

FedAnchor: Differentially Private Federated LoRA via Weight Anchoring

Jie Xu¹ Haaris Mehmood¹ Rogier van Dalen² Karthikeyan Saravanan¹
Mete Ozay¹

¹Samsung R&D Institute UK (SRUK) ²Samsung AI Centre Cambridge
{jie1.xu, h.mehmood, r.vandalen, k1.saravanan, m.ozay}@samsung.com

Abstract

Federated learning (FL) with differential privacy (DP) allows several organizations to learn a shared model from user data that only one organization is allowed to see. However, guaranteeing DP requires adding noise to gradients that scales with update dimensionality. While low-rank adaptation (LoRA) is a popular method for parameter-efficient fine-tuning, it is inherently sensitive to DP noise due to the multiplication operator. This noise amplification leads to significant utility degradation, hindering practical deployment in enterprise settings. To overcome this, we propose FedAnchor. Every round, FedAnchor captures compressed information using one of the LoRA matrices A and B , and then decompresses this to the complete adapted weight matrix via singular value decomposition (SVD) at the server. This relieves the model from DP noise exacerbation while maintaining full learning capacity. Experiments across multiple benchmarks demonstrate that FedAnchor matches the strongest baseline in non-private settings and outperforms it by 4.38% under stringent DP constraints ($\epsilon = 3$). At scale, FedAnchor maintains communication efficiency, computation robustness, and utility under strict privacy budgets, meeting the demands for enterprise-level cross-silo deployments.

1 Introduction

Enterprises are increasingly adopting large language models (LLMs) for domain-specific tasks (Devlin et al., 2019; Brown et al., 2020), yet their deployment is constrained by the fragmentation of sensitive data across organizational silos (Lee et al., 2020; Zhao et al., 2023). Strict regulatory compliance (e.g., GDPR, CCPA) and internal policies often forbid the centralization of raw data, creating a critical barrier to collaborative model training. While federated learning (FL) has emerged as a privacy-preserving solution to enable cross-silo

collaboration, practical deployments face significant stability and efficiency challenges that hinder adoption in production environments.

Beyond privacy, communication and computational costs associated with adapting massive models remain significant barriers in cross-silo deployments (Kaplan et al., 2020). Full fine-tuning is often infeasible due to bandwidth constraints between subsidiaries, necessitating methods that minimize the optimization footprint (Li et al., 2020). Parameter-efficient fine-tuning (PEFT) offers a solution by freezing pre-trained weights and modifying only a fraction of the parameters. Among these, low-rank adaptation (LoRA) (Hu et al., 2022) has gained traction in FL contexts. It approximates weight updates using low-rank matrices, which significantly reduces communication overhead, making large model adaptation accessible to clients with limited network resources.

While FL mitigates data leakage risks, it lacks formal protection against gradient-based attacks (Zhu et al., 2019; Geiping et al., 2020). In enterprise environments, such vulnerabilities pose significant compliance risks, particularly given the memorization capacity of LLMs which can inadvertently expose sensitive information (Carlini et al., 2021). Therefore, integrating differential privacy (DP) (Dwork et al., 2006) is essential to satisfy rigorous regulatory standards and provide provable privacy guarantees. However, the standard mechanism, DP-SGD (Abadi et al., 2016), enforces privacy by clipping gradients and adding calibrated noise, creating a critical trade-off between privacy and model utility (Geyer et al., 2017). Notably, PEFT benefits DP-FL beyond communication efficiency. By reducing update size, it improves the signal-to-noise ratio against injected DP noise.

Yet, direct application of DP-SGD to LoRA training using the FL baseline algorithm FedAvg (McMahan et al., 2017) has been shown to substantially undermine model utility (Sun et al., 2024).

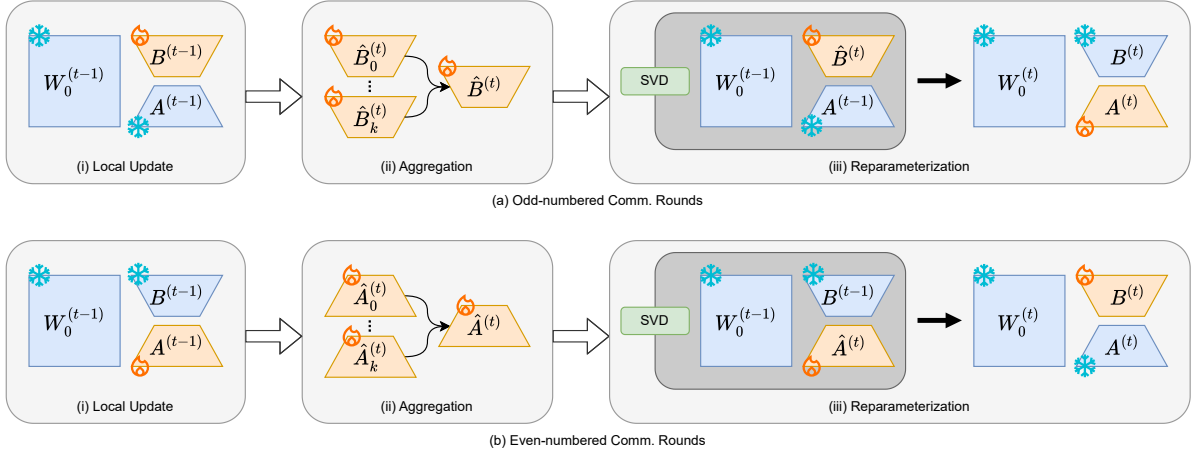


Figure 1: Workflow of the FedAnchor framework. The process comprises three stages: (i) **Local Update**: Clients alternately fine-tune LoRA matrix B (odd rounds) or A (even rounds), freezing the other matrix and base weights. (ii) **Aggregation**: Updated LoRA matrices are aggregated at the server. (iii) **Reparameterization**: SVD is applied to the combined base weights and aggregated LoRA product to re-initialize both the LoRA matrices and the base weights. This strategy anchors updates to the pre-trained manifold, preserving full model adaptability without amplifying DP noise under privacy constraints. For more information, refer to Section 3.

The two LoRA matrices A and B are aggregated separately at the server side, leading to an inconsistency with regard to the ideal aggregation as shown in Equation 4. Additionally, LoRA training with DP-SGD introduces an additional noise term as in Equation 5, further compromising performance.

FFA-LoRA (Sun et al., 2024) attempted to mitigate these by freezing the randomly initialized matrices A and only fine-tuning the zero-initialized matrices B . However, this approach inadvertently constrains the expressive capacity of the LoRA adapter by preventing matrices A from adapting throughout training. FedSVD (Lee et al., 2025) addressed this by applying SVD to the product of the updated B and previous A , refactoring both to allow matrix A to adaptively capture the principal directions of the aggregate updates rather than remaining frozen. However, LoRA weights accumulate noise at each round of training. Applying SVD to this matrix product produces singular vectors that are heavily perturbed by noise artifacts, leading to lower model utility. Although FedSVD allows A to adapt, its reliance on the noisy BA product limits training stability, particularly in high-noise regimes, ultimately hindering reliable deployment in performance-critical enterprise settings.

To this end, we propose our solution named FedAnchor to address these limitations. Instead of applying SVD directly to the product of updated B and previous A to propagate updates, we perform global reparameterization on the composition of the base weights and the LoRA product, a primi-

tive introduced outside the private setting by PiSSA (Meng et al., 2024) and FRLoRA (Yan et al., 2025). By including the base weights, which carry high-magnitude semantic signal in the SVD, we leverage them as a stable prior, effectively decompressing the updates to the base and LoRA weights at every round. Because the residual is folded back into the base weights, the accumulated update is not confined to a fixed rank- r subspace, even though each round’s communication remains low-rank. Moreover, as inspired by RoLoRA (Chen et al., 2025), we alternate between training A and B instead of keeping A always frozen, allowing the pair to explore the joint optimization space more thoroughly. We summarize our contributions as follows:

- We propose FedAnchor, a framework anchoring LoRA updates via model-centric SVD on the sum of base weights and LoRA product, accumulating each round’s update into the base weights at full rank while keeping per-round updates low-rank and improving convergence under DP noise.
- We integrate an alternating optimization strategy for A and B matrices, synergizing this approach with our SVD-based re-initialization to ensure both matrices are optimized via gradient descent, enabling thorough exploration of the joint optimization space.
- We demonstrate accuracy improvements over state-of-the-art baselines across multiple

benchmarks, validating the method’s effectiveness in differentially private federated learning settings.

- We provide a comprehensive deployment feasibility analysis, confirming FedAnchor’s suitability for cross-silo enterprise deployments.

2 Related Work

2.1 Federated learning

At the start of each round t , the server broadcasts the global model $W^{(t-1)}$ and selects a random subset of clients $\mathcal{C}^{(t)} \subseteq \mathcal{K}$. Each client $k \in \mathcal{C}^{(t)}$ performs local training to obtain an updated model $W_k^{(t)}$, then computes the model update $\Delta_k^{(t)} = W_k^{(t)} - W^{(t-1)}$ and transmits it to the server. In baseline methods such as FedAvg (McMahan et al., 2017), the server aggregates these updates by

$$W^{(t)} = W^{(t-1)} + \frac{1}{|\mathcal{C}^{(t)}|} \sum_{k \in \mathcal{C}^{(t)}} \Delta_k^{(t)}, \quad (1)$$

before proceeding to the next round.

2.2 Differentially private federated learning

Although user data is never directly shared with other parties in FL, model updates transmitted to the server still expose private information regarding local training data. Differential privacy (DP) (Dwork et al., 2006; Dwork and Roth, 2014) mitigates this threat by providing formal privacy guarantees that bound the influence of any individual data point on the model output.

Definition 1 (Differential Privacy). *A randomized mechanism $\mathcal{M} : \mathcal{D} \rightarrow \mathcal{R}$ satisfies (ϵ, δ) -DP if for any pair of adjacent datasets $\mathcal{D}$ and $\mathcal{D}'$ and any subset of outputs $\mathcal{S} \subseteq \mathcal{R}$, we have:*

$$\Pr[\mathcal{M}(\mathcal{D}) \in \mathcal{S}] \leq \exp(\epsilon) \cdot \Pr[\mathcal{M}(\mathcal{D}') \in \mathcal{S}] + \delta \quad (2)$$

where (ϵ, δ) is known as the privacy budget. More details on DP-FL are provided in Appendix A.

2.3 DP-FL with PEFT

Full fine-tuning incurs substantial communication and computational overhead in FL deployments. Parameter-efficient fine-tuning (PEFT) addresses these challenges by freezing pre-trained weights and optimizing only a small subset of parameters. Naturally, this also helps in DP-FL as the reduced number of trainable parameters leads to higher signal-to-noise ratio and less noisy training. Recent

works have focused on utilizing LoRA in DP-FL due to its proven effectiveness in this setting (Yu et al., 2022). LoRA freezes the pre-trained weights and adds trainable rank decomposition matrices into each transformer layer. Let $W_0 \in \mathbb{R}^{m \times n}$ be a pre-trained weight matrix. LoRA adds a low-rank term BA with rank r by:

$$W_0 + \Delta W = W_0 + BA, \quad (3)$$

where $B \in \mathbb{R}^{m \times r}$ is an up-projection and $A \in \mathbb{R}^{r \times n}$ is a down-projection. Here, A is randomly initialized and B is initialized to all zeros. BA is therefore zero at the start of training.

2.4 Freezing LoRA matrix A in DP-FL

A few intrinsic problems with applying LoRA in DP-FL have been studied by Sun et al. (2024). The first problem comes from the FL side. Clients compute the loss for back propagation on the sum of the pre-trained weights and the product of LoRA matrices A and B . The optimization is subsequently performed on A and B jointly. However, FL algorithms such as FedAvg (McMahan et al., 2017) perform server-side aggregation by averaging the two matrices separately, which deviates from the ideal aggregation as follows:

$$\underbrace{W_0 + \frac{1}{k} \sum_{i=1}^k B_i \times \frac{1}{k} \sum_{i=1}^k A_i}_{\text{Parameters after aggregation with LoRA + FedAvg}} \neq \underbrace{W_0 + \frac{1}{k} \sum_{i=1}^k (B_i A_i)}_{\text{Ideal parameters following aggregation}} \quad (4)$$

The second problem is caused by using DP-SGD to optimize LoRA in a privacy-preserving manner. An additional noise term $\xi_B \xi_A$ is introduced which amplifies the DP noise and harms model utility:

$$\begin{aligned} & W_0 + (B + \xi_B)(A + \xi_A) \\ &= W_0 + BA + B\xi_A + \xi_B A + \xi_B \xi_A. \end{aligned} \quad (5)$$

2.5 SVD-based reparameterization

Re-deriving the LoRA pair from the principal singular components of a weight matrix was introduced by PiSSA (Meng et al., 2024), which initializes A and B from the top- r components of W_0 and freezes the residual. FRLoRA (Yan et al., 2025) brought this into federated learning: it merges the aggregated low-rank product into the global base

weights every round and re-initializes both LoRA matrices from principal singular components, so that the accumulated global update is not confined to a rank- r subspace. FedSVD (Lee et al., 2025) instead decomposes the aggregated LoRA product alone. FedAnchor inherits the reparameterization primitive from this line of work; what we claim is its combination with alternating single-matrix training, which is what makes per-round current-weight SVD viable under DP noise (Section 4.2).

3 Method: FedAnchor

FedSVD improves on FFA-LoRA by applying SVD on the product of the updated B matrix after aggregation and the previous A matrix:

$$U^{(t)}\Sigma^{(t)}V^{(t)\top} \leftarrow \text{SVD}(\hat{B}^{(t)}A^{(t-1)}), \quad (6)$$

$$B^{(t)} \leftarrow U^{(t)}[:, :r]\Sigma^{(t)}[:, :r], \quad (7)$$

$$A^{(t)} \leftarrow V^{(t)\top}[:, r:]. \quad (8)$$

This allows A to adapt to the principal directions of the model updates instead of being frozen throughout the training process. However, DP introduces noise into BA at each training round. Performing SVD directly on this noisy matrix yields unstable vectors dominated by noise artifacts, poorly reflecting the principal directions.

3.1 Model-centric decomposition

To address this, we propose a new DP-FL method named FedAnchor. Instead of updating A via SVD of $\hat{B}^{(t)}A^{(t-1)}$, we perform SVD on the sum of the base weights and the updated LoRA product. Specifically, after applying the aggregated updates to obtain $\hat{B}^{(t)}$, we re-initialize $B^{(t)}$, $A^{(t)}$ and $W^{(t)}$ as follows:

$$\tilde{W}^{(t)} = W^{(t-1)} + \hat{B}^{(t)}A^{(t-1)}, \quad (9)$$

$$U^{(t)}\Sigma^{(t)}V^{(t)\top} \leftarrow \text{SVD}(\tilde{W}^{(t)}), \quad (10)$$

$$B^{(t)} \leftarrow U^{(t)}[:, :r]\sqrt{\Sigma^{(t)}[:, :r]}, \quad (11)$$

$$A^{(t)} \leftarrow \sqrt{\Sigma^{(t)}[:, :r]}V^{(t)\top}[:, r:], \quad (12)$$

$$W^{(t)} \leftarrow W^{(t-1)} + \hat{B}^{(t)}A^{(t-1)} - B^{(t)}A^{(t)}. \quad (13)$$

where $W^{(0)}$ is the pre-trained weights, $B^{(0)}$ is initialized as zeros, $A^{(0)}$ is initialized with Kaiming uniform, $M[:, :r]$ and $M[:, r:]$ denote the first r columns and rows of the matrix M , respectively.

Compared to FedSVD’s LoRA-centric decomposition, our method performs model-centric decomposition. The model base weights constitute

the vast majority of parameters and carry the high-magnitude semantic signal. While LoRA weights capture task-specific adaptations, they are constrained by low-rank approximations and can be obscured by DP noise. By including the base weights in the SVD, we leverage them as a stable structural prior. Under FedSVD the cumulative update stays inside a rank- r subspace through the training process, whereas our method folds the residual into the base weights, so the accumulated update is full-rank. Anchoring in the base weights additionally means the retained top- r directions are set by the base spectrum rather than by the DP noise in the LoRA product. We report a singular-value comparison in Appendix E.

Moreover, existing methods (e.g., FedAvg, FFA-LoRA, FedSVD) capture and store new information using either B , A or both. This limits the model’s learning capacity over consecutive rounds, particularly when DP noise corrupts the gradient signals within this constrained subspace. By effectively decompressing updates to both the base and LoRA weights at every round, our method accumulates updates in a higher-rank space, following the capacity argument of FRLoRA (Yan et al., 2025). This prevents the low-rank approximation from becoming a bottleneck in the training procedure.

3.2 Block coordinate descent optimization

FedSVD treats A and B asymmetrically. B is optimized via gradients, while A is optimized via server-side SVD. This decoupling means that the optimization of A is not directly driven by the local loss gradients during training. Our method implements a form of Block Coordinate Descent directly on the client side by alternating between training B and A . This enables both B and A to be optimized by gradient descent during the training process instead of having one matrix, such as A , relying solely on SVD. Consequently, this alternating optimization allows the pair to explore the joint optimization manifold more thoroughly. Notably, while RoLoRA also alternates fine-tuning between B and A , it does so primarily to ensure both matrices are updated over successive rounds. Conversely, our method re-initializes both matrices each round using SVD on the combined BA and base weights, thereby enhancing the optimization process beyond simple round-robin updates. Algorithm 1 in Appendix F gives the full procedure.

4 Experiments

4.1 Experimental setup

Datasets and models. To validate the effectiveness of our method, we conduct experiments on five natural language understanding (NLU) tasks: SNLI (Bowman et al., 2015), MNLI (matched and mismatched) (Williams et al., 2018), QNLI (Wang et al., 2018), and QQP (Sharma et al., 2019). As test sets are unavailable for all tasks except SNLI, we select hyperparameters and evaluate model performance on the validation sets. SAMSum (Section 5.4) has a held-out test split and provides a properly separated evaluation. We use pre-trained RoBERTa-base (125M parameters) as the primary backbone, and additionally evaluate RoBERTa-large, TinyLlama-1.1B, T5-3B and T5-11B in Section 5.4. Details on implementation and data distribution are provided in Appendix B and C.

Baselines. We compare our method against state-of-the-art baselines, including FedAvg (McMahan et al., 2017), FFA-LoRA (Sun et al., 2024), RoLoRA (Chen et al., 2025), FedEx-LoRA (Singhal et al., 2025) and FedSVD (Lee et al., 2025).

4.2 Results

Non-private FL. We begin by evaluating FedAnchor in a non-private setting ($\epsilon = \infty$). Table 1 shows that FedAnchor attains the highest accuracy on QQP and QNLI, while FedAvg remains strongest on SNLI and MNLI. In contrast, FFA-LoRA achieves the worst average performance due to having a fixed A and therefore having fewer trainable parameters in total. Our method solves this problem by re-initializing both $B^{(t)}$ and $A^{(t)}$ at the end of each communication round. Consequently, FedAnchor outperforms recent baselines by 3.28% on average and the strongest baseline by 0.57% in test accuracy, which we read as parity with FedAvg rather than an improvement.

DP-FL. The effectiveness of FedAnchor is clearly substantiated under DP constraints ($\epsilon \in \{3, 6\}$, $\delta = 10^{-5}$). As shown in Table 1, FedAnchor preserves its accuracy gains across both privacy budgets, improving on the strongest baseline by 4.38% at $\epsilon = 3$ and 2.37% at $\epsilon = 6$. The average gains are 5.30% and 4.25%. These results are driven by two factors: First, the model-centric decomposition re-initializes LoRA matrices using SVD on the combined BA and base weights, which improves convergence and prioritizes high-signal information. Second, the alternating fine-tuning of B and

A ensures both matrices are optimized via gradient descent, allowing the model to traverse the joint optimization manifold more effectively.

Comparison with SVD-reparameterization baselines. Table 2 compares FedAnchor against the closest structural predecessors at $\epsilon = 3$. FRLoRA re-initializes the local LoRA matrices each round from $W^{(0)}$ in the manner of PiSSA, while moving the LoRA updates into the base weights. Its v3 variant re-derives the local LoRA matrices from the current model weights and is therefore the closest predecessor to our per-round current-weight SVD. FedAvg+PiSSA shares the initialization but keeps the base weights frozen. The comparison isolates one factor at a time rather than being a full factorial: FRLoRA to FRLoRA-v3 isolates static versus adaptive re-derivation, and FRLoRA-v3 to FedAnchor isolates alternating single-matrix training. Training a single matrix per round is what carries the gain here. It reduces the noise entering the product and removes the quadratic noise term of Equation 5. It also makes aggregation exact, so no aggregation error is propagated into the base weights. A variant that trains only B without alternating reaches 77.02 ± 3.27 on QNLI, against 77.70 ± 3.47 for the full method. FRLoRA targets data heterogeneity rather than privacy, so to our knowledge no DP evaluation of this family exists.

5 Deployment Feasibility

To assess the practical viability of FedAnchor, we analyze system performance under real-world deployment constraints. Industrial adoption requires robustness against data heterogeneity, scalability across client counts, and efficient communication. We evaluate stability under varying non-IID distributions and client scales. We also quantify communication overhead via SVD-based compression. Finally, we compare decomposition methods to balance re-initialization latency and utility. These experiments characterize FedAnchor along the axes that determine deployability in enterprise-grade federated learning systems. We primarily use the QNLI dataset in this section.

5.1 Scalability and robustness

Scalability across client counts. To reflect the variability in participating entities found in cross-silo deployments, we evaluate FedAnchor’s scalability with total client counts $K \in \{6, 9, 12\}$ while keeping the participation count fixed at $K' = 3$

Table 1: Experimental results on five NLU datasets **without DP** ($\epsilon = \infty$) and **with DP** ($\epsilon \in \{3, 6\}$, $\delta = 10^{-5}$).

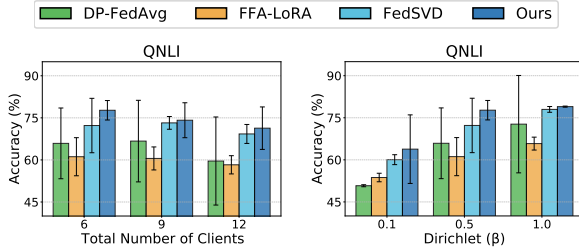
Privacy budget	Method	SNLI	MNLI		QQP	QNLI	Average
			Matched	Mismatched			
$\epsilon = \infty$	FedAvg	85.56 $\pm$ 1.99	78.11 $\pm$ 1.09	79.65 $\pm$ 1.23	77.91 $\pm$ 1.72	73.33 $\pm$ 17.80	78.91
	FFA-LoRA	80.03 $\pm$ 4.50	76.61 $\pm$ 0.97	77.36 $\pm$ 1.80	73.77 $\pm$ 9.19	62.89 $\pm$ 5.30	74.13
	RoLoRA	82.57 $\pm$ 2.50	75.53 $\pm$ 6.72	78.56 $\pm$ 0.36	73.28 $\pm$ 8.33	62.00 $\pm$ 14.87	74.38
	FedEx-LoRA	<u>84.06</u> $\pm$ 0.97	76.44 $\pm$ 7.68	<u>79.13</u> $\pm$ 0.80	76.54 $\pm$ 5.37	71.85 $\pm$ 18.48	77.60
	FedSVD	81.09 $\pm$ 0.68	73.47 $\pm$ 0.68	68.96 $\pm$ 3.68	77.13 $\pm$ 3.05	<u>79.10</u> $\pm$ 0.43	75.95
	FedAnchor (ours)	83.48 $\pm$ 0.72	<u>78.10</u> $\pm$ 0.54	77.32 $\pm$ 0.49	78.25 $\pm$ 2.44	80.26 $\pm$ 1.52	79.48
$\epsilon = 3$	FedAvg	74.39 $\pm$ 8.42	69.54 $\pm$ 3.64	66.62 $\pm$ 5.06	74.64 $\pm$ 3.84	65.90 $\pm$ 12.61	70.21
	FFA-LoRA	<u>77.72</u> $\pm$ 2.42	72.18 $\pm$ 1.54	<u>70.79</u> $\pm$ 3.20	70.69 $\pm$ 7.11	61.14 $\pm$ 6.79	<u>70.50</u>
	RoLoRA	72.88 $\pm$ 15.37	67.54 $\pm$ 5.77	69.38 $\pm$ 3.77	71.18 $\pm$ 4.48	59.60 $\pm$ 14.81	68.11
	FedEx-LoRA	77.48 $\pm$ 4.45	69.44 $\pm$ 3.41	64.39 $\pm$ 10.35	70.22 $\pm$ 8.37	61.50 $\pm$ 17.15	68.60
	FedSVD	74.82 $\pm$ 2.55	64.54 $\pm$ 1.14	65.72 $\pm$ 3.04	<u>74.96</u> $\pm$ 0.58	<u>72.26</u> $\pm$ 9.70	70.46
	FedAnchor (ours)	77.95 $\pm$ 2.76	<u>71.29</u> $\pm$ 2.94	71.43 $\pm$ 0.28	76.05 $\pm$ 2.84	77.70 $\pm$ 3.47	74.88
$\epsilon = 6$	FedAvg	76.11 $\pm$ 3.93	<u>72.94</u> $\pm$ 1.09	73.04 $\pm$ 5.07	76.48 $\pm$ 0.72	69.13 $\pm$ 16.14	73.54
	FFA-LoRA	78.39 $\pm$ 3.22	72.88 $\pm$ 0.60	72.59 $\pm$ 5.46	73.05 $\pm$ 5.58	66.22 $\pm$ 11.97	72.62
	RoLoRA	78.03 $\pm$ 5.39	68.73 $\pm$ 5.15	67.42 $\pm$ 10.20	72.55 $\pm$ 1.44	62.10 $\pm$ 13.77	69.76
	FedEx-LoRA	<u>79.49</u> $\pm$ 3.48	73.01 $\pm$ 0.87	<u>73.41</u> $\pm$ 3.89	75.76 $\pm$ 3.23	69.14 $\pm$ 15.61	<u>74.16</u>
	FedSVD	75.62 $\pm$ 1.30	64.93 $\pm$ 1.18	67.56 $\pm$ 0.20	75.15 $\pm$ 4.43	<u>73.33</u> $\pm$ 6.33	71.31
	FedAnchor (ours)	80.54 $\pm$ 1.14	72.68 $\pm$ 1.62	74.55 $\pm$ 0.96	<u>76.32</u> $\pm$ 2.55	78.56 $\pm$ 3.15	76.53

Table 2: Comparison with SVD-reparameterization methods at $\epsilon = 3$, $\delta = 10^{-5}$.

Method	SNLI	QQP	QNLI
FedAvg	74.39 $\pm$ 8.42	74.64 $\pm$ 3.84	65.90 $\pm$ 12.61
FedAvg+PiSSA	56.61 $\pm$ 20.50	72.02 $\pm$ 3.46	58.48 $\pm$ 7.26
FRLoRA	36.44 $\pm$ 1.09	63.29 $\pm$ 0.07	52.98 $\pm$ 2.35
FRLoRA-v3	60.63 $\pm$ 15.82	75.19 $\pm$ 2.66	67.88 $\pm$ 14.74
FedAnchor (ours)	77.95 $\pm$ 2.76	76.05 $\pm$ 2.84	77.70 $\pm$ 3.47

Table 3: Results of FedAnchor on QNLI with full and compressed updates to the base weights.

Method	$\epsilon = \infty$	$\epsilon = 3$	$\epsilon = 6$
Full	80.26 $\pm$ 1.52	77.70 $\pm$ 3.47	78.56 $\pm$ 3.15
Compressed	79.08 $\pm$ 3.64	73.98 $\pm$ 7.19	77.04 $\pm$ 2.99



(a) Total number of clients

(b) Data heterogeneity

Figure 2: (a) Results of different total numbers of clients. (b) Results of different levels of data heterogeneity.

per round. As shown in Figure 2a, FedAnchor maintains superior accuracy across all scales. This resilience indicates that FedAnchor remains effective even as the federation scales to accommodate more subsidiaries or regional offices.

Robustness to data heterogeneity. Real-world data is rarely uniformly distributed across silos. We evaluate robustness using Dirichlet distributions with $\beta \in \{0.1, 0.5, 1.0\}$, where lower β represents higher non-IIDness. Figure 2b shows FedAnchor consistently outperforms baselines, particularly un-

der high heterogeneity, indicating it can learn effective representations despite severe data skew across silos. This stability under varying distribution conditions is critical for practical enterprise adoption where data availability varies by region.

5.2 Optimizing communication overhead

In cross-silo deployments, network bandwidth is often a critical constraint, particularly for client-to-server upload. Since FedAnchor trains only one of A and B per round like FFA-LoRA and FedSVD, it effectively halves the upload cost. However, updating the base weights increases the download overhead from $2 \cdot r \cdot h \cdot L$ to $h \cdot L \cdot (h + 2 \cdot r)$, where h denotes the hidden dimension of the model, r the low-rank dimension, and L the number of transformer layers. To mitigate this, we apply SVD-based compression to the base weight updates, significantly reducing the data download overhead to just $4 \cdot r \cdot h \cdot L$. Table 3 demonstrates that this asymmetric optimization preserves model utility without significant degradation, except under stringent DP constraints ($\epsilon = 3$) where gradients are inherently noisy. The degradation is largest at $\epsilon = 3$ on the smallest model (3.72% on QNLI

Table 4: Results of FedAnchor on QNLI with full-rank SVD, low-rank SVD and QR decomposition at $\epsilon = 3$.

Method	Accuracy (%)	Decomp. time (s)
SVD (niter= ∞)	77.70 $\pm$ 3.47	1.85
SVD (niter=8)	77.41 $\pm$ 3.38	0.28
QR	75.44 $\pm$ 4.37	0.19

Table 5: Results on QNLI with RoBERTa-large and TinyLlama-1.1B.

Model	Method	$\epsilon = 3$
RoBERTa-large	FedAvg	73.56 $\pm$ 15.32
	FFA-LoRA	73.44 $\pm$ 12.96
	FedSVD	77.57 $\pm$ 2.06
	FedAnchor (ours)	77.79 $\pm$ 2.36
TinyLlama-1.1B	FedAvg	77.85 $\pm$ 14.21
	FFA-LoRA	83.38 $\pm$ 0.77
	FedSVD	85.06 $\pm$ 0.88
	FedAnchor (ours)	85.42 $\pm$ 1.01

with RoBERTa-base) and shrinks with scale (2.05% with RoBERTa-large, 1.62% with TinyLlama-1.1B, and 0.73/1.12 ROUGE-L on SAMSum with T5-3B/T5-11B at $\epsilon = 1$). That is, it is smallest in the regime where compression is actually needed.

5.3 Computational trade-offs

We evaluate re-initialization efficiency by comparing full SVD, low-rank SVD and QR decomposition. As shown in Table 4, while full SVD achieves the highest accuracy, it incurs latency that may be prohibitive for large-scale architectures. Low-rank SVD offers a superior trade-off, retaining near-optimal accuracy while reducing latency by an order of magnitude. In contrast, QR decomposition provides similar speed but sacrifices more accuracy. For latency-constrained deployments, low-rank SVD is recommended as it captures the majority of the semantic signal without the computational overhead of full SVD, validating FedAnchor’s flexibility for practical industrial applications.

5.4 Model size generalization

To assess scalability across model sizes, we evaluate FedAnchor on RoBERTa-large and TinyLlama-1.1B (Table 5) and, on a different task family, on abstractive summarization with T5-3B and T5-11B on SAMSum (Table 6). FedAnchor is best on every model, at $\epsilon = 3$ on QNLI and $\epsilon = 1$ on SAMSum, and the compressed variant still beats every baseline at both T5 scales, so the base-weight download can be reduced without giving up the gain at the sizes where that download actually matters. Un-

Table 6: Results on the SAMSum dataset with T5-3B and T5-11B at $\epsilon = 1$. Scores are ROUGE-L.

Model	Method	$\epsilon = 1$
T5-3B	FedAvg	27.02 $\pm$ 0.34
	FFA-LoRA	25.07 $\pm$ 0.16
	FedSVD	26.20 $\pm$ 0.38
	FedAnchor (Full)	27.89 $\pm$ 0.23
	FedAnchor (Compressed)	<u>27.16</u> $\pm$ 0.63
T5-11B	FedAvg	26.16 $\pm$ 2.06
	FFA-LoRA	25.91 $\pm$ 1.14
	FedSVD	26.90 $\pm$ 0.35
	FedAnchor (Full)	29.16 $\pm$ 0.42
	FedAnchor (Compressed)	<u>28.04</u> $\pm$ 0.49

Table 7: Results on QNLI with LoRA rank $r \in \{4, 8, 16\}$ at $\epsilon = 3$.

Method	$r = 4$	$r = 8$	$r = 16$
FedAvg	67.04 $\pm$ 12.65	65.90 $\pm$ 12.61	68.08 $\pm$ 15.66
FFA-LoRA	58.00 $\pm$ 8.19	61.14 $\pm$ 6.79	56.34 $\pm$ 1.52
FedSVD	74.02 $\pm$ 2.37	72.26 $\pm$ 9.70	72.48 $\pm$ 5.35
FedAnchor (ours)	74.15 $\pm$ 5.96	77.70 $\pm$ 3.47	74.83 $\pm$ 5.84

like the NLU tasks, SAMSum has a held-out test split, so these numbers are tuned and evaluated on separate splits.

5.5 LoRA rank generalization

We vary the LoRA rank $r \in \{4, 8, 16\}$ on QNLI at $\epsilon = 3$. Table 7 shows FedAnchor is best at every rank, so the benefit is not an artifact of the $r = 8$ setting used elsewhere.

6 Conclusion

In this paper, we introduced FedAnchor, a framework that improves the convergence of LoRA training in DP-FL by anchoring SVD decomposition to pre-trained weights. Our method effectively mitigates noise amplification while maintaining model utility under strict privacy budgets. Extensive experiments on standard benchmarks demonstrate accuracy improvements over state-of-the-art baselines. Furthermore, our deployment analysis confirms robustness in scalability, communication efficiency, and computational trade-offs. These results indicate that FedAnchor is a practical candidate for cross-silo enterprise environments.

Limitations

While our current evaluation focuses on NLP benchmarks, the FedAnchor framework is architecture-agnostic and generalizable to other modalities. Our silos are also simulated and

our tasks are public benchmarks rather than a production workload, so Section 5 should be read as system-level measurement of the method’s costs rather than a report from a fielded deployment. In the future, we plan to extend our investigation to the vision and audio domains.

Regarding communication efficiency, with the SVD-based compression strategy we studied, FedAnchor incurs approximately 2x download overhead compared to FedAvg due to base weight updates. Compression reduces the per-round download from approximately 1.2 GB to 23.6 MB at 3B parameters, and from 4.8 GB to 80.2 MB at 11B. Given typical bandwidth asymmetries where download rates exceed upload rates significantly, combined with payloads of this size, this cost remains negligible on standard enterprise networks.

Finally, while per-round SVD computation introduces server-side overhead, our study shows truncated SVD reduces latency by an order of magnitude. Since decomposition is performed centrally, it can be parallelized across multiple GPUs, ensuring negligible impact on overall training latency.

References

- Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.
- Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, and 1 others. 2021. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pages 2633–2650.
- Shuangyi Chen, Yuanxin Guo, Yue Ju, Hardik Dalal, Zhongwen Zhu, and Ashish Khisti. 2025. Robust federated finetuning of LLMs via alternating optimization of LoRA. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.
- Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer.
- Cynthia Dwork and Aaron Roth. 2014. The algorithmic foundations of differential privacy. *Foundations and trends® in theoretical computer science*, 9(3-4):211–407.
- Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. 2020. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in neural information processing systems*, 33:16937–16947.
- Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*.
- Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. 2019. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.
- Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.
- Seanie Lee, Sangwoo Park, Dong Bok Lee, Dominik Wagner, Haebin Seong, Tobias Bocklet, Juho Lee, and Sung Ju Hwang. 2025. FedSVD: Adaptive orthogonalization for private federated learning with LoRA. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450.

- Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282.
- Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 121038–121072.
- Lakshay Sharma, Laura Graesser, Nikita Nangia, and Utku Evci. 2019. Natural language understanding with the quora question pairs dataset. *arXiv preprint arXiv:1907.01041*.
- Raghav Singhal, Kaustubh Ponkshe, and Praneeth Vepakomma. 2025. FedEx-LoRA: Exact aggregation for federated and efficient fine-tuning of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1316–1336.
- Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. 2024. Improving LoRA in privacy-preserving federated learning. In *The Twelfth International Conference on Learning Representations*.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.
- Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.
- Yunlu Yan, Chun-Mei Feng, Wangmeng Zuo, Rick Siow Mong Goh, Yong Liu, and Lei Zhu. 2025. Federated residual low-rank adaptation of large language models. In *International Conference on Learning Representations*.
- Ashkan Yousefpour, Igor Shilov, Alexandre Sablayrolles, Davide Testuggine, Karthik Prasad, Mani Malek, John Nguyen, Sayan Ghosh, Akash Bharadwaj, Jessica Zhao, Graham Cormode, and Ilya Mironov. 2021. Opacus: User-friendly differential privacy library in pytorch. In *NeurIPS 2021 Workshop Privacy in Machine Learning*.
- 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. 2022. Differentially private fine-tuning of language models. In *International Conference on Learning Representations*.
- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, and 1 others. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*, pages 1–124.
- Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. *Advances in neural information processing systems*, 32.

A Federated Learning with DP-SGD

Differential privacy (DP) provides a rigorous guarantee that the presence or absence of any single data record has a negligible impact on the model’s output distribution. Consequently, this indistinguishability limits the amount of information an adversary can infer about specific training samples. This is also known as per-sample privacy. The strength of this privacy protection is parameterized by (ϵ, δ) , where smaller values indicate stricter privacy guarantees.

To apply this protection in deep learning, differentially private stochastic gradient descent (DP-SGD) is commonly employed (Abadi et al., 2016). Unlike standard SGD, DP-SGD privatizes gradients before updating the model weights. This process introduces two additional steps per iteration: gradient clipping and noise injection. First, per-sample gradients are clipped to a ℓ_2 norm bound C to constrain their influence. Second, Gaussian noise $z \sim \mathcal{N}(0, C^2 \sigma^2 I)$ is added to the aggregated clipped gradients within a batch $\mathcal{B}$, where I denotes the identity matrix. The privatized batch gradient is then computed as $\bar{g} = (\sum_{i \in \mathcal{B}} \text{Clip}(\nabla f_i, C) + z) / |\mathcal{B}|$. After local training, clients compute the model update and transmit it to the server, which aggregates these updates according to Equation 1. The noise scale factor σ is determined by each client using privacy composition rules (Abadi et al., 2016), calculated based on the target privacy budget (ϵ, δ) , the total number of iterations T , and the sampling rate $q = |\mathcal{B}|/N$, where N denotes the local dataset size.

In federated learning, the level of privacy assurance hinges on whether the central server is assumed to be trusted. In the central DP setting, clients transmit unmodified updates, with privacy mechanisms enforced solely during the global aggregation phase. Conversely, the local DP paradigm assumes an untrusted server, requiring each participant to privatize their updates prior to transmission. Our approach follows this more rigorous local DP model: we implement DP at the

client level, ensuring that all shared model updates are already protected. This ensures compliance with data sovereignty laws, as no raw or sensitive intermediate data leaves the organization, and the final model inherits these guarantees through post-processing invariance (Dwork and Roth, 2014).

B Implementation Details

For LoRA, we configure the rank r and scaling factor α to 8, with a dropout rate of 0.05. We apply LoRA to the query and value projection matrices of each attention layer. We run each experiment for 100 communication rounds. In each round, sampled clients perform parameter updates using DP-SGD with a batch size of 200. The number of local steps τ is set to 10 for QQP and QNLI, and to 30 for SNLI and MNLI. For each experiment, we report the best result by running a grid search over learning rate $\eta \in \{0.2, 0.5\}$ and clipping threshold $C \in \{2.0, 5.0\}$. Following FFA-LoRA (Sun et al., 2024) and FedSVD (Lee et al., 2025), we evaluate privacy budgets $\epsilon \in \{3, 6\}$ on the NLU tasks, and we use the stricter $\epsilon = 1$ on SAMSum. We use $\delta = 10^{-5}$ throughout. The noise multiplier σ is computed using the Opacus library (Yousefpour et al., 2021). We report accuracy on the NLU tasks and ROUGE-L on SAMSum. Full privacy accounting is given in Appendix D.

C Data Distribution

Following Hsu et al. (2019), we simulate non-Independent and Identically Distributed (non-IID) client data by partitioning the dataset according to a Dirichlet distribution $\text{Dir}(\beta)$ with $\beta = 0.5$. Consistent with FFA-LoRA (Sun et al., 2024) and FedSVD (Lee et al., 2025), we simulate a total of six clients. We sample three clients uniformly at random in each communication round across all experiments. These settings are maintained across all experiments except in Section 5.1.

D Privacy Accounting

We adopt local DP. Each client privatizes with DP-SGD before transmission. Noise is added to the gradient rather than to the update. Gaussian noise is added to the sum of clipped per-sample gradients within a batch of size $b = 200$, which is then divided by b . The SVD is applied at the server after aggregation. The active LoRA matrix alternates, B in odd rounds and A in even rounds. Aggregation, SVD, re-initialization, Equations 9–13 and

Table 8: Privacy accounting. $b = 200$, $R = 100$, $\delta = 10^{-5}$ throughout.

Dataset	N_k	q	τ	T	$\sigma_{\epsilon=3}$	$\sigma_{\epsilon=6}$
SNLI	91,692	0.0021	30	1500	0.59	0.48
MNLI	65,450	0.0030	30	1500	0.62	0.50
QQP	60,641	0.0032	10	500	0.60	0.48
QNLI	17,457	0.0114	10	500	0.73	0.59

Table 9: Singular value analysis on QNLI.

Method	Relative error	MAPE
FedSVD	0.0001775	0.01437
FedAnchor (ours)	0.00008775	0.007474

compression are all deterministic functions of privatized quantities, so they incur no further loss by the post-processing invariance of DP.

Each client is sampled in $K'/K = 1/2$ of the $R = 100$ rounds and takes τ local steps per participation, so the total number of composed DP-SGD steps per client is $T = \tau RK'/K$: $T = 1500$ for SNLI and MNLI and $T = 500$ for QQP and QNLI. We do not claim amplification from client subsampling. Composition runs over every step in which a client’s data is accessed. Our Dirichlet partition is label-skewed at fixed client size, so all clients share the same $N_k = N/6$, $q = b/N_k$ and σ , and Table 8 is exact rather than an average over clients. We compute σ with the Opacus RDP accountant at the target (ϵ, δ) for T steps at sampling rate q . Table 8 reports these quantities per dataset.

E Singular Value Analysis

We evaluate the noise robustness of FedSVD and FedAnchor by computing the relative spectral ℓ_2 distance (relative error) and mean absolute percentage error (MAPE) from the singular values of the combined model weights ($W + BA$) obtained under non-private ($\epsilon = \infty$) and differentially private ($\epsilon = 3$) settings. We train with a fixed random seed for 1 round, compute the singular values of the last attention layer, and then truncate the singular values to top- r rank with $r = 8$. Results show that FedAnchor cuts the spectral error roughly in half compared to FedSVD across both metrics. Singular values are stable under small perturbations by Weyl’s inequality, so this measures distortion of the retained spectrum rather than recovery of the retained directions, which depends on spectral gaps. We report it as consistent with, not decisive for, the anchoring hypothesis.

Algorithm 1 FedAnchor

Input: pre-trained weights $\{W_0^{(l)}\}_{l=1}^L$; client datasets $\{\mathcal{D}_k\}_{k=1}^K$; rounds R ; clients per round K' ; local steps τ ; batch size b ; rank r ; learning rate η ; clipping norm C ; noise multiplier σ .

Output: effective weights $\{W_R^{(l)} + B_R^{(l)} A_R^{(l)}\}_{l=1}^L$.

Initialize: $B_0^{(l)} \leftarrow \mathbf{0}$, $A_0^{(l)} \leftarrow \text{KAIMINGUNIFORM}$ for all l .

```
1: for  $t = 1, \dots, R$  do
2:   sample  $S_t \subseteq \{1, \dots, K\}$  with  $|S_t| = K'$ ;  $m_t \leftarrow \sum_{k \in S_t} n_k$ 
3:   if  $t$  odd then  $active \leftarrow B$  else  $active \leftarrow A$  ▷ alternating schedule
4:   for each client  $k \in S_t$  in parallel do
5:     receive  $\{W_{t-1}^{(l)}, B_{t-1}^{(l)}, A_{t-1}^{(l)}\}_{l=1}^L$ 
6:     initialize  $B_{k,t} \leftarrow B_{t-1}$ ,  $A_{k,t} \leftarrow A_{t-1}$ 
7:     for  $s = 1, \dots, \tau$  do ▷ DP-SGD on the active block only
8:       sample a batch of size  $b$  from  $\mathcal{D}_k$ 
9:       clip per-example gradients of the active block to  $\ell_2$  norm  $C$ 
10:      add  $\mathcal{N}(0, C^2 \sigma^2 I)$  to the summed clipped gradients; take an  $\eta$  step
11:    end for
12:    send the active block only:  $\hat{B}_{k,t}$  if  $t$  odd, else  $\hat{A}_{k,t}$ 
13:  end for
14:  for  $l = 1, \dots, L$  do ▷ aggregation, then model-centric SVD
15:    if  $t$  odd then
16:       $\hat{B}_t^{(l)} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} \hat{B}_{k,t}^{(l)}$ ;  $\tilde{W}_t^{(l)} \leftarrow W_{t-1}^{(l)} + \hat{B}_t^{(l)} A_{t-1}^{(l)}$ 
17:    else
18:       $\hat{A}_t^{(l)} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} \hat{A}_{k,t}^{(l)}$ ;  $\tilde{W}_t^{(l)} \leftarrow W_{t-1}^{(l)} + B_{t-1}^{(l)} \hat{A}_t^{(l)}$ 
19:    end if
20:     $U, \Sigma, V^\top \leftarrow \text{SVD}(\tilde{W}_t^{(l)})$  ▷ truncated SVD suffices; see Table 4
21:     $B_t^{(l)} \leftarrow U[:, :r] \sqrt{\Sigma[:, :r]}$ ;  $A_t^{(l)} \leftarrow \sqrt{\Sigma[:, :r]} V^\top[:, :r]$ 
22:     $W_t^{(l)} \leftarrow \tilde{W}_t^{(l)} - B_t^{(l)} A_t^{(l)}$  ▷ anchoring update;  $W_t^{(l)} + B_t^{(l)} A_t^{(l)} = \tilde{W}_t^{(l)}$ 
23:  end for
24:  broadcast  $\{W_t^{(l)}, B_t^{(l)}, A_t^{(l)}\}$  (optionally compressing the base-weight delta)
25: end for
```

F Pseudocode

We outline FedAnchor in Algorithm 1, making explicit the alternating schedule (line 3), the placement of noise injection at the gradient level within the local loop (lines 9–10), and the placement of the SVD after server-side aggregation (lines 14–22).