).$$

Public initialization includes the initial model parameters, optimizer state, preconditioners, positive layer-wise scales $\rho_{l,1}$, and fixed positive clipping thresholds C_l. The initial preconditioners are invertible, and all algorithmic hyperparameters are public. The clipping thresholds remain fixed throughout training. We adopt the convention $\mathrm{Clip}(\mathbf{0}, C) = \mathbf{0}$.

The per-example gradients are defined as in the main text, $\boldsymbol{g}_l^{(i)} = \nabla_{\boldsymbol{\theta}_l} \ell(\boldsymbol{\theta}_{t-1}; x_i)$. If their computation uses auxiliary randomness independent of the dataset and previous history, we first condition on this randomness. The bounds below hold uniformly over its realizations and therefore also hold after averaging over their common distribution. This conditioning excludes the randomness used for sampling and Gaussian privatization.

Lemma D.1 (Dependence on private history and post-processing) *For every t, the state $\mathcal{X}_t$ is a function of the public initialization and H_{t-1}. Consequently, conditioning on the same history fixes the model parameters, layer-wise scales, and complete preconditioners used in the current iteration; the clipping thresholds are public and fixed. Computing the next state from $\mathbf{Y}_t$ incurs no additional privacy loss.*

Proof. The initial state $\mathcal{X}_1$ is public, so the claim holds for $t = 1$. Suppose that $\mathcal{X}_t$ is determined by the public initialization and H_{t-1}. After releasing $\mathbf{Y}_t$, the private gradient in the original parameter space is $\tilde{\boldsymbol{g}}_l = \boldsymbol{P}_{l,t}^{-1} \tilde{\boldsymbol{h}}_l$. Adam updates the model parameters and optimizer state using only this private gradient and the previous state. The updated private optimizer state determines the next base preconditioner as specified in Algorithm 1:

$$\mathbf{D}_{l,t+1} = \mathrm{diag}\left(\frac{1}{\sqrt{\hat{\boldsymbol{v}}_{l,t}} + \epsilon_{\mathrm{num}}}\right).$$

Since $\lambda_l = C_l/\sqrt{K}$ is public and fixed, $\hat{\boldsymbol{s}}_l = \tilde{\boldsymbol{s}}_l/\lambda_l$ and the resulting target $\gamma_{l,t}$ in Algorithm 1 are functions of the public constants and $\mathbf{Y}_t$. The scale and complete preconditioner are updated as follows:

$$\rho_{l,t+1} = \rho_{l,t} \exp[\eta_\rho(\hat{s}_{l,1} - \gamma_{l,t})], \qquad \boldsymbol{P}_{l,t+1} = \rho_{l,t+1} \mathbf{D}_{l,t+1}.$$

Both updates therefore depend only on the previous state and $\mathbf{Y}_t$. The exponential update preserves $\rho_{l,t+1} > 0$. Together with the positive diagonal entries of $\mathbf{D}_{l,t+1}$ for $\epsilon_{\mathrm{num}} > 0$, this ensures that the next preconditioner remains invertible.

Hence, there exists a map F_t that requires no further access to the original data and satisfies $\mathcal{X}_{t+1} = F_t(\mathcal{X}_t, \mathbf{Y}_t)$. Induction proves the first claim.

For a fixed history h, this update is a deterministic map $F_{t,h}$ of the private output. For any measurable output set $\mathbb{O}$, $F_{t,h}(\mathbf{Y}_t) \in \mathbb{O}$ if and only if $\mathbf{Y}_t \in F_{t,h}^{-1}(\mathbb{O})$. The privacy inequality for $\mathbf{Y}_t$ on neighboring datasets therefore applies directly to this preimage. The updated state inherits the same privacy guarantee, establishing the post-processing claim. □

Lemma D.2 (Slack encoding preserves the clipping norm bound) *Let $C > 0$, let $K \geq 1$ be an integer, and let $\mathbf{z} \in \mathbb{R}^d$. Define*

$$r = \|\mathbf{z}\|_2, \qquad \lambda = \frac{C}{\sqrt{K}}, \qquad u = \sqrt{K}(C - r)_+,$$

where $(a)_+ = \max\{a, 0\}$. Construct the K-dimensional slack vector $\boldsymbol{s}$ using the filling rule (Zou et al., 2026) in equation 1. Then

$$\|\boldsymbol{s}\|_2^2 \leq C(C - r)_+, \qquad \big\|[\mathrm{Clip}(\mathbf{z}, C),\ \boldsymbol{s}]\big\|_2 \leq C. \tag{6}$$

Proof. The encoding $[\lambda \mathbf{1}_a, b, \mathbf{0}]$ in equation 1 can be written coordinate-wise as

$$s_j = \min\{\lambda, (u - (j-1)\lambda)_+\}, \qquad j = 1, \ldots, K.$$

This assigns the slack u to successive coordinates, each with capacity λ. Since $0 \leq u \leq K\lambda$, every coordinate satisfies $0 \leq s_j \leq \lambda$. Using

$$\min\{\lambda, (u - (j-1)\lambda)_+\} = (u - (j-1)\lambda)_+ - (u - j\lambda)_+,$$

and summing over j gives

$$\|\boldsymbol{s}\|_1 = \sum_{j=1}^{K} s_j = u - (u - K\lambda)_+ = u.$$

This identity also covers the endpoint $u = K\lambda$, where all K coordinates equal λ and no remainder coordinate is appended. It follows that

$$\|\boldsymbol{s}\|_2^2 = \sum_{j=1}^{K} s_j^2 \leq \lambda \sum_{j=1}^{K} s_j = \lambda u = C(C - r)_+.$$

If $r \geq C$, then $u = 0$, $\boldsymbol{s} = \mathbf{0}$, and $\|\,\mathrm{Clip}(\mathbf{z}, C)\|_2 = C$. The concatenated vector therefore has norm C. If $0 \leq r < C$, then $\mathrm{Clip}(\mathbf{z}, C) = \mathbf{z}$, and

$$\begin{aligned} \big\|[\mathrm{Clip}(\mathbf{z}, C),\ \boldsymbol{s}]\big\|_2^2 &= r^2 + \|\boldsymbol{s}\|_2^2 \\ &\leq r^2 + C(C - r) \\ &= C^2 - r(C - r) \leq C^2. \end{aligned}$$

The desired bound holds in both cases. □

Lemma D.3 (The global joint query and its Gaussian mechanism) *Fix a history $\mathsf{H}_{t-1} = h$ and define*

$$\boldsymbol{g}_l^{+(i)} = [\overline{\boldsymbol{g}}_l^{(i)},\ \boldsymbol{s}_l^{(i)}], \qquad \boldsymbol{g}^{+(i)} = [\boldsymbol{g}_1^{+(i)}; \ldots; \boldsymbol{g}_L^{+(i)}].$$

Let $C = (\sum_{l=1}^{L} C_l^2)^{1/2}$ be the equivalent global clipping threshold. Then $\|\boldsymbol{g}^{+(i)}\|_2 \leq C$. The batch query $f_{t,h}^{+}(\mathbb{B}) = B^{-1} \sum_{i \in \mathbb{B}} \boldsymbol{g}^{+(i)}$ has ℓ_2 sensitivity at most C/B under add/remove adjacency, where B is the public fixed normalization constant. Moreover, the joint release across all layers is exactly

$$\mathbf{Y}_t = f_{t,h}^{+}(\mathbb{B}_t) + \mathcal{N}\left(\mathbf{0}, \frac{\sigma^2 C^2}{B^2} \boldsymbol{I}_p\right), \qquad p = \sum_{l=1}^{L} (d_l + K). \tag{7}$$

Proof. By Lemma D.1, conditioning on the same history fixes all complete preconditioners $\boldsymbol{P}_{l,t} = \rho_{l,t} \mathbf{D}_{l,t}$; the clipping thresholds C_l are public and fixed. Apply equation 6 to each layer with $\mathbf{z} = \boldsymbol{P}_{l,t} \boldsymbol{g}_l^{(i)}$ and clipping threshold C_l. This gives $\|\boldsymbol{g}_l^{+(i)}\|_2 \leq C_l$. Since different layers occupy disjoint coordinate blocks,

$$\|\boldsymbol{g}^{+(i)}\|_2^2 = \sum_{l=1}^{L} \|\boldsymbol{g}_l^{+(i)}\|_2^2 \leq \sum_{l=1}^{L} C_l^2 = C^2.$$

Consider neighboring batches $\mathbb{B}' = \mathbb{B} \cup \{x\}$ under add/remove adjacency. Shared examples have identical gradients at the fixed model parameters. Their preconditioning, clipping, and encoding also use the same state, so their extended contributions cancel. Thus,

$$\begin{aligned}\|f^+_{t,h}(\mathbb{B}') - f^+_{t,h}(\mathbb{B})\|_2 &= \frac{1}{B}\|\boldsymbol{g}^+(x)\|_2 \\ &\le \frac{C}{B}.\end{aligned}$$

Removing an example only reverses the sign of the difference, so the same sensitivity bound applies.

For layer l, the algorithm adds $\boldsymbol{\xi}_{l,t} \sim \mathcal{N}(\mathbf{0}, \sigma^2 C^2 \boldsymbol{I}_{d_l+K})$ to the sum of extended gradients. These noise vectors are independent across layers. Their concatenation $\boldsymbol{\xi}_t = [\boldsymbol{\xi}_{1,t}; \ldots; \boldsymbol{\xi}_{L,t}]$ is therefore Gaussian with covariance

$$\operatorname{diag}\big(\sigma^2 C^2 \boldsymbol{I}_{d_1+K}, \ldots, \sigma^2 C^2 \boldsymbol{I}_{d_L+K}\big) = \sigma^2 C^2 \boldsymbol{I}_p.$$

Dividing the concatenated sum and noise by B yields the joint release in equation 7. This representation has exactly the same joint distribution as the layer-wise execution. □

Lemma D.4 (One-dimensional reduction of a Gaussian pair) *Let $\mathbf{a}, \mathbf{b} \in \mathbb{R}^p$ and $\tau > 0$. The distribution pair $\mathcal{N}(\mathbf{a}, \tau^2 \boldsymbol{I}_p)$ and $\mathcal{N}(\mathbf{b}, \tau^2 \boldsymbol{I}_p)$ has the same pairwise differential privacy guarantees as $\mathcal{N}(\kappa, 1)$ and $\mathcal{N}(0, 1)$, where $\kappa = \|\mathbf{a} - \mathbf{b}\|_2/\tau$. Consequently, sensitivity-calibrated privacy bounds for isotropic Gaussian mechanisms do not depend on the output dimension (Dwork & Roth, 2014, Appendix A).*

Proof. If $\mathbf{a} = \mathbf{b}$, the distributions coincide and the claim is immediate. Otherwise, choose an orthogonal matrix $\mathbf{U}$ such that $\mathbf{U}(\mathbf{a} - \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_2 \mathbf{e}_1$. Apply the same invertible affine transformation to both distributions:

$$\mathbf{z} = \frac{1}{\tau}\mathbf{U}(\mathbf{y} - \mathbf{b}).$$

The transformed distributions are $\mathcal{N}(\kappa \mathbf{e}_1, \boldsymbol{I}_p)$ and $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_p)$, which factorize as

$$\begin{aligned}\mathcal{N}(\kappa \mathbf{e}_1, \boldsymbol{I}_p) &= \mathcal{N}(\kappa, 1) \otimes \mathcal{N}(\mathbf{0}, \boldsymbol{I}_{p-1}), \\ \mathcal{N}(\mathbf{0}, \boldsymbol{I}_p) &= \mathcal{N}(0, 1) \otimes \mathcal{N}(\mathbf{0}, \boldsymbol{I}_{p-1}).\end{aligned}$$

For $p = 1$, the second factor is omitted. The remaining $p - 1$ coordinates have the same distribution under both laws and are independent of the first coordinate. Projection onto the first coordinate gives the one-dimensional pair. Conversely, appending independent standard Gaussian coordinates reconstructs the multidimensional pair from the one-dimensional pair. The two pairs are thus related by common post-processing maps in both directions and have identical pairwise privacy guarantees. Equivalently, their log-density ratio is

$$\log \frac{\varphi(z_1 - \kappa)}{\varphi(z_1)} = \kappa z_1 - \frac{\kappa^2}{2},$$

where φ is the standard normal density. This ratio does not depend on the remaining coordinates.

Furthermore, let $0 \le \kappa_1 \le \kappa_2$ with $\kappa_2 > 0$. The common randomized map

$$z \longmapsto \frac{\kappa_1}{\kappa_2} z + \sqrt{1 - \frac{\kappa_1^2}{\kappa_2^2}}\, W, \qquad W \sim \mathcal{N}(0, 1),$$

with independent W, transforms the pair with mean separation κ_2 into the pair with mean separation κ_1. By post-processing, a uniform upper bound on κ therefore gives a dimension-independent privacy bound for the Gaussian mechanism. □

Proof of Theorem 4.1. Fix an arbitrary iteration t and the same history $\mathsf{H}_{t-1} = h$. By Lemma D.1, the state $\mathcal{X}_t$ is fixed. Lemma D.3 bounds each extended gradient by C and the add/remove sensitivity of the batch query by C/B. The noise standard deviation is $\sigma C/B$. Thus, adding the feedback coordinates preserves the noise scale relative to the sensitivity bound. Lemma D.4 further shows that the additional coordinates introduce no dimension-dependent factor in the privacy bound for the Gaussian mechanism.

To connect this mechanism to the sampling rule and privacy accountant in the main text, define the normalized per-example query

$$\mathbf{w}_{t,h}(x) = \frac{\boldsymbol{g}^{+}(x)}{C}.$$

Lemma D.3 gives $\|\mathbf{w}_{t,h}(x)\|_2 \le 1$. Since B/C is a public fixed positive scalar, rescaling the joint release in equation 7 yields

$$\frac{B}{C}\mathbf{Y}_t = \sum_{x\in\mathbb{B}_t} \mathbf{w}_{t,h}(x) + \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}_p). \tag{8}$$

This is precisely the norm-bounded Gaussian sum mechanism analyzed by standard DP-SGD privacy accountants. It uses the same sampling rule, sums per-example vectors with norm at most 1, and adds isotropic Gaussian noise with multiplier σ. Indeed, applying standard global clipping with threshold 1 to $\mathbf{w}_{t,h}(x)$ leaves every query vector unchanged.

Under the same add/remove adjacency, sampling rule and rate, and noise multiplier, the same standard per-step privacy upper bound therefore applies. This bound is valid for the query class at every fixed history and does not depend on the output dimension. The reduction in equation 8 identifies the joint release with the standard bounded-sum mechanism; the same valid privacy accountant handles the effect of sampling. Restoring the factor C/B is invertible post-processing and preserves the guarantee. Hence, the joint release admits the same per-step privacy upper bound as standard DP-SGD with clipping threshold C.

Finally, the slack feedback is already included in the joint release. Its contribution is controlled by the common norm bound established in Lemmas D.2 and D.3, so it requires no separate privacy accounting. The base preconditioner is constructed from historical private optimizer states, and the layer-wise scale is updated using historical private slack feedback. Their product determines the complete preconditioner, while the preconditioned query on current raw gradients is protected by the joint mechanism. By Lemma D.1, the inverse transformation, optimizer update, and updates to the layer-wise scales and next preconditioners are all post-processing. They therefore incur no additional privacy loss. The argument holds for every fixed history, proving the per-step guarantee in Theorem 4.1. □

E Proof of the Joint Utility Bound

We prove Theorem 4.2 for the complete mechanism in Algorithm 1. Throughout, the data are fixed when using empirical-risk/Poisson sampling, and all conditional expectations are taken given the private history H_{t-1}. The current parameters, private optimizer state, base preconditioners, and layer-wise scales are consequently fixed. A fresh example x is independent of the current release randomness conditionally on this history. Expectations over a fresh example include any auxiliary per-example gradient randomness in the definition of F. For this appendix, write $G^2 = \sum_l G_l^2$.

1. Original-space form and privacy-noise covariance. For a layer and example, put

$$\alpha_{l,t}(x) = \min\left\{1, \frac{C_l}{\rho_{l,t}\|\mathbf{D}_{l,t}\boldsymbol{g}_{l,t}(x)\|_2}\right\}, \qquad \boldsymbol{c}_t(x) = [\alpha_{l,t}(x)\boldsymbol{g}_{l,t}(x)]_{l=1}^{L}.$$

The zero-gradient convention is as in the main text. Since clipping is radial in the transformed space,

$$\boldsymbol{P}_{l,t}^{-1}\,\mathrm{Clip}(\boldsymbol{P}_{l,t}\boldsymbol{g}_{l,t}(x), C_l) = \alpha_{l,t}(x)\boldsymbol{g}_{l,t}(x).$$

Thus, writing $\overline{\boldsymbol{c}}_t = B^{-1}\sum_{i\in\mathbb{B}_t}\boldsymbol{c}_t(x_i)$,

$$\tilde{\boldsymbol{g}}_t = \overline{\boldsymbol{c}}_t + \boldsymbol{\zeta}_t, \qquad \mathrm{Cov}_t(\boldsymbol{\zeta}_t) = \frac{\sigma^2 C^2}{B^2}\,\mathrm{blkdiag}_{l=1}^{L}\big(\rho_{l,t}^{-2}\mathbf{D}_{l,t}^{-2}\big). \tag{9}$$

The Gaussian vector $\boldsymbol{\zeta}_t$ is centered and independent of the current sampled batch given the history. In particular,

$$N_t := \mathbb{E}_t\|\boldsymbol{\zeta}_t\|_2^2 = \frac{\sigma^2 C^2}{B^2}\sum_l \rho_{l,t}^{-2}\|\mathbf{D}_{l,t}^{-1}\|_F^2. \tag{10}$$

This calculation concerns the actual original space input to Adam. The current slack release updates only the next scale, so it does not alter equation 9 within this iteration.

2. Exact finite-dimensional feedback and its second moment. Let

$$r_l(x;u) = \frac{e^u \|\mathbf{D}_{l,t}\boldsymbol{g}_{l,t}(x)\|_2}{C_l}.$$

The j-th normalized slack coordinate produced by the encoding in Section 3.2 is exactly

$$q_{l,j}(x;u) = \Pi_{[0,1]}\big(K(1-r_l(x;u)) - (j-1)\big), \qquad j = 1,\dots,K. \tag{11}$$

To see this, express the slack in units of λ_l:

$$\frac{\sqrt{K}\max\{C_l - e^u\|\mathbf{D}_{l,t}\boldsymbol{g}_l\|_2, 0\}}{\lambda_l} = K\max\{1 - r_l(x;u), 0\}.$$

Filling coordinates consecutively gives equation 11. In particular,

$$q_{l,1} = \Pi_{[0,1]}(K(1-r_l)), \qquad q_{l,K} = \Pi_{[0,1]}(1 - Kr_l).$$

Define the normalized sampled averages

$$A_{l,t}(u) = \frac{1}{B}\sum_{i\in\mathbb{B}_t} q_{l,1}(x_i;u), \qquad Q_{l,t}(u) = \frac{1}{B}\sum_{i\in\mathbb{B}_t} q_{l,K}(x_i;u).$$

After normalizing the joint release, the Gaussian noise in each slack coordinate has variance

$$\tau_l^2 = \frac{\sigma^2 C^2}{B^2\lambda_l^2} = \frac{\sigma^2 C^2 K}{B^2 C_l^2}.$$

Writing these noises as $\xi_{l,1}$ and $\xi_{l,K}$, the actual feedback at log scale u is

$$Y_{l,t}(u) = A_{l,t}(u) + \xi_{l,1} - \Pi_{[0,1]}\left(\frac{1 + (Q_{l,t}(u) + \xi_{l,K})/C_l}{2}\right). \tag{12}$$

Following SlaClip (Zou et al., 2026) and its public implementation, the target scales the normalized last slack coordinate $\hat{s}_{l,K} = Q_{l,t}(u) + \xi_{l,K}$ by $1/C_l$ to account for the clipping scale in the near-zero-gradient feedback. When $K = 1$, the two displayed noises are the same random variable; our bounds do not require independence between them. In all cases, $h_{l,t}(u) = \mathbb{E}_t[Y_{l,t}(u)]$ is the expectation of this complete expression, including the nonlinear projection of the noisy target.

For both sampling choices in the theorem,

$$0 \le p := \mathbb{E}_t A_{l,t}(u) \le 1, \qquad \mathrm{Var}_t(A_{l,t}(u)) \le \frac{1}{B}. \tag{13}$$

For B independent draws this follows directly from $q_{l,1} \in [0,1]$ (and the sharper bound $1/(4B)$ holds). For Poisson inclusion, let q be the inclusion probability and $B = nq$. Then

$$A = \frac{1}{B}\sum_{i=1}^{n} I_i q_{l,1}(x_i;u), \qquad I_i \sim \mathrm{Bernoulli}(q) \quad \text{independently}.$$

Its mean is the dataset average, and its variance is at most $q(1-q)n/B^2 = (1-q)/B \le 1/B$.

Let Γ denote the projected target in equation 12, so $0 \le \Gamma \le 1$. For every real A and $\Gamma \in [0,1]$,

$$(A-\Gamma)^2 \le \max\{A^2, (A-1)^2\} \le A^2 + (A-1)^2.$$

Consequently, using equation 13,

$$\mathbb{E}_t(A-\Gamma)^2 \le 2\,\mathrm{Var}_t(A) + 2p^2 - 2p + 1 \le 1 + \frac{2}{B}.$$

No independence between A and Γ is needed. The inequality $(a+b)^2 \le 2a^2 + 2b^2$ now gives

$$\mathbb{E}_t Y_{l,t}(u)^2 \le 2\left(1 + \frac{2}{B} + \tau_l^2\right) =: M_l. \tag{14}$$

For an exactly fixed batch size, $A \in [0,1]$ also holds pathwise, and one may replace M_l by $2(1+\tau_l^2)$. If additionally $K \ge 2$, $\xi_{l,1}$ is independent of $A - \Gamma$, and the sharper replacement $M_l = 1 + \tau_l^2$ is valid. The stated theorem uses the common bound equation 14 to cover both sampling rules and every $K \ge 1$.

3. Tracking under the actual multiplicative update. The multiplicative scale update is exactly $u_{l,t+1} = u_{l,t} + \eta_\rho Y_{l,t}$ in log coordinates. Write $e_{l,t} = u_{l,t} - u^\star_{l,t}$, $b(e) = \min\{e^2, 1\}$, and introduce the potential

$$\Psi(e) = \begin{cases} e^2/2, & |e| \le 1, \\ |e| - 1/2, & |e| > 1. \end{cases} \tag{15}$$

This nonnegative function satisfies $\Psi(e) \le |e|$. Its derivative $\Psi'(e) = \Pi_{[-1,1]}(e)$ is 1-Lipschitz. The function itself is also 1-Lipschitz. The condition in equation 4 implies $\Psi'(e_{l,t})h_{l,t}(u_{l,t}) \le -\mu b(e_{l,t})$: for $|e_{l,t}| \le 1$ this is the condition itself, and for $|e_{l,t}| > 1$ it follows by dividing that condition by $|e_{l,t}|$. Therefore, for real a, b,

$$\Psi(a+b) \le \Psi(a) + \Psi'(a)b + \frac{b^2}{2}, \qquad |\Psi(a) - \Psi(b)| \le |a - b|. \tag{16}$$

Using the exact log-scale update and conditioning on the history,

$$\begin{aligned} \mathbb{E}_t \Psi(e_{l,t} + \eta_\rho Y_{l,t}) &\le \Psi(e_{l,t}) + \eta_\rho \Psi'(e_{l,t}) h_{l,t}(u_{l,t}) + \frac{\eta_\rho^2}{2} \mathbb{E}_t Y_{l,t}^2 \\ &\le \Psi(e_{l,t}) - \mu \eta_\rho b(e_{l,t}) + \frac{\eta_\rho^2 M_l}{2}. \end{aligned} \tag{17}$$

For $t < T$ let $\Delta_{l,t} = u^\star_{l,t+1} - u^\star_{l,t}$. The actual next tracking error equals $e_{l,t} + \eta_\rho Y_{l,t} - \Delta_{l,t}$. The second inequality in equation 16 gives, pathwise, an additional potential cost at most $|\Delta_{l,t}|$. This remains valid even though the next equilibrium is correlated with the current gradient release and scale feedback. For the terminal step set $\Delta_{l,T} = 0$ and use the auxiliary error $u_{l,T+1} - u^\star_{l,T}$ only to close the telescoping sum. Summing equation 17, averaging over histories, and dropping the nonnegative terminal potential yields

$$\frac{1}{T} \sum_{t=1}^{T} \mathbb{E} b(e_{l,t}) \le \frac{\mathbb{E}\Psi(e_{l,1}) + \sum_{t=1}^{T-1} \mathbb{E}|\Delta_{l,t}|}{\mu \eta_\rho T} + \frac{\eta_\rho M_l}{2\mu}. \tag{18}$$

4. Transferring scale error to clipping error. Let $\boldsymbol{c}^\star_t(x) = [\alpha^\star_{l,t}(x) \boldsymbol{g}_{l,t}(x)]_{l=1}^L$ be the original-space clipped gradient at the reference scales in the main theorem. Writing an argument x makes the fresh-example dependence explicit; it is suppressed in the main text. For a nonzero layer gradient, set

$$v = \log(\|\mathbf{D}_{l,t} \boldsymbol{g}_{l,t}(x)\|_2 / C_l).$$

The clipping coefficient at log scale u is $\varphi(u + v)$, where

$$\varphi(z) = \min\{1, e^{-z}\} = e^{-\max\{z,0\}}.$$

This function takes values in $[0, 1]$ and is 1-Lipschitz. Hence

$$|\alpha_{l,t}(x) - \alpha^\star_{l,t}(x)|^2 \le \min\{|u_{l,t} - u^\star_{l,t}|^2, 1\} = b(e_{l,t}). \tag{19}$$

The same inequality holds for a zero layer gradient under the stated convention. The layer moment bounds imply

$$\mathbb{E}_t \|\boldsymbol{c}_t(x) - \boldsymbol{c}^\star_t(x)\|_2^2 \le \sum_l G_l^2 b(e_{l,t}).$$

With $R^\star_t = \mathbb{E}_t \|\boldsymbol{c}^\star_t(x) - \boldsymbol{g}_t(x)\|_2^2$, we obtain

$$\mathbb{E}_t \|\boldsymbol{c}_t(x) - \boldsymbol{g}_t(x)\|_2^2 \le 2R^\star_t + 2 \sum_l G_l^2 b(e_{l,t}). \tag{20}$$

Thus, the tracking bound directly controls errors in the clipping coefficients.

5. Conditional estimation error and the unified bound. Let $\mathbf{m}_t = \mathbb{E}_t \boldsymbol{g}_t(x) = \nabla F(\boldsymbol{\theta}_{t-1})$ and $\mathbf{a}_t = \mathbb{E}_t \boldsymbol{c}_t(x)$. For either sampling rule,

$$\mathbb{E}_t \bar{\boldsymbol{c}}_t = \mathbf{a}_t, \qquad \mathbb{E}_t \|\bar{\boldsymbol{c}}_t - \mathbf{a}_t\|_2^2 \le \frac{\mathbb{E}_t \|\boldsymbol{c}_t(x)\|_2^2}{B} \le \frac{G^2}{B}. \tag{21}$$

For independent fixed-size sampling this is the usual variance of an average. For Poisson sampling, the first bound equals $q(1-q)B^{-2}\sum_i \|\boldsymbol{c}_t(x_i)\|_2^2 = (1-q)B^{-1}\mathbb{E}_x\|\boldsymbol{c}_t(x)\|_2^2$ before discarding $1-q$. The final inequality uses $0 \le \alpha_{l,t} \le 1$. By equation 9, independence and centering of the current Gaussian gradient noise, and Jensen's inequality,

$$\begin{aligned}\mathbb{E}_t\|\tilde{\boldsymbol{g}}_t - \mathbf{m}_t\|_2^2 &= \|\mathbf{a}_t - \mathbf{m}_t\|_2^2 + \mathbb{E}_t\|\overline{\boldsymbol{c}}_t - \mathbf{a}_t\|_2^2 + N_t \\ &\le \mathbb{E}_t\|\boldsymbol{c}_t(x) - \boldsymbol{g}_t(x)\|_2^2 + \frac{G^2}{B} + N_t \\ &\le 2R_t^\star + 2\sum_l G_l^2 b(e_{l,t}) + \frac{G^2}{B} + N_t. \end{aligned} \tag{22}$$

Averaging over histories and $t = 1, \dots, T$ and substituting equation 18 gives the following explicit bound:

$$\begin{aligned}&\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\tilde{\boldsymbol{g}}_t - \nabla F(\boldsymbol{\theta}_{t-1})\|_2^2 \\ &\le \frac{2}{T}\sum_{t=1}^{T}\sum_{l=1}^{L}\mathbb{E}\big[(1-\alpha_{l,t}^\star)^2\|\boldsymbol{g}_{l,t}\|_2^2\big] + \frac{G^2}{B} + \frac{\sigma^2C^2}{B^2T}\sum_{t=1}^{T}\sum_{l=1}^{L}\mathbb{E}\|\boldsymbol{P}_{l,t}^{-1}\|_F^2 \\ &\quad + \frac{2}{\mu\eta_\rho T}\sum_l G_l^2\left(\mathbb{E}\Psi(e_{l,1}) + \sum_{t=1}^{T-1}\mathbb{E}|u_{l,t+1}^\star - u_{l,t}^\star|\right) + \frac{2\eta_\rho}{\mu}\sum_l G_l^2\left(1 + \frac{2}{B} + \tau_l^2\right).\end{aligned} \tag{23}$$

Here the clipping term follows from $\|\boldsymbol{c}_t^\star - \boldsymbol{g}_t\|_2^2 = \sum_l (1-\alpha_{l,t}^\star)^2\|\boldsymbol{g}_{l,t}\|_2^2$, and the noise term uses $\|\boldsymbol{P}_{l,t}^{-1}\|_F^2 = \rho_{l,t}^{-2}\|\mathbf{D}_{l,t}^{-1}\|_F^2$.

To obtain the compact statement in the main text, use $\Psi(e) \le |e|$ and

$$\sum_l G_l^2\sum_{t=1}^{T-1}\mathbb{E}|u_{l,t+1}^\star - u_{l,t}^\star| \le \big(\max_l G_l^2\big)V_T.$$

One explicit admissible constant is

$$\begin{aligned}c = \frac{2}{\mu}\Bigg[&\sum_l G_l^2\mathbb{E}|u_{l,1} - u_{l,1}^\star| + \max_l G_l^2 \\ &+ \sum_l G_l^2\left(1 + \frac{2}{B} + \frac{\sigma^2C^2K}{B^2C_l^2}\right)\Bigg].\end{aligned} \tag{24}$$

The tracking terms in equation 23 are therefore at most $c((1+V_T)/(\eta_\rho T) + \eta_\rho)$. For fixed initialization, moment bounds, stability constant, and mechanism parameters, c is independent of T, η_ρ, and V_T. This proves equation 5. □

6. Geometric interpretation of the same bound. For a nonzero full gradient, define

$$w_l = \frac{\|\boldsymbol{g}_l\|_2^2}{\|\boldsymbol{g}\|_2^2}, \qquad \bar{\alpha}^\star = \sum_l w_l\alpha_l^\star.$$

Expanding the square and using $\sum_l w_l(\alpha_l^\star - \bar{\alpha}^\star) = 0$ gives

$$\begin{aligned}\|\boldsymbol{c}_t^\star - \boldsymbol{g}\|_2^2 &= \|\boldsymbol{g}\|_2^2\sum_l w_l(1-\alpha_l^\star)^2 \\ &= \|\boldsymbol{g}\|_2^2(1-\bar{\alpha}^\star)^2 + \|\boldsymbol{g}\|_2^2\sum_l w_l(\alpha_l^\star - \bar{\alpha}^\star)^2.\end{aligned}$$

The two terms are the squared parallel attenuation and the squared orthogonal distortion, respectively, because $\langle \boldsymbol{c}_t^\star - \bar{\alpha}^\star\boldsymbol{g}, \boldsymbol{g}\rangle = 0$. For nonzero layers define $z_l^\star = u_{l,t}^\star + \log(\|\mathbf{D}_{l,t}\boldsymbol{g}_l\|_2/C_l)$, so

$\alpha_l^\star = \varphi(z_l^\star)$. The pairwise identity for weighted variance and the Lipschitz property of φ imply

$$\begin{aligned}\sum_l w_l(\alpha_l^\star - \bar{\alpha}^\star)^2 &= \frac{1}{2}\sum_{l,k} w_l w_k(\varphi(z_l^\star) - \varphi(z_k^\star))^2 \\ &\le \frac{1}{2}\sum_{l,k} w_l w_k(z_l^\star - z_k^\star)^2 \\ &= \sum_l w_l \left(z_l^\star - \sum_k w_k z_k^\star\right)^2.\end{aligned}$$

Zero-weight layers are omitted from logarithmic expressions; every error term is defined as zero for a zero full gradient.

A scale-dominated example. Consider a fixed iteration with equal thresholds $C_l = C_0$ and $\boldsymbol{g}_l(x) = a_l r(x)\mathbf{v}_l(x)$, where $a_l > 0$, $r(x) > 0$, and $\|\mathbf{v}_l(x)\|_2 = 1$. If historical calibration accurately identifies these scales so that $\mathbf{D}_l = a_l^{-1}\mathbf{I}$, then $\|\mathbf{D}_l \boldsymbol{g}_l(x)\|_2 = r(x)$ in every layer. Since the thresholds, normalized slack-noise distributions, and transformed norm processes agree, the functions $h_l(u)$ agree. Under the stability condition, they consequently have a common equilibrium $u^\star$. At that equilibrium,

$$\alpha_l^\star(x) = \min\{1, C_0/(e^{u^\star} r(x))\}$$

is identical across layers, making the additional perpendicular error exactly zero. In contrast, ordinary per-layer clipping has coefficients $\min\{1, C_0/(a_l r(x))\}$; its perpendicular error is strictly positive whenever unequal layer scales cause unequal coefficients with positive probability. When calibration is imperfect, the log-norm dispersion above quantifies its residual geometric error.

F Clipping-Bias Diagnostics

Experimental setup. We fully fine-tune RoBERTa-base on the full SST-2 training split for 20 epochs (approximately 1,300 updates), with an effective batch size of 1,024 and a peak learning rate of 2×10^{-4}. The remaining optimization settings follow Appendix B.1. We compare standard per-layer clipping (PL) and base-preconditioned per-layer clipping (PC), using the partition in Appendix B ($L = 102$) and fixed thresholds $C_l = C_0/\sqrt{L}$, where $C_0 = 1$. PC uses $\rho_{l,t} = 1$ throughout.

Shared-checkpoint gradient measurements. For each seed, Figure 1 and Figures 2(a,b) use shared model checkpoints and Adam states. At each checkpoint $\boldsymbol{\theta}_{t-1}$, we construct $\mathbf{D}_{l,t}$ from the saved second-moment estimate $\hat{\boldsymbol{v}}_{l,t-1}$ as in Section 3.1, retaining $\mathbf{D}_{l,1} = \mathbf{I}_{d_l}$ at initialization. We freeze the model and preconditioner, compute the per-example gradients $\boldsymbol{g}_{l,t}^{(i)} = \nabla_{\boldsymbol{\theta}_l}\ell(\boldsymbol{\theta}_{t-1}; x_i)$ once on the current mini-batch $\mathbb{B}_t$, and apply both clipping rules to these identical gradients, with the same dropout realization. With $B_t = |\mathbb{B}_t|$, the batch gradients in the original parameter space are

$$\begin{aligned}\boldsymbol{g}_t &= \frac{1}{B_t}\sum_{i\in\mathbb{B}_t}\left[\boldsymbol{g}_{l,t}^{(i)}\right]_{l=1}^{L}, \\ \boldsymbol{g}_{\mathrm{c},t}^{\mathrm{PL}} &= \frac{1}{B_t}\sum_{i\in\mathbb{B}_t}\left[\mathrm{Clip}(\boldsymbol{g}_{l,t}^{(i)}, C_l)\right]_{l=1}^{L}, \\ \boldsymbol{g}_{\mathrm{c},t}^{\mathrm{PC}} &= \frac{1}{B_t}\sum_{i\in\mathbb{B}_t}\left[\mathbf{D}_{l,t}^{-1}\,\mathrm{Clip}(\mathbf{D}_{l,t}\boldsymbol{g}_{l,t}^{(i)}, C_l)\right]_{l=1}^{L},\end{aligned}$$

where brackets concatenate all clipping groups. Figure 1 and Figure 2(a) respectively report, for $m \in \{\mathrm{PL}, \mathrm{PC}\}$,

$$\mathrm{CosSim}_t^m = \frac{\langle \boldsymbol{g}_{\mathrm{c},t}^m, \boldsymbol{g}_t\rangle}{\|\boldsymbol{g}_{\mathrm{c},t}^m\|_2\,\|\boldsymbol{g}_t\|_2}, \qquad \mathrm{RelErr}_t^m = \frac{\|\boldsymbol{g}_{\mathrm{c},t}^m - \boldsymbol{g}_t\|_2}{\|\boldsymbol{g}_t\|_2}.$$

Both metrics use the shared unclipped reference $\boldsymbol{g}_t$ and are computed after batch aggregation, before current-step noise addition or an optimizer update. The unclipped cosine reference equals one. We record the first measurement at the start of the second training step, and then every 50 updates. Curves average the metrics over three seeds at matched steps.

Group-wise norm ratios. For Figure 2(b), we compute

$$r_{l,t}^{(i,\mathrm{PL})} = \frac{\|\boldsymbol{g}_{l,t}^{(i)}\|_2}{C_l}, \qquad r_{l,t}^{(i,\mathrm{PC})} = \frac{\|\mathbf{D}_{l,t}\boldsymbol{g}_{l,t}^{(i)}\|_2}{C_l}.$$

Each bar averages the ratios within each mini-batch, over every update in the final 100-update window, and across three seeds. The six displayed groups are the attention and feed-forward groups in encoder blocks 1, 6, and 12. The line $r = 1$ marks the clipping boundary for an individual example.

Training-loss comparison. Figure 2(c) reports separate PL, PC, and unclipped training runs with $\sigma = 0$, sharing initialization, training-data order, and optimization hyperparameters. PC updates its preconditioner from its own preceding Adam second moments. We record mini-batch training loss initially and every 50 optimizer updates.

G Implementation with BK and Kronecker Scaling

We implement SCP-Clip on the existing BK per-layer clipping path. Forward hooks cache activations, and backward hooks obtain output derivatives, compute per-example norms, and directly accumulate clipped gradients within each clipping group. We retain this workflow and incorporate coordinate scaling through diagonal Kronecker factors.

Kronecker coordinate scaling. For a matrix parameter, reshape the diagonal entries of $\mathbf{D}_{l,t}$ into a positive matrix $\mathbf{S}$, with $S_{oj} = (\sqrt{\hat{v}_{oj,t-1}} + \epsilon_{\mathrm{num}})^{-1}$ for $t \geq 2$. We suppress l, t below. Let $Z_{oj} = \log S_{oj}$ and denote its row, column, and overall means by $\bar{Z}_{o\cdot}$, $\bar{Z}_{\cdot j}$, and $\bar{Z}$. The positive factors

$$q_o = \exp(\bar{Z}_{o\cdot} - \bar{Z}/2), \qquad r_j = \exp(\bar{Z}_{\cdot j} - \bar{Z}/2)$$

minimize $\sum_{o,j}(Z_{oj} - \log q_o - \log r_j)^2$. Thus, $\mathbf{S}^{\mathrm{K}} = \mathbf{q}\mathbf{r}^\top$ approximates the target coordinate scales in log space. Under column-wise vectorization, the corresponding transform is

$$\mathbf{D}^{\mathrm{K}} = \mathrm{diag}(\mathbf{r}) \otimes \mathrm{diag}(\mathbf{q}), \qquad \mathbf{D}^{\mathrm{K}}\,\mathrm{vec}(\mathbf{G}) = \mathrm{vec}(\mathrm{diag}(\mathbf{q})\mathbf{G}\,\mathrm{diag}(\mathbf{r}))\,.$$

Trainable biases are included as an additional column of $\mathbf{S}$, with a constant-one activation. Convolutional kernels are reshaped analogously, while vector parameters retain their coordinate scales. The resulting group transform $\boldsymbol{P}_{l,t}^{\mathrm{K}} = \rho_{l,t}\mathbf{D}_{l,t}^{\mathrm{K}}$ is fixed throughout each logical batch, including all microbatches; initialization follows Appendix B.

Implicit norms and per-layer clipping. For a linear layer, write $\mathbf{G}^{(i)} = (\mathbf{E}^{(i)})^\top \mathbf{A}^{(i)}$, where rows index positions and $\mathbf{E}^{(i)}$ contains output derivatives; the bias column is included when present. Set $\mathbf{A}'^{(i)} = \mathbf{A}^{(i)}\,\mathrm{diag}(\mathbf{r})$ and $\mathbf{E}'^{(i)} = \rho_{l,t}\mathbf{E}^{(i)}\,\mathrm{diag}(\mathbf{q})$. The transformed norm is computed as

$$(n_l^{(i)})^2 = \|(\mathbf{E}'^{(i)})^\top \mathbf{A}'^{(i)}\|_F^2 = \left\langle \mathbf{A}'^{(i)}(\mathbf{A}'^{(i)})^\top, \mathbf{E}'^{(i)}(\mathbf{E}'^{(i)})^\top \right\rangle_F.$$

Position pairs are processed in blocks, without materializing per-example weight gradients. Squared contributions are summed over all parameters in a clipping group. BK then uses $\alpha_l^{(i)} = \min\{1, C_l/n_l^{(i)}\}$ (one for zero norm) to accumulate $\sum_i \alpha_l^{(i)}(\mathbf{E}'^{(i)})^\top \mathbf{A}'^{(i)}$ directly. These operations are exact for the factorized transform.

Private release and updates. The same norms produce the slack vectors in equation 1. At the logical-batch boundary, we jointly privatize the accumulated transformed gradients and slack as in Algorithm 1, then invert the same transform:

$$\tilde{\boldsymbol{g}}_l = \frac{1}{B}(\boldsymbol{P}_{l,t}^{\mathrm{K}})^{-1}\left(\sum_{i\in\mathbb{B}_t} \alpha_l^{(i)} \boldsymbol{P}_{l,t}^{\mathrm{K}} \boldsymbol{g}_l^{(i)} + \boldsymbol{z}_l\right), \qquad \boldsymbol{z}_l \sim \mathcal{N}(\mathbf{0}, \sigma^2 C^2 \boldsymbol{I}_{d_l}). \tag{25}$$

The optimizer moments and private slack feedback determine the factors and layer scales for the next iteration. This implementation preserves the efficiency advantages of BK per-layer clipping: one backward pass, direct layer-wise aggregation, and no storage of the full batch of per-example parameter gradients. The measured overhead is reported in Appendix H.

H Computational Efficiency

Setup. We benchmark Flat clipping (BK), Per-layer clipping (BK), and SCP-Clip on one NVIDIA A100 (80 GB), using the same PyTorch/BK codebase and FP32/TF32 settings. Flat clipping uses optimized implicit all-layer BK; SCP-Clip includes exact weighted norms, preconditioner construction, noise transformation, and private scale updates. Input length is 256 for QNLI; CIFAR-100 images are 224×224. Timings cover complete optimizer updates, including all physical batches, and are averaged over 100 updates after 20 warm-up updates across three profiling runs.

Results. Table 6 shows that SCP-Clip retains 98.4–99.7% of the throughput of Per-layer clipping (BK), with 0.3–1.6% more time per optimizer update and 0.7–2.6% more peak memory. Compared with Flat clipping (BK), SCP-Clip reduces peak memory by 12.0–32.2% across all four models. These results indicate that SCP-Clip retains the memory benefits of Per-layer clipping with modest additional computational cost under the evaluated settings.

Table 6: Full fine-tuning efficiency. Memory is peak allocated GPU memory (GB); time is per complete optimizer update. B and b denote logical and physical batch sizes. Throughput is B divided by the reported time per update. Bold values indicate the best entry within each model.

Method	Peak memory (GB) ↓	Time/update (s) ↓	Throughput (examples/s) ↑
RoBERTa-base / QNLI ($\epsilon = 3$, $B = 4000$, $b = 40$)			
Flat clipping (BK)	15.8	19.84	201.6
Per-layer clipping (BK)	**11.9**	**19.60**	**204.1**
SCP-Clip	12.1	19.65	203.6
RoBERTa-large / QNLI ($\epsilon = 3$, $B = 4000$, $b = 40$)			
Flat clipping (BK)	34.3	50.64	79.0
Per-layer clipping (BK)	**30.0**	**50.12**	**79.8**
SCP-Clip	30.2	50.87	78.6
ViT-small / CIFAR-100 ($\epsilon = 2$, $B = 1000$, $b = 50$)			
Flat clipping (BK)	5.9	1.86	537.6
Per-layer clipping (BK)	**3.9**	**1.84**	**543.5**
SCP-Clip	4.0	1.87	534.8
ViT-large / CIFAR-100 ($\epsilon = 2$, $B = 1000$, $b = 50$)			
Flat clipping (BK)	25.5	10.03	99.7
Per-layer clipping (BK)	**21.6**	**9.95**	**100.5**
SCP-Clip	21.9	9.98	100.2

I Additional Comparisons with Closely Related Clipping Methods

AdaCliP and GeoClip already transform gradients before clipping and noise injection, and construct their transformations from previously privatized statistics (Pichapati et al., 2019; Gilani et al., 2025). Therefore, this comparison asks whether SCP-Clip offers additional utility over a layer-wise adaptation of an existing geometry-aware method, including when that method receives the same slack feedback, and over slack-based layer-wise adaptive thresholding without preconditioning. The comparison contains AdaCliP-L, AdaCliP-L + slack, SlaClip-L, GeoClip-L, GeoClip-L + slack, and SCP-Clip; the suffix L denotes a layer-wise adaptation rather than an unchanged reproduction of the original method. We use full fine-tuning of ViT-small on CIFAR-100 at the same setting as Appendix B.2.

AdaCliP-L. The adaptation retains AdaCliP's historical mean centering, coordinate scaling rule, and noise-corrected variance estimation. Let $\mathbf{m}_{l,t}$ be the historical private mean and $\mathbf{A}_{l,t} = \mathrm{diag}(1/\mathbf{b}_{l,t})$ the diagonal transform obtained from AdaCliP's scale vector. For each example,

$$\mathbf{z}_{l,t}^{(i)} = \rho_{l,t}\mathbf{A}_{l,t}\big(\mathbf{g}_{l,t}^{(i)} - \mathbf{m}_{l,t}\big), \qquad C_l = C_0/\sqrt{L}.$$

The layer-wise private gradient supplied to the shared optimizer is

$$\tilde{\mathbf{g}}_{l,t} = \mathbf{m}_{l,t} + \frac{1}{B}(\rho_{l,t}\mathbf{A}_{l,t})^{-1}\left[\sum_{i\in\mathcal{B}_t}\mathrm{Clip}(\mathbf{z}_{l,t}^{(i)}, C_l) + \boldsymbol{\xi}_{l,t}\right], \quad \boldsymbol{\xi}_{l,t} \sim \mathcal{N}(\mathbf{0}, \sigma^2 C^2 \mathbf{I}_{d_l}),$$

where $C = (\sum_l C_l^2)^{1/2}$. AdaCliP-L fixes $\rho_{l,t} = 1$.

AdaCliP-L + slack. This variant keeps the same AdaCliP geometry-construction rule and adds exactly the slack encoding and feedback controller used by SCP-Clip. Specifically, it encodes $(C_l - \|\mathbf{z}_{l,t}^{(i)}\|_2)_+$, jointly releases the encoded slack and clipped transformed gradients, and updates

$$\rho_{l,t+1} = \rho_{l,t}\exp\big[\eta_\rho\big(\hat{s}_{l,1} - \gamma_{l,t}\big)\big].$$

The initialization $\rho_{l,1} = 1$, slack dimension K, normalization, target rule $\gamma_{l,t}$, and step size $\eta_\rho = 0.2$ match Algorithm 1. Each method computes feedback from its own transformed norms.

SlaClip-L. We adapt the slack-based adaptive clipping mechanism of Zou et al. (2026) to the same layer-wise grouping. For each layer, SlaClip-L clips the original per-example gradients at an adaptive threshold $C_{l,t}$, encodes $(C_{l,t} - \|\mathbf{g}_{l,t}^{(i)}\|_2)_+$, and jointly privatizes the clipped gradients and encoded slack. The private slack feedback updates $C_{l,t}$ for the next iteration. Unlike SCP-Clip, this baseline adjusts thresholds directly and does not apply a preconditioning transform before clipping. It tests whether layer-wise slack-based threshold adaptation alone accounts for SCP-Clip's gains.

GeoClip-L and GeoClip-L + slack. GeoClip is retained as a closely related comparator, but GeoClip's covariance-based preconditioning incurs substantial computational and memory overhead when applied to large models. We therefore do not report results for GeoClip-L or GeoClip-L + slack in this evaluation. A dense layer-wise adaptation would require covariance/transform storage of order $\sum_l d_l^2$, where d_l is the number of trainable parameters in a clipping group. The paper's streaming rank-k alternative avoids dense storage, but still maintains a $d \times k$ basis and updates it by a singular value decomposition. Applied independently to all groups, its basis storage is of order $\sum_l d_l k_l$ and its basis-update cost is of order $\sum_l (d_l k_l^2 + k_l^3)$ (Gilani et al., 2025).

Table 7: Comparison with closely related geometry-aware and slack-based adaptive clipping methods on ViT-small/CIFAR-100. The reporting convention is top-1 test accuracy (%) averaged over three seeds.

	ViT-small, CIFAR-100		
Method	$\epsilon = 2$	$\epsilon = 4$	$\epsilon = 8$
AdaCliP-L	76.18	78.36	83.24
AdaCliP-L + slack	77.42	79.24	83.90
SlaClip-L	77.21	78.94	83.57
GeoClip-L	–	–	–
GeoClip-L + slack	–	–	–
SCP-Clip	**78.56**	**80.31**	**84.61**

As shown in Table 7, adding the same slack controller to AdaCliP-L increases accuracy by 1.24, 0.88, and 0.66 percentage points at $\epsilon = 2, 4, 8$, respectively. This shows that part of the improvement is also available by augmenting an existing geometric method with the inherited feedback mechanism. SCP-Clip nevertheless retains margins of 1.14, 1.07, and 0.71 percentage points over AdaCliP-L + slack, compared with 2.38, 1.95, and 1.37 points over AdaCliP-L. SlaClip-L achieves 77.21%, 78.94%, and 83.57% at $\epsilon = 2, 4, 8$, respectively; SCP-Clip exceeds these results by 1.35, 1.37, and 1.04 percentage points. Thus, SCP-Clip offers incremental utility over both the evaluated AdaCliP adaptations and slack-based layer-wise adaptive thresholding.