Federated Dynamical Low-Rank Training with Global Loss Convergence Guarantees

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

In this work, we propose a federated dynamical low-rank training (FeDLRT) scheme to reduce client compute and communication costs - two significant performance bottlenecks in horizontal federated learning. Our method builds upon dynamical low-rank splitting schemes for manifold-constrained optimization to create a global low-rank basis of network weights, which enables client training on a small coefficient matrix. A consistent global low-rank basis allows us to incorporate a variance correction scheme and prove global loss descent and convergence to a stationary point. Dynamic augmentation and truncation of the low-rank bases automatically optimizes computing and communication resource utilization. We demonstrate the efficiency of FeDLRT in an array of computer vision benchmarks and show a reduction of client compute and communication costs by up to an order of magnitude with minimal impacts on global accuracy.

1 Introduction

Federated learning (FL) [20, 33, 23] builds a global model on a central *server* from data distributed on multiple devices, i.e., *clients*, by iteratively aggregating local models trained with the computation resource on the clients. In horizontal FL, where all clients share identical model architecture and data features, computation is often limited by (i) the communication bandwidth between clients and the server and (ii) the restricted compute and memory resources at each client. The former could be addressed by deploying various compression techniques, such as sparse randomized sketching [9], subsampling [18], or by allowing for partial [23, 26] or asynchronous [35, 4] communications. The latter could be addressed by sparse training [29, 41] and transfer learning [5].

Since FedAvg [23], low-rank methods have been proposed to increase communication and compute efficiency for FL in [28, 43, 21, 42, 40, 12, 18, 30]. These methods can be categorized into: 1) methods that purely reduce communication cost by communicating only the low-rank factors obtained by performing a full-size SVD (or similar factorization methods) on the weight matrix after client optimization [28, 37, 40] and 2) methods that reduce both communication and client compute costs by learning only low-rank factors on clients [21, 43, 42, 12, 18].

Contribution: This work focuses on the horizontal FL setting and addresses the challenges of communication bandwidth and client compute resources simultaneously by leveraging low-rank approximations of weight matrices that follow the dynamics of the gradient flow. The proposed method features 1) **Efficient communication** — only transmitting low-rank factors; 2) **Low client compute and memory footprint** — clients optimizing only a small coefficient matrix; 3) **Automatic server-side compression** — minimizing memory and communication requirements during training via server-side dynamical rank adjustment; 4) **Global loss convergence guarantees** — converging to a stationary point by incorporating a variance correction scheme [24]. Each of these features is demonstrated on benchmark problems. To the best of the authors' knowledge, this is the first low-rank method possessing all these features.

2 Background and problem statement

Federated optimization typically considers *distributed* setups and with *limited communication* and *limited client compute and memory* resources [23]. In this work, we consider a general federated optimization problem, i.e.,

$$\min_w \mathcal{L}(w) := \frac{1}{C}\sum_{c=1}^{C} \mathcal{L}_c(w), \tag{1}$$

where w is a trainable weight, $\mathcal{L}$ is the global loss function associated to a global dataset X, and $\mathcal{L}_c$ is the local loss function of client c with local dataset X_c in a federated setup with C clients. For notational simplicity, we consider that $X = \cup_{c=1}^{C} X_c$ and each X_c is of the same size. Therefore, $\mathcal{L}$ is an average of $\mathcal{L}_c$ with uniform weights. The extension to handle a (non-uniform) weighted average case is straightforward. As the first baseline for federated optimization, we consider FedAvg [23], see Algorithm 3. Here, each client optimizes its local loss function $\mathcal{L}_c$ for s_* local iterations using gradient descent,

$$w_c^{s+1} = w_c^s - \lambda \nabla_w \mathcal{L}(w_c^s), \tag{2}$$

with learning rate λ, for $s = 0, \ldots, s_* - 1$. The initial value for the local iteration is the last global weight, i.e., $w_c^0 = w^t$. After local iterations, the weights are communicated to and aggregated at the server to update the global weight following

$$w^{t+1} = \frac{1}{C}\sum_{c=1}^{C} w_c^{s_*}. \tag{3}$$

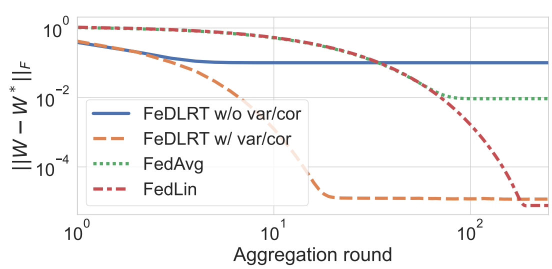

Figure 1: Federated, heterogeneous least squares regression problem, see Section 4.1, for $C = 4$ clients, $s_* = 100$ iterations, learning rate $\lambda = 1\mathrm{e}-3$ and C rank-1 local target functions. FL methods without variance correction plateau quickly, whereas FedLin and FeDLRT with variance correction converge to $1\mathrm{e}-5$. FeDLRT converges faster than FedLin and has lower communication costs.

Client-drift effect is a common challenge in FL, where the iterative client updates (2) of FedAvg converge to local minima and jeopardize global training performance since the average of the local minimizers may be far away from the global minimizer. These effects are particularly pronounced for a large number of local iterations s_*, or high discrepancies between local loss functions $\mathcal{L}_c$, as illustrated by Figure 1. Multiple methods [33, 20, 27, 14, 39] have been proposed to mitigate this issue. However, these methods often exhibit a *speed-accuracy conflict*, where learning rates need to be heavily reduced; thus, convergence is slow.

Variance correction[1] introduced in the FedLin method [24] constructs a variance correction term $V_c = \nabla_w \mathcal{L}_c(w^t) - \frac{1}{C}\sum_{c=1}^{C} \nabla_w \mathcal{L}_c(w^t)$ and modifies the client update iteration to

$$w_c^{s+1} = w_c^s - \lambda \left(\nabla_w \mathcal{L}(w_c^s) - V_c\right), \qquad s = 0, \ldots, s_* - 1. \tag{4}$$

This technique leads to global convergence to the minimizer of (1) with constant learning rates [24] for convex $\mathcal{L}$ and else to convergence to a stationary point, at the cost of an additional communication round for computing the variance correction.

Federated neural network training considers problem (1) with the trainable weight w being the set of weight matrices $\{W_i\}_i^L$ of an L layer neural network. In each iteration, the weight updates in (2) and (4) are applied to all layers simultaneously. Therefore, w.l.o.g., we express the local loss function as $\mathcal{L}_c(W)$, where $W \in \mathbb{R}^{n \times n}$ denotes the weight matrix of an arbitrary layer.

Low-rank neural network training: An array of recent work has provided theoretical and experimental evidence that layer weights of over-parameterized networks tend to be low rank [1, 2, 8, 22] and that removing small singular values may even lead to increased model performance while dramatically reducing model size [34, 32] in non-federated scenarios. This beneficial feature has spawned a

[1]Variance correction is commonly referred to as "variance reduction" [17, 24].

rich landscape of methods to compress neural networks to a low-rank factorization after training with subsequent fine-tuning [31, 6, 36, 19], train the factorized network with fixed rank [13, 38, 15], dynamically adjust the rank during training [32, 44], or use low-rank adapters for fine-tuning foundation models [11, 7, 45].

Dynamical Low-rank Approximation of the gradient flow of neural network training. The core contribution of this paper builds on the dynamical low-rank approximation (DLRA) method, which was initially proposed for solving matrix equations [16] and recently extended to neural network training [32, 44, 10]. Let $\dot{W}(t) = -\nabla_W \mathcal{L}(W(t))$ denote the gradient flow for minimizing $\mathcal{L}$.

The DLRA method restricts the trajectory of W to $\mathcal{M}_r$, the manifold of $n \times n$, rank-r matrices, by projecting $\dot{W}$ onto a local tangent plane of $\mathcal{M}_r$ via an orthogonal projection. This guarantees a low-rank solution when following the projected dynamics from a low-rank initial guess. Let the low-rank matrix take the form $W_r = USV^\top \in \mathcal{M}_r$ with $U, V \in \mathbb{R}^{n\times r}$ the orthonormal bases of $\mathcal{M}_r$ and $S \in \mathbb{R}^{r\times r}$ the coefficient matrix. The dynamics for each low-rank factor in DRLA are then derived in [16, Proposition 2.1] as

$$\begin{aligned}
\dot{S}(t) &= -U^\top(t)\nabla_W \mathcal{L}(U(t)S(t)V(t)^\top)V(t),\\
\dot{U}(t) &= -\left(I - P_{U(t)}\right)\nabla_W \mathcal{L}(U(t)S(t)V(t)^\top)V(t)S(t)^{-1},\\
\dot{V}(t) &= -\left(I - P_{V(t)}\right)\nabla_W \mathcal{L}(U(t)S(t)V(t)^\top)U(t)S(t)^{-\top},
\end{aligned} \tag{5}$$

where $P_U = UU^\top$ and $P_V = VV^\top$ are the projections onto the column spaces of U and V, respectively. By using the *basis update & Galerkin* (BUG) scheme [3], (5) can be split into a basis update step for U and V and a coefficient update step for S. This splitting scheme allows for dynamic adjustment of the rank via a basis augmentation before the coefficient update step and a basis truncation after the coefficient update, as shown in [32].

3 FeDLRT: Federated dynamical low-rank training with variance correction

In this section, we present the core contribution of this paper, *federated dynamical low-rank training* (FeDLRT), which features a low-rank client optimization step with optional variance correction and an efficient server aggregation process that dynamically determines the optimal weight matrix rank for automatic compression.

In the context of FL, the BUG of DLRA splitting scheme is particularly interesting since it allows for learning the low-rank bases and coefficients in separate steps. This gives rise to a globally shared basis for the local client iterations, reducing communication and client compute cost of the proposed FeDLRT scheme, see Figure 2: First, the factorization is broadcast to the clients (panel 1), and the basis gradients[2] U, V are aggregated on the server (panel 2). Next, the basis is augmented on the server (panel 3) and broadcast. On the clients, only the augmented coefficient matrix S is updated repeatedly (panel 4) before aggregation to the server. After aggregation of the local augmented coefficient matrices, redundant basis directions are eliminated to optimize the accuracy-to-compression ratio of the model on the server.

The strategy yields the following benefits compared to "full-rank" FL schemes as FedLin [24] and low-rank schemes with local compression:

Low client compute cost: Server-based basis augmentation and compression enables an automatic compression without a-priori knowledge of the layer rank r and at no cost for the resource-constrained clients. The clients only evaluate gradients of low-rank factors and optimize the small matrix $S \in \mathbb{R}^{r\times r}$.

Efficient communication: Similar to FedLin, FeDLRT requires *in practice* two communication rounds – one for aggregating and distributing global gradients for basis augmentation and variance correction and one for aggregating locally updated coefficients. However, communication cost for each round is significantly reduced since only low-rank factors are communicated. We refer to Section 3.3 on communication and compute cost.

Existing federated low-rank schemes effectively generate individual and incompatible representations of $W_r \in \mathcal{M}_r$ for each client. While the factors can still be efficiently communicated, averaging on

[2]and later on the coefficient gradients for variance correction

the server requires a reconstruction of the full weigh matrix $W^* = \frac{1}{C}\sum_{c=1}^{C} U_c S_c V_c^\top$, since the local manifolds possibly diverge. Thus, the local rank information is lost and needs to be costly recovered by a full $n \times n$ SVD on the server; see Algorithm 6 for details. Since the average of low-rank matrices is not necessarily of low rank, these schemes may lose crucial information on the manifold if client solutions drift too far apart from each other. FeDLRT, in contrast, provides the advantage of **client-wide manifold consistency:** Splitting the low-rank update and sharing bases amongst clients provides a globally consistent manifold basis. This furthermore allows for bounding the coefficient drift, see Theorem 1, and enables a variance correction for the federated low-rank similar to the FedLin scheme.

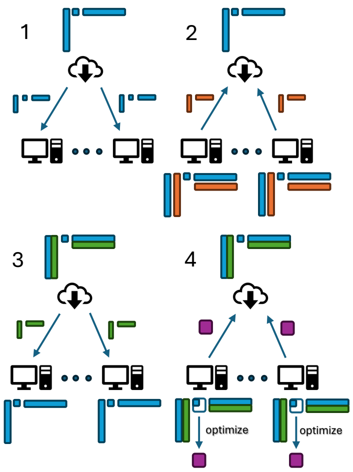

Figure 2: Communication of FeDLRT without variance correction. 1) Broadcast global basis U, V (blue). 2) Aggregate basis gradients $G_{c,U}, G_{c,V}$ (orange). 3) Broadcast global augmented basis $\bar{U}, \bar{V}$ (green). 4) Aggregate client coefficient update $\widetilde{S}_c^{s_*}$ (purple).

3.1 Description of Algorithm 1 - FeDLRT

In this section, we elaborate on the details in Algorithm 1. The orthonormal factors U^t, V^t and the coefficient matrix S^t are initialized with rank r and then broadcast to the clients. Note that FeDLRT ensures that, for all $t > 1$, U^t and V^t are orthonormal, and S^t is diagonal and full rank.

Basis augmentation of the bases U^t and V^t is performed using concatenation with the corresponding global basis gradients $G_U = \frac{1}{C}\sum_{c=1}^{C} \nabla_U \mathcal{L}_c(U^t S^t V^{t,\top})$ and $G_V = \frac{1}{C}\sum_{c=1}^{C} \nabla_V \mathcal{L}_c(U^t S^t V^{t,\top})$, obtained by aggregating the local basis gradients. G_U and G_V encapsulate the gradient flow dynamics (5) projected onto the original bases, thus yielding an intuitive choice for basis augmentation. Further, this choice is consistent with the basis update step of the augmented BUG splitting scheme, see Appendix E, which ensures the robustness of the client optimizer. Subsequent orthonormalization, e.g., by a QR decomposition, yields the augmented basis, i.e.,

$$\begin{aligned} &[U^t \mid \bar{U}]R = \mathtt{qr}([U^t \mid G_U]) \in \mathbb{R}^{n\times 2r}, \\ \text{and} \quad &[V^t \mid \bar{V}]R = \mathtt{qr}([V^t \mid G_V]) \in \mathbb{R}^{n\times 2r}. \end{aligned} \tag{6}$$

We denote the augmented bases by $\widetilde{U} = [U^t \mid \bar{U}]$ and $\widetilde{V} = [V^t \mid \bar{V}]$. The orthonormalization is performed on the server, providing compute cost reduction for the client.

Basis broadcasting of $\widetilde{U}$ and $\widetilde{V}$ only requires to broadcast the new bases $\bar{U}$ and $\bar{V}$, since U^t and V^t are readily available on the clients. Formally, the coefficients S^t are projected onto the augmented basis, i.e., $\widetilde{S} = \widetilde{U}^\top U^t S^t V^{t,\top} \widetilde{V} \in \mathbb{R}^{2r\times 2r}$, before broadcasting them to the clients. Exploiting the orthonormality of the basis results in further reduction of the communication and compute cost:

Lemma 1. *$\widetilde{S} = \widetilde{U}^\top U^t S^t V^{t,\top} \widetilde{V}$ takes the form* $\widetilde{S} = \begin{bmatrix} S^t & 0 \\ 0 & 0 \end{bmatrix}$.

See Appendix F for the proof. With Lemma 1, only $\bar{U}$ and $\bar{V}$ have to be broadcast, and the augmented bases and coefficients $\widetilde{U}$, $\widetilde{V}$, and $\widetilde{S}$ can be assembled on each client as needed. Furthermore, only $S \in \mathbb{R}^{r\times r}$, instead of $\widetilde{S} \in \mathbb{R}^{2r\times 2r}$, needs to be communicated.

Below, we discuss three options for the client coefficient update step.

Client coefficient update without variance correction is implemented similarly to FedAvg (3). On each client c, the augmented coefficient matrix $\widetilde{S}_c$ is trained for s_* iterations[3] with learning rate λ,

$$\widetilde{S}_c^{s+1} = \widetilde{S}_c^{s} - \lambda \nabla_{\widetilde{S}} \mathcal{L}_c(\widetilde{U}\widetilde{S}_c^{s}\widetilde{V}^\top), \quad s = 0, \ldots, s_* - 1, \quad \text{with} \qquad \widetilde{S}_c^{s=0} = \widetilde{S}. \tag{7}$$

Client coefficient update with variance correction is required in certain federated scenarios, e.g., the case considered in Figure 1. Based on FedLin [24], we introduce a correction step for the local coefficient update of FeDLRT. It extends the above local iteration by another communication round,

[3] Our analysis focuses on the case where all clients share the same number of local iterations s_*. The analysis can be extended to the case where s_* is client dependent, following a similar strategy as in [24].

where the gradient of the augmented coefficients $G_{\widetilde{S},c} = \nabla_{\widetilde{S}}\mathcal{L}_c(\widetilde{U}\widetilde{S}\widetilde{V}^\top)$ is computed, aggregated to $G_{\widetilde{S}} = \frac{1}{C}\sum_{c=1}^{C} G_{\widetilde{S},c}$ and subsequently broadcast. This yields a correction term $V_c = G_{\widetilde{S}} - G_{\widetilde{S},c}$ for each client c and thus the client iterations read

$$\widetilde{S}_c^{s+1} = \widetilde{S}_c^s - \lambda\left(\nabla_{\widetilde{S}}\mathcal{L}_c(\widetilde{U}\widetilde{S}_c^s\widetilde{V}^\top) + V_c\right), \quad s = 0,\dots,s_*-1, \quad \text{with} \qquad \widetilde{S}_c^{s=0} = \widetilde{S}. \tag{8}$$

The correction term results in a bound on the coefficient drift and leads to convergence guarantees for FeDLRT, as detailed in Section 3.2.

Client coefficient update with simplified variance correction: Empirically, we observe that a simplified variance correction, which only considers the correction term of the *non-augmented* coefficients S^t, is sufficient, see Figure 6. The simplified variance correction term takes the form

$$V_c = G_{\widetilde{S}} - G_{\widetilde{S},c} \approx \check{V}_c := \check{G}_{\widetilde{S}} - \check{G}_{\widetilde{S},c} = \begin{bmatrix} \nabla_S\mathcal{L}(U^tS^tV^{t,\top}) - \nabla_S\mathcal{L}_c(U^tS^tV^{t,\top}) & 0 \\ 0 & 0 \end{bmatrix}, \tag{9}$$

which makes lines 10 and 12 in Algorithm 1 redundant, since $\check{G}_{\widetilde{S}}$ can be aggregated in one step with the basis gradients G_U,G_V in line 4 and broadcast with $\bar{U}, \bar{V}$ in line 6, reducing the communication rounds to two - the same as FedLin. See Algorithm 5 for details.

Coefficient averaging is performed after (any of the above variants of) the client iterations. The server computes the updated global coefficients by averaging the local updates, i.e., $\widetilde{S}^* = \frac{1}{C}\sum_{c=1}^{C}\widetilde{S}_c^{s_*}$. With the shared augmented bases $\widetilde{U}$ and $\widetilde{V}$, this is equivalent to the FedAvg aggregation

$$\widetilde{W}_r^* = \tfrac{1}{C}\textstyle\sum_{c=1}^{C}\widetilde{W}_r^{s_*} = \tfrac{1}{C}\sum_{c=1}^{C}\left(\widetilde{U}\widetilde{S}_c^{s_*}\widetilde{V}^\top\right) = \widetilde{U}(\tfrac{1}{C}\sum_{c=1}^{C}\widetilde{S}_c^{s_*})\widetilde{V}^\top = \widetilde{U}\widetilde{S}^*\widetilde{V}^\top. \tag{10}$$

Since the basis is fixed, the rank $2r$ is preserved in the aggregation, which is in contrast to other federated low-rank schemes where the aggregated weights could be full rank and, in turn, require a full matrix SVD to determine the new rank [28, 40].

Automatic compression via rank truncation is necessary 1) to identify the optimal rank of the weight matrix and 2) to ensure that S is full rank[4]. To this end, a truncated SVD of $\widetilde{S}^* \in \mathbb{R}^{2r\times 2r}$ is performed, i.e. $P_{r_1}, \Sigma_{r_1}, Q_{r_1}^\top = \mathtt{svd}(\widetilde{S}^*)$, where $P_{r_1}, Q_{r_1} \in \mathbb{R}^{2r\times r_1}$ and $\Sigma_{r_1} = \mathrm{diag}(\sigma_1,\dots,\sigma_{r_1})$ contains the r_1 largest singular values of $\widetilde{S}^*$. The new rank r_1 can be chosen by a variety of criteria, e.g., a singular value threshold $\|[\sigma_{r_1},\dots,\sigma_{2r}]\|_2 < \vartheta$. Once a suitable rank is determined, the factorization is updated by the projection of the bases $U^{t+1} = \widetilde{U}P_{r_1} \in \mathbb{R}^{n\times r_1}$, $V^{t+1} = \widetilde{V}Q_{r_1} \in \mathbb{R}^{n\times r_1}$ and update of the coefficient $S^{t+1} = \Sigma_{r_1}$. Remarkably, Algorithm 1 is a federated low-rank learning scheme whose solution is close to a full-rank solution, see Theorem 5.

FeDLRT can readily be extended to tensor-valued, e.g., convolutional, layers by applying Algorithm 1 to each basis and the core tensor in a Tucker Tensor factorization. We refer to Appendix B for details.

3.2 Analysis of FeDLRT with variance correction

In this section, we analyze the FeDLRT algorithm under the general assumption that $\mathcal{L}_c$ and $\mathcal{L}$ are L-smooth with constant L. Theorems 2 and 3 give the convergence results for FeDLRT with full variance correction (8) in Algorithm 1. Theorem 4 and Corollary 1 provide the convergence for FeDLRT with simplified variance correction in (9), as detailed in Algorithm 5, under additional assumptions given therein. We note that the analysis does not require convexity of $\mathcal{L}_c$ or $\mathcal{L}$.

FeDLRT convergence with full variance correction. The variance-corrected client iteration (8) leads to the following bound the client coefficient drift.

Theorem 1. *Given augmented basis and coefficient matrices $\widetilde{U}$, $\widetilde{V}$, and $\widetilde{S}$. If the local learning rate $0 < \lambda \leq \frac{1}{Ls_*}$ with $s_* \geq 1$ the number of local steps, for all clients c,*

$$\|\widetilde{S}_c^s - \widetilde{S}_c\| \leq \exp(1)s_*\lambda\|\nabla_{\widetilde{S}}\mathcal{L}(\widetilde{U}\widetilde{S}\widetilde{V}^\top)\|, \quad \text{for} \quad s = 1,\dots,s^*-1, \tag{11}$$

where $\widetilde{S}_c^s$ is the variance corrected coefficient as given in (8).

[4] Full rank S is required to show consistency of the basis update step (6) with the robust operator splitting of [3, 32], see Appendix E.

Algorithm 1: FeDLRT (See Algorithm 2 for auxiliary function definitions)

Input : Initial orthonormal bases $U^1, V^1 \in \mathbb{R}^{n\times r}$ and full rank $S^1 \in \mathbb{R}^{r\times r}$;
Client-server setup with clients $c = 1, \dots, C$;
`var_cor`: Boolean flag to activate variance correction;
τ: singular value threshold for rank truncation.

for $t = 1, \dots, T$ **do**
- `broadcast`$(\{U^t, V^t, S^t\})$
- $G_{U,c} \leftarrow \nabla_U \mathcal{L}_c(U^t S^t V^{t,\top})$; $G_{V,c} \leftarrow \nabla_V \mathcal{L}_c(U^t S^t V^{t,\top})$ `/* On client */`
- $G_U, G_V \leftarrow$ `aggregate`$(\{G_{U,c}, G_{V,c}\})$
- $\bar{U} \leftarrow$`basis_augmentation`(U^t, G_U); $\bar{V} \leftarrow$`basis_augmentation`(V^t, G_V)
- `broadcast`$(\{\bar{U}, \bar{V}\})$
- $\widetilde{U} \leftarrow [U^t \mid \bar{U}]$; $\widetilde{V} \leftarrow [V^t \mid \bar{V}]$ `/* Basis assembly on client */`
- $\widetilde{S}^{s=0} \leftarrow \begin{bmatrix} S^t & 0 \\ 0 & 0 \end{bmatrix}$ `/* Coefficient matrix assembly on client */`
- **if** `var_cor` **then**
  - $G_{\widetilde{S},c} \leftarrow \nabla_{\widetilde{S}} \mathcal{L}_c(\widetilde{U}\widetilde{S}\widetilde{V}^\top)$ `/* Augmented gradient on client */`
  - $G_{\widetilde{S}} \leftarrow$ `aggregate`$(\{G_{\widetilde{S},c}\})$
  - `broadcast`$(\{G_{\widetilde{S}}\})$
  - `coefficient_update_var_cor`$(c,\ G_{\widetilde{S}} - G_{\widetilde{S},c})$ `/* On client */`
- **else**
  - `coefficient_update`(c) `/* On client */`
- $\widetilde{S}^* \leftarrow$ `aggregate`$(\{\widetilde{S}_c^{s_*}\})$
- $P_{r_1}, \Sigma_{r_1}, Q_{r_1} \leftarrow$ `svd`$(\widetilde{S}^*)$ with threshold ϑ `/* Compression step */`
- $U^{t+1} \leftarrow \widetilde{U} P_{r_1}$; $V^{t+1} \leftarrow \widetilde{V} Q_{r_1}$; $S^{t+1} \leftarrow \Sigma_{r_1}$ `/* Basis and coefficient update */`

Table 1: Comparison of the computational footprint of FeDLRT with FedAvg, FedLin and several low-rank FL methods. The FeDLRT variants are the only low-rank schemes with linearly scaling (in n) memory, compute, and communication costs with automatic compression and variance correction.

Method	Client compute	Client memory	Server compute	Server memory	Com. Cost	Com. Rounds	var/cor.	rank adaptive
FedAVG [23]	$\mathcal{O}(s_* b n^2)$	$\mathcal{O}(2n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(2n^2)$	$\mathcal{O}(2n^2)$	1	✗	✗
FedLin [24]	$\mathcal{O}(s_* b n^2)$	$\mathcal{O}(2n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(2n^2)$	$\mathcal{O}(4n^2)$	2	✓	✗
FeDLRT w/o var/cor	$\mathcal{O}(s_* b(4nr + 4r^2))$	$\mathcal{O}(4(nr + 2r^2))$	$\mathcal{O}(2nr + (8+4n)r^2 + 8r^3)$	$\mathcal{O}(2nr + 4r^2)$	$\mathcal{O}(6nr + 6r^2))$	2	✗	✓
FeDLRT simpl. var/cor	$\mathcal{O}(s_* b(4nr + 4r^2) + r^2)$	$\mathcal{O}(4(nr + 2r^2))$	$\mathcal{O}(2nr + (8+4n)r^2 + 8r^3)$	$\mathcal{O}(2nr + 4r^2)$	$\mathcal{O}(6nr + 8r^2)$	2	✓	✓
FeDLRT full var/cor	$\mathcal{O}(s_* b(4nr + 4r^2) + 4r^2)$	$\mathcal{O}(4(nr + 2r^2))$	$\mathcal{O}(2nr + (8+4n)r^2 + 8r^3)$	$\mathcal{O}(2nr + 4r^2)$	$\mathcal{O}(6nr + 10r^2)$	3	✓	✓
FeDLR [28]	$\mathcal{O}(s_* b n^2 + n^3)$	$\mathcal{O}(2n^2)$	$\mathcal{O}(n^2 + n^3)$	$\mathcal{O}(4nr)$	$\mathcal{O}(4nr)$	1	✗	✓
Riemannian FL [40]	$\mathcal{O}(2n^2 r + 4nr^2 + 2nr)$	$\mathcal{O}(2n^2)$	$\mathcal{O}(2nr + n^2 r)$	$\mathcal{O}(4nr)$	$\mathcal{O}(4nr)$	1	✗	✓

The critical ingredient for the proof, provided in Appendix G.1, is the globally shared augmented bases. Theorem 1 bounds the drift of the low-rank representations of the local weight, which gives rise to the following global loss descent guarantee.

Theorem 2. *Let $U^t S^t V^{t,\top}$ and $U^{t+1} S^{t+1} V^{t+1,\top}$ be the factorization before and after iteration t of Algorithm 1 with variance correction and singular value truncation threshold ϑ. Let the local learning rate be $0 < \lambda \leq \frac{1}{12 L s_*}$, then the global loss descent is bounded by*

$$\mathcal{L}(U^{t+1} S^{t+1} V^{t+1,\top}) - \mathcal{L}(U^t S^t V^{t,\top}) \leq -s_* \lambda (1 - 12 s_* \lambda L) \|\nabla_{\widetilde{S}} \mathcal{L}(\widetilde{U}\widetilde{S}\widetilde{V}^\top)\|^2 + L\vartheta. \tag{12}$$

The proof is provided in Appendix G.2. Theorem 2 paves the way for the following result on convergence to a global stationary point.

Theorem 3. *Algorithm 1 guarantees that, for learning rate $\lambda \leq \frac{1}{12 L s_*}$ and final iteration T,*

$$\min_{t=1,\dots,T} \left\|\nabla_{\widetilde{S}} \mathcal{L}(U^t S^t V^{t,\top})\right\|^2 \leq \frac{48L}{T} \left(\mathcal{L}(U^1 S^1 V^{1,\top}) - \mathcal{L}(U^{T+1} S^{T+1} V^{T+1,\top})\right) + 48 L^2 \vartheta. \tag{13}$$

The proof is given in Appendix G.3. In particular, this theorem implies convergence of Algorithm 1 for $T \to \infty$ up to a ϑ-distance to a global stationary point. This is consistent with the numerical

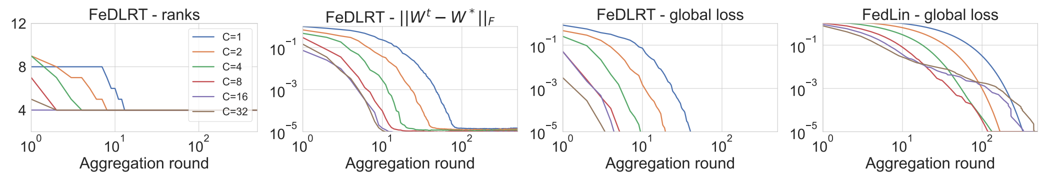

Figure 3: Comparison between FeDLRT with simplified variance correction and FedLin in the homogeneous linear least squares regression test. Each line represents the median result of 20 random initialization with C clients. The plots from left to right show the rank evolution, the distance to the global optimizer, the global loss values by FeDLRT, and the global loss values by FedLin. The results show that FeDLRT converges faster in this low-rank test case by identifying (and never underestimating) the target rank $r = 4$ early in the training.

results in Figure 1, where FedLin converges to the global minimizer (the only stationary point) while FeDLRT with variance correction stops at a point with slightly higher loss value due to a nonzero ϑ. In the case that the FL problem has a low-rank solution, the truncation error bounded by ϑ vanishes, and convergence to a stationary point is guaranteed, see, e.g., Figure 3.

FeDLRT convergence with simplified variance correction. FeDLRT with simplified variance correction is detailed in Algorithm 5 with the variance correction term given in (9), which makes variance correction more communication and computation efficient but comes at a cost of the following additional assumption for convergence analysis.

Assumption 1. *There exists $\delta \ll 1$ such that, at each client coefficient update,*

$$\|\nabla_{\widetilde{S}}\mathcal{G}(\widetilde{U}\widetilde{S}_c^s\widetilde{V}^\top)\| - \|\nabla_S\mathcal{G}(\widetilde{U}\widetilde{S}_c^s\widetilde{V}^\top)\| < \delta\|\nabla_{\widetilde{S}}\mathcal{L}(\widetilde{U}\widetilde{S}\widetilde{V}^\top)\|, \tag{14}$$

for functions $\mathcal{G} = \mathcal{L}$ and $\mathcal{G} = \mathcal{L}_c$, $c = 1, \dots, C$.

This assumption can be interpreted as that most of dynamics in the gradient flow are captured in the coefficient update for the original rank-r matrix S, and the basis augmentation provides little information. This scenario occurs when FeDLRT identifies the optimal rank, which could happen early for simpler problems as shown in Figure 3, or when FeDLRT approaches a stationary point.

Theorem 4. *Under Assumption 1, let $\mathsf{C} := s_*\lambda(1 - \delta^2 - 12s_*\lambda L + \delta^2\, s_*\lambda)$. If the local learning rate $0 < \lambda \le \frac{1}{12Ls_*}$, Algorithm 5 leads to the global loss descent*

$$\mathcal{L}(U^{t+1}S^{t+1}V^{t+1,\top}) - \mathcal{L}(U^tS^tV^{t,\top}) \le -\mathsf{C}\|\nabla_{\widetilde{S}}\mathcal{L}(\widetilde{W}_r)\|^2 + L\vartheta.$$

The proof is provided in Appendix H.1. When δ is small, this bound is slightly weaker than the one in Theorem 2, which leads to the following corollary.

Corollary 1. *Assume that Assumption 1 holds. Algorithm 5 guarantees that, for the local learning rate $0 < \lambda \le \frac{1}{s_*(12L+\delta^2)}$,*

$$\min_{t=1,\dots,T}\left\|\nabla_{\widetilde{S}}\mathcal{L}(U^tS^tV^{t,\top})\right\|^2 \le \frac{96L}{T}(\mathcal{L}(U^1S^1V^{1,\top}) - \mathcal{L}(U^{T+1}S^{T+1}V^{T+1,\top})) + 96L^2\vartheta.$$

The proof is analogous to the one for Theorem 3, see Appendix H.2.

3.3 Compute and communication cost

The proposed FeDLRT methods significantly reduce server and client memory footprint, the required communication bandwidth, as well as the client compute cost compared to various baselines, see Table 1. We remark that the complete federated learning process is performed on the low-rank factors, and the full matrix W_r is never required, as, e.g., in [28, 40] and FeDLRT is the only low-rank method with adaptive compression incorporating variance correction, whose server compute cost scales linearly with the layer dimension since the SVD for rank truncation only needs to be computed on the augmented coefficient matrix of size $2r \times 2r$.

4 Numerical evaluation

4.1 Distributed linear least squares regression

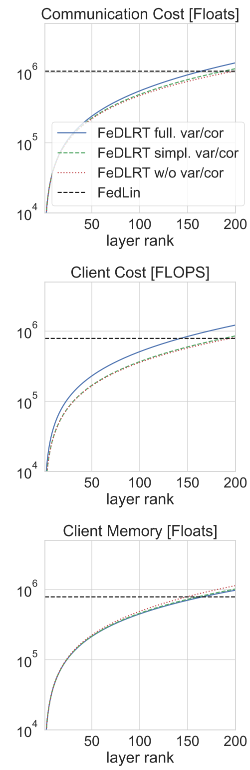

Figure 4: Scaling of communication cost (top) compute cost at a single client (middle), and client memory footprint (bottom) for $s_* = 1$ client iteration and a single data-point for $W \in \mathbb{R}^{n \times n}$ with $n = 512$. In practice we have $r \ll n$, see Section 4.

Homogeneous test. We first consider a (convex) FL problem (1) for linear least squares regression with local loss $\mathcal{L}_c(W) = \frac{1}{2|X_c|}\sum_{(x,y)\in X_c} \left\| p(x)^\top W p(y) - f(x,y) \right\|_2^2$, where $W \in \mathbb{R}^{n\times n}$ and $p : [-1, 1] \to \mathbb{R}^n$ is the Legendre polynomial basis of degree $n-1$. The target function f is manufactured as $f(x, y) = p(x)^\top W_r p(y)$, where $\operatorname{rank}(W_r) = r$. We consider problems with $n = 20$, $r = 4$, and randomly generated W_r, with $10,000$ data points uniformly sampled on $[-1, 1]^2$ and uniformly distributed among clients. We compare FeDLRT with variance correction and FedLin with $s_* = 20$ local iterations and $\lambda = 1e-3$ learning rate on $C = 1, 2, 4, 8, 16, 32$ clients. This setting satisfies the step-size restriction given in Theorem 2. In FeDLRT, the singular value truncation threshold $\vartheta = \tau||\widetilde{S}^*||$ with $\tau = 0.1$ was used.

Figure 3 reports the dynamically updated ranks, errors, and loss values with respect to the aggregation rounds. The reported data are the medians over 20 randomly generated initial weights[5] The results indicate that FeDLRT is able to identify the correct rank within a few aggregation rounds and, furthermore, never underestimates it – which would have increased the loss value significantly. FeDLRT converges to the minimizer $W^* = W_r$ up to a $1e-5$ error and converges faster with more clients. On this problem, FeDLRT shows up to 10x faster convergence than FedLin. We attribute this behavior to the fact that, by identifying a suitable low-rank manifold early in the training, FeDLRT significantly reduces the degrees of freedom in the FL problem.

Heterogeneous test. Inspired by [24], we consider a variation of the linear least squares regression with $\mathcal{L}_c(W) = \frac{1}{2|X|}\sum_{(x,y)\in X} \left\| p(x)^\top W p(y) - f_c(x,y) \right\|^2$, where the target function f_c is different for each client, and the $10,000$ training data points are available to all clients. The local target functions f_c cause each client to optimize a different local problem. We choose problem size $n = 10$ with $C = 4$ clients and use learning rate $\lambda = 1\mathrm{e}-3$ with $s_* = 100$ local epochs. As seen in Figure 1, FeDLRT with variance correction converges (to single precision accuracy) to the minimizer W^* of (1) much faster than FedLin, whereas FeDLRT without correction quickly plateaus, similar to FedAvg.

4.2 ResNet18 on CIFAR10

We demonstrate the performance of FeDLRT for training the exemplary ResNet18 model on CIFAR10, where we apply FeDLRT to train its fully connected head. The truncation tolerance is set to $\vartheta = \tau||\widetilde{S}^*||$ with $\tau = 0.01$. The test case setup is summarized in Table 2. The training data is equally partitioned across clients; see Appendix C.2 for the data-preprocessing details. A local iteration of Algorithm 1 at client c describes one mini-batch update on the client training data set X_c for a given batch size, s_* is the maximum number of local iterations, and T denotes the number of aggregation rounds. We display the statistics for 10 random initializations; each warm-started with 5 iterations with one client. We set $s_* = 240/C$ so that in each training run, the global network iterates through the same amount of data. This setup favors low client counts, and, as expected, the validation accuracy drops as C grows for FedAvg and FeDLRT without variance correction, see Figure 6 (upper row). We note that FeDLRT ties or outperforms FedAvg in terms of final validation accuracy. Using full variance correction (second row) increases the validation accuracy of FeDLRT by up to 12% in this test case, matching the accuracy of FedLin and enabling FL with 93% accuracy for 32 clients. For $C = 8$ clients, the communication cost saving of the compressed layers is up to 90%. The computationally more efficient simplified variance correction, using Algorithm 5, (third row), yields similar validation accuracy, notably at higher compression ratio and communication cost reduction. Similar results are obtained for AlexNet, VGG16 on CIFAR10, and ViT on CIFAR100,

[5]We chose to display the median trajectory to point out its convergence and monotonicity. The test case also converges in the mean.

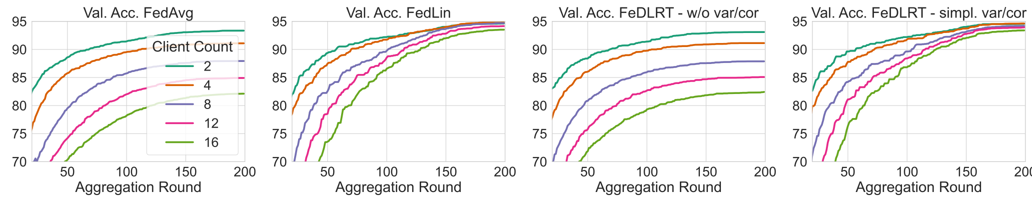

Figure 5: ResNet18 CIFAR10. We compare the convergence behavior of the median result of 10 initializations displaying the best validation accuracy until the current epoch for FedAvg (top left), FedLin (top right), FeDLRT w/o var/cor (bottom left) and FeDLRT w/ simplified var/cor (bottom right). We observe 1) the low-rank methods (bottom) closely follows the convergence dynamics of their full rank counterpart (top), and 2) variance correction starts to improve the convergence behavior during later stages of the training, where the non-corrected methods level off.

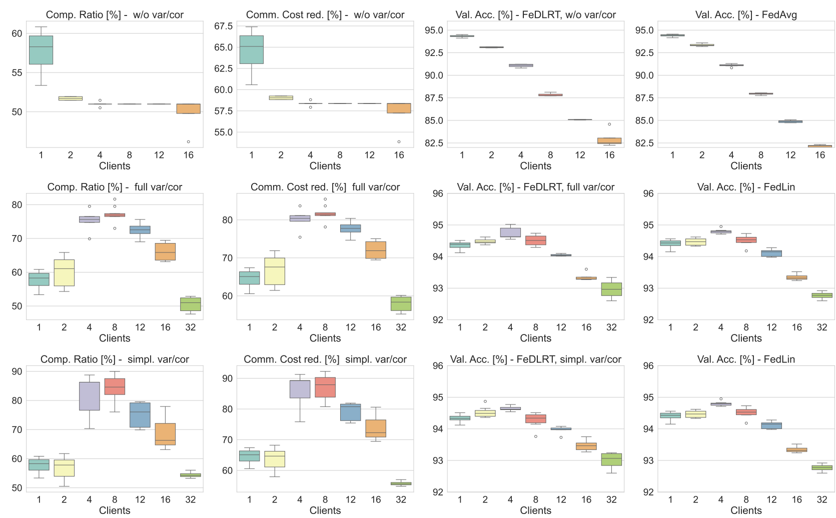

Figure 6: Comparisons for training ResNet18 on CIFAR10 benchmark. Top row compares FeDLRT without variance correction to FedAvg, middle and bottom rows compare FeDLRT with full and simplified variance correction to FedLin, respectively. In each row, the left two panels show the model compression ratio and the communication cost reduction from FeDLRT, and the right two panels show the validation accuracy for FeDLRT and the full-rank counterparts. In each plot, the results are reported for $C = 1, \ldots, 16$ or 32 clients with $240/C$ local iterations. FeDLRT matches the accuracy of FedAvg and FedLin well, while substantially reducing the server and client memory and communication costs. Variance correction leads to an up to 12% increase in validation accuracy for large C, mitigating the client drift problem. The simplified variance correction (bottom row) gives comparable results to full version (middle row) at a lower communication and computation cost.

see Appendix C, where we observe that FeDLRT closely matches the full-rank accuracy of FedLin. Lastly, we remark that variance correction ins beneficial for convergence behavior in neural network training, as shown in Figure 5.

In conclusion, we have presented FeDLRT, an efficient low-rank FL scheme with convergence guarantees and automatic compression, and demonstrated its capabilities in several test cases.

Limitations and future work: We remark that the underlying assumption for this work is that the target model can be expressed sufficiently well via a low-rank representation. Although the communication cost in terms of transferred parameters is significantly reduced compared to existing method, FeDLRT still requires two communication handshakes for one aggregation round, just like its full-rank counterpart FedLin. Therefore, the method needs to be refined for scenarios where the clients have different communication latencies or for completely asynchronous scenarios. Potential future research directions include performing large-scale tests with thousands of clients, extending the algorithm to accommodate partial client participation or asynchronous communication, and analyzing the convergence properties in these scenarios.

References

[1] S. Arora, N. Cohen, W. Hu, and Y. Luo. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32, 2019.

[2] B. Bah, H. Rauhut, U. Terstiege, and M. Westdickenberg. Learning deep linear neural networks: Riemannian gradient flows and convergence to global minimizers. *Inf. Inference*, 11(1):307–353, 2022.

[3] G. Ceruti, J. Kusch, and C. Lubich. A rank-adaptive robust integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 2022.

[4] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala. Asynchronous online federated learning for edge devices with non-iid data. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 15–24. IEEE, 2020.

[5] Y. Chen, X. Qin, J. Wang, C. Yu, and W. Gao. Fedhealth: A federated transfer learning framework for wearable healthcare. *IEEE Intelligent Systems*, 35(4):83–93, 2020.

[6] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27, 2014.

[7] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

[8] T. Galanti, Z. S. Siegel, A. Gupte, and T. Poggio. Sgd and weight decay provably induce a low-rank bias in neural networks. 2022.

[9] F. Haddadpour, B. Karimi, P. Li, and X. Li. Fedsketch: Communication-efficient and private federated learning via sketching, 2020.

[10] A. Hnatiuk, J. Kusch, L. Kusch, N. R. Gauger, and A. Walther. Stochastic aspects of dynamical low-rank approximation in the context of machine learning. *Optimization Online*, 2024.

[11] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models, 2021.

[12] N. Hyeon-Woo, M. Ye-Bin, and T.-H. Oh. Fedpara: Low-rank hadamard product for communication-efficient federated learning. In *International Conference on Learning Representations*, 2022.

[13] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference. BMVA Press*, 2014.

[14] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[15] M. Khodak, N. Tenenholtz, L. Mackey, and N. Fusi. Initialization and regularization of factorized neural layers. In *International Conference on Learning Representations*, 2021.

[16] O. Koch and C. Lubich. Dynamical low-rank approximation. *SIAM Journal on Matrix Analysis and Applications*, 29(2):434–454, 2007.

[17] J. Konečný, H. B. McMahan, D. Ramage, and P. Richtárik. Federated optimization: Distributed machine learning for on-device intelligence, 2016.

[18] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency, 2017.

[19] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky. Speeding-up convolutional neural networks using fine-tuned CP-decomposition. In *International Conference on Learning Representations*, 2015.

[20] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

[21] X.-Y. Liu, R. Zhu, D. Zha, J. Gao, S. Zhong, and M. Qiu. Differentially private low-rank adaptation of large language model using federated learning, 2023.

[22] C. H. Martin and M. W. Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *arXiv preprint arXiv:1810.01075*, 2018.

[23] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data, 2016.

[24] A. Mitra, R. Jaafar, G. J. Pappas, and H. Hassani. Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 14606–14619. Curran Associates, Inc., 2021.

[25] A. Mitra, R. Jaafar, G. J. Pappas, and H. Hassani. Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients, 2021.

[26] T. Nishio and R. Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE international conference on communications (ICC)*, pages 1–7. IEEE, 2019.

[27] R. Pathak and M. J. Wainwright. Fedsplit: An algorithmic framework for fast federated optimization. *arXiv*.

[28] Z. Qiao, X. Yu, J. Zhang, and K. B. Letaief. Communication-efficient federated learning with dual-side low-rank compression, 2021.

[29] X. Qiu, J. Fernandez-Marques, P. P. Gusmao, Y. Gao, T. Parcollet, and N. D. Lane. Zerofl: Efficient on-device training for federated learning with local sparsity. *arXiv preprint arXiv:2208.02507*, 2022.

[30] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization, 2020.

[31] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6655–6659, 2013.

[32] S. Schotthöfer, E. Zangrando, J. Kusch, G. Ceruti, and F. Tudisco. Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations. In *Advances in Neural Information Processing Systems*, 2022.

[33] O. Shamir, N. Srebro, and T. Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning*, pages 1000–1008. PMLR, 2014.

[34] P. Sharma, J. T. Ash, and D. Misra. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. In *International Conference on Learning Representations (ICLR)*, 2024.

[35] M. R. Sprague, A. Jalalirad, M. Scavuzzo, C. Capota, M. Neun, L. Do, and M. Kopp. Asynchronous federated learning for geospatial applications. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 21–28. Springer, 2018.

[36] A. Tjandra, S. Sakti, and S. Nakamura. Compressing recurrent neural network with tensor train. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4451–4458. IEEE, 2017.

[37] T. Vogels, S. P. Karimireddy, and M. Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization, 2020.

[38] H. Wang, S. Agarwal, and D. Papailiopoulos. Pufferfish: Communication-efficient models at no extra cost. *Proceedings of Machine Learning and Systems*, 3:365–386, 2021.

[39] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in Neural Information Processing Systems*, 33, 2020.

[40] Y. Xue and V. Lau. Riemannian low-rank model compression for federated learning with over-the-air aggregation. *IEEE Transactions on Signal Processing*, 71:2172–2187, 2023.

[41] K. Yang, T. Jiang, Y. Shi, and Z. Ding. Federated learning via over-the-air computation. *IEEE transactions on wireless communications*, 19(3):2022–2035, 2020.

[42] D. Yao, W. Pan, M. J. O'Neill, Y. Dai, Y. Wan, H. Jin, and L. Sun. Fedhm: Efficient federated learning for heterogeneous models via low-rank factorization, 2022.

[43] L. Yi, H. Yu, G. Wang, X. Liu, and X. Li. pfedlora: Model-heterogeneous personalized federated learning with lora tuning, 2024.

[44] E. Zangrando, S. Schotthöfer, G. Ceruti, J. Kusch, and F. Tudisco. Rank-adaptive spectral pruning of convolutional layers during training, 2023.

[45] J. Zhao, Z. Zhang, B. Chen, Z. Wang, A. Anandkumar, and Y. Tian. Galore: Memory-efficient llm training by gradient low-rank projection, 2024.

[46] H. Zhu, B. Chen, and C. Yang. Understanding why vit trains badly on small datasets: An intuitive perspective, 2023.

A Additional algorithms

In the following, we list a set of algorithms that are used in the paper as a contribution or as a baseline method. In particular, Algorithm 2 contains auxiliary function definitions for Algorithm 1 and Algorithm 5. Algorithm 3 is the standard FedAvg method as presented in [23]. Algorithm 4 is the FedLin Algorithm [24], i.e. the extension of Algorithm 4 with variance correction. Algorithm 5 represents the FeDLRT method with simplified variance correction, as analyzed in Theorem 4 and Corollary 1 with the additional Assumption 1.

Algorithm 2: Auxiliary functions

def `broadcast`($\{M_i\}_i$*: list of matrices*)**:**
  Send M_i from server to all clients $\forall i$
def `aggregate`($\{M_{c,i}\}_i$*: list of matrices*)**:**
  Send $M_{c,i}$ from client to server $\forall c, i$
  $M_i \leftarrow \frac{1}{C}\sum_{c=1}^{C} M_c \quad \forall i$
  return $\{M_i\}_i$;
def `coefficient_update_var_cor`(*c: client,* V_c*: correction term*)**:**
  for $s = 0, \dots, s_* - 1$ **do** `/* On client */`
    $\widetilde{S}_c^{s+1} \leftarrow \widetilde{S}_c^{s} - \lambda\left(\nabla_{\widetilde{S}}\mathcal{L}_c(\widetilde{U}_c\widetilde{S}_c^{s}\widetilde{V}_c^{\top}) + V_c\right)$
def `coefficient_update`(*c: client*)**:**
  for $s = 0, \dots, s_* - 1$ **do** `/* On client */`
    $\widetilde{S}_c^{s+1} \leftarrow \widetilde{S}_c^{s} - \lambda\nabla_{\widetilde{S}}\mathcal{L}_c(\widetilde{U}_c\widetilde{S}_c^{s}\widetilde{V}_c^{\top})$
def `basis_augmentation`(B*: old basis,* G_B*: basis dynamics*)**:**
  $[B \mid \bar{B}] \leftarrow \mathtt{qr}([B \mid G_B])$ `/* On server */`
  return $\bar{B}$

Algorithm 3: FedAvg [23]. (See Algorithm 2 for auxiliary function definitions)

Input : Initial values for weight matrix W
Client-server setup with clients $c = 1, \dots, C$.
for $t = 1, \dots, T$ **do**
  `broadcast`($\{W^t\}$)
  $W_c^{s=0} \leftarrow W^t$
  for $s = 0, \dots, s_* - 1$ **do**
    $W_c^{s+1} \leftarrow W_c^{s} - \lambda\nabla_W\mathcal{L}_c(W_c^{s})$ `/* Gradient descent on client */`
  $W^{t+1} \leftarrow$ `aggregate`($\{W_c^{s_*}\}$) `/* Aggregation on server */`

Algorithm 4: FedLin [24]. (See Algorithm 2 for auxiliary function definitions)

Input : Initial values for weight matrix W
Client-server setup with clients $c = 1, \dots, C$.
for $t = 1, \dots, T$ **do**
  `broadcast`($\{W^t\}$)
  $G_{W,c} \leftarrow \nabla_W\mathcal{L}_c(W^t)$ `/* Gradient computation on client */`
  $G_W \leftarrow$ `aggregate`($\{G_{W,c}\}$) `/* Aggregation on server */`
  `broadcast`($\{G_W\}$)
  $W_c^{s=0} \leftarrow W^t$
  $V_c \leftarrow G_W - G_{W,c}$ `/* Correction term computation on client */`
  for $s = 0, \dots, s_* - 1$ **do**
    $W_c^{s+1} \leftarrow W_c^{s} - \lambda\nabla_W\mathcal{L}_c(W_c^{s}) + V_c$ `/* Corrected iteration on client */`
  $W^{t+1} \leftarrow$ `aggregate`($\{W_c^{s_*}\}$) `/* Aggregation on server */`

Algorithm 5: FeDLRT with simplified variance correction. (See Algorithm 2 for auxiliary function definitions)

Input: Initial orthonormal bases $U^1, V^1 \in \mathbb{R}^{n\times r}$ and full rank $S^1 \in \mathbb{R}^{r\times r}$;
Client-server setup with clients $c = 1, \dots, C$;
τ: singular value threshold for rank truncation.

for $t = 1, \dots, T$ **do**
- $\texttt{broadcast}(\{U^t, V^t, S^t\})$
- $G_{U,c} \leftarrow \nabla_U \mathcal{L}_c(U^t S^t V^{t,\top})$ `/* On client */`
- $G_{V,c} \leftarrow \nabla_V \mathcal{L}_c(U^t S^t V^{t,\top})$ `/* On client */`
- $G_{S,c} \leftarrow \nabla_S \mathcal{L}_c(U^t S^t V^{t,\top})$ `/* On client */`
- $G_U, G_V, G_S \leftarrow \texttt{aggregate}(\{G_{U,c}, G_{V,c}, G_{S,c}\})$
- $\bar{U} \leftarrow \texttt{basis\_augmentation}(U^t, G_U)$, $\bar{V} \leftarrow \texttt{basis\_augmentation}(V^t, G_V)$
- $\texttt{broadcast}\left(\{\bar{U}, \bar{V}, G_S\}\right)$
- $\widetilde{U} \leftarrow [U^t \mid \bar{U}]$, $\widetilde{V} \leftarrow [V^t \mid \bar{V}]$ `/* Basis assembly on client */`
- $\widetilde{S}^{s=0} \leftarrow \begin{bmatrix} S^t & 0 \\ 0 & 0 \end{bmatrix}$ `/* Coefficient matrix assembly on client */`
- $\check{G}_{\widetilde{S},c} \leftarrow \begin{bmatrix} G_{S,c} & 0 \\ 0 & 0 \end{bmatrix}$ `/* Client coeff. gradient approximation on client */`
- $\check{G}_{\widetilde{S}} \leftarrow \begin{bmatrix} G_S & 0 \\ 0 & 0 \end{bmatrix}$ `/* Global coeff. gradient approximation on client */`
- $\texttt{coefficient\_update\_var\_cor}\left(c,\, \check{G}_{\widetilde{S}} - \check{G}_{\widetilde{S},c}\right)$ `/* On client */`
- $\widetilde{S}^* \leftarrow \texttt{aggregate}\left(\left\{\widetilde{S}_c^{s_*}\right\}\right)$
- $P_{r_1}, \Sigma_{r_1}, Q_{r_1} \leftarrow \texttt{svd}(\widetilde{S}^*)$ with threshold ϑ `/* Compression step */`
- $U^{t+1} \leftarrow \widetilde{U} P_{r_1}$, and $V^{t+1} \leftarrow \widetilde{V} Q_{r_1}$ `/* Basis projection */`
- $S^{t+1} \leftarrow \Sigma_{r_1}$

B Extension to convolutions and tensor-valued weights

FeDLRT can readily be extended to tensor-valued neural network layers, e.g. convolutional layers, following [44], where, e.g., a 2D convolution kernel is interpreted as an order-4 tensor and factorized by using the Tucker decomposition. To this end, the Tucker bases $U_i \in \mathbb{R}^{n_i \times r_i}$ for $i = 1, 2, 3, 4$ replace the U and V bases in the matrix case, and the Tucker core tensor $C \in \mathbb{R}^{r_1\times r_2\times r_3\times r_4}$ replaces the coefficient matrix S, to which the variance correction is applied. The analysis holds for the Tucker Tensor case, since Tucker Tensors have a manifold structure. In the analysis, one needs to consider the gradient projected upon all bases U_i instead of U and V. The compression step is performed with an truncated Tucker decomposition of the core tensor C, instead of an SVD of S. For intuition, one can also refer to the matrix case as the order-2 Tucker Tensor case. Remark that the bases U_i are all updated simultaneously, thus the adaption to the tensor case does not require more communication rounds.

C Additional numerical evaluation

C.1 Compute resources

The convex test cases are computed on a single Nvidia RTX 4090 GPU. The computer vision benchmarks use a set of Nvidia Tesla V100-SXM2-16GB and Tesla P100-PCIE-16GB. For prototyping, a Nvidia GTX1080ti is used.

C.2 Data augmentation

We use standard data augmentation techniques for the proposed test cases. That is, for CIFAR10, we augment the training data set by a random horizontal flip of the image, followed by a normal-

Algorithm 6: Naive implementation of FeDLRT. (See Algorithm 2 for auxiliary function definitions)

Input : Initial orthonormal bases $U^1, V^1 \in \mathbb{R}^{n\times r}$ and full rank $S^1 \in \mathbb{R}^{r\times r}$;
Client-server setup with clients $c = 1, \dots, C$;
τ: singular value threshold for rank truncation.

for $t = 1, \dots, T$ **do**
  $\texttt{broadcast}(\{U^t, V^t, S^t\})$
  $U_c^{s=0}, V_c^{s=0}, S_c^{s=0} \leftarrow U^t, V^t, S^t$
  for $s = 0, \dots, s_* - 1$ **do** `/* On client */`
    $G_{U,c} \leftarrow \nabla_U \mathcal{L}_c(U_c^s S_c^s V_c^{s,\top})$
    $G_{V,c} \leftarrow \nabla_V \mathcal{L}_c(U_c^s S_c^s V_c^{s,\top})$
    $\widetilde{U}_c, \_ \leftarrow \texttt{qr}([U_c^s \mid G_{U,c}])$
    $\widetilde{V}_c, \_ \leftarrow \texttt{qr}([V_c^s \mid G_{V,c}])$
    $\widetilde{S}_c = \widetilde{U}_c^\top U_c^s S_c^s V_c^{s,\top} \widetilde{V}_c$
    $\widetilde{S}_c^* \leftarrow \widetilde{S}_c - \lambda \nabla_{\widetilde{S}} \mathcal{L}_c(\widetilde{U}_c \widetilde{S}_c \widetilde{V}_c^\top)$
    $\widetilde{S}^* \leftarrow \texttt{aggregate}\left(\left\{\widetilde{S}_c^*\right\}\right)$
    $P_{r_1}, \Sigma_{r_1}, Q_{r_1} \leftarrow \texttt{svd}(\widetilde{S}^*)$ with threshold ϑ `/* Compression step */`
  $U^{t+1} \leftarrow \widetilde{U} P_{r_1}$, and $V^{t+1} \leftarrow \widetilde{V} Q_{r_1}$ `/* Basis projection */`
  $S^{t+1} \leftarrow \Sigma_{r_1}$

ization using mean $[0.4914, 0.4822, 0.4465]$ and std. dev. $[0.2470, 0.2435, 0.2616]$. The test data set is only normalized. The same augmentation is performed for CIFAR100, where with mean $[0.5071, 0.4867, 0.4408]$ and std. dev. $[0.2673, 0.2564, 0.2762]$.

C.3 Additional computer vision results

AlexNet on CIFAR10: We train AlexNet on CIFAR10, where the fully connected head of the network is replaced by a low-rank counterpart. A federated neural network setup with C clients trains on CTs_* random batches of the dataset, that is the number of seen training data batches scales with the client count. Figure 7 displays the validation accuracy of FeDLRT with variance correction compared to FedLin, where one can see that the performance of FeDLRT mirrors the performance of FedLin with more degrees of freedom. The measured validation accuracy peaks at $C = 4$ clients in both cases, where the higher number of seen training data-points offsets the negative effects of more clients on the validation performance. All reported runs are within close distance of the non-federated, full-rank baseline accuracy of 85.6%. Communication cost savings of the fully connected layers amount between 96% and 97% [6] We observe, similarly to the results in Section 4.1, that the maximum achieved communication cost savings, which depend on the layer ranks scales with the number of clients $C = 4$, indicating that the decay rate of the singular values of the averaged coefficient matrix $\widetilde{S}^*$ depends on C.

VGG16 on CIFAR10: We train AlexNet on CIFAR10, where the fully connected head of the network is replaced by a low-rank counterpart. A federated neural network setup with $240/C$ local iterations for C clients. Figure 8 displays the validation accuracy of FeDLRT with variance correction compared to FedLin, where one can see that the performance of FeDLRT mirrors the performance of FedLin with more degrees of freedom. All reported runs are within close distance of the non-federated, full-rank baseline accuracy of 85.6%. Communication cost savings of the fully connected layers amount between 96% and 97% [7] We observe, similarly results as in the ResNet18 test case.

VGG16 on CIFAR10 with low-rank convolutions: Mirroring the compute setup of the VGG16 test-case above, we now rewrite all convolutional layers of VGG16 as order 4 tensors in low-rank

[6] For clarity of exposition we consider only the fully connected layers. Taking into account the non low-rank convolution layers, the communication cost savings reduces to 87.5% to 87.3%.

[7] For clarity of exposition we consider only the fully connected layers. Taking into account the non low-rank convolution layers, the communication cost savings reduces to 87.5% to 87.3%.

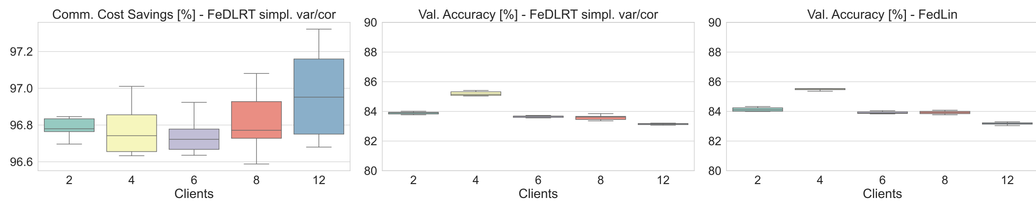

Figure 7: AlexNet CIFAR10 benchmark with fixed number of local iterations. (Left Panel) shows the savings in communication cost of simplified variance corrected FeDLRT vs FedLin. (Mid and right panel) compares the validation accuracy of FeDLRT and FedLin, where we see that FeDLRT behaves similarly to FedLin and achieves accuracy levels near the non-federated baseline value of 85.6%.

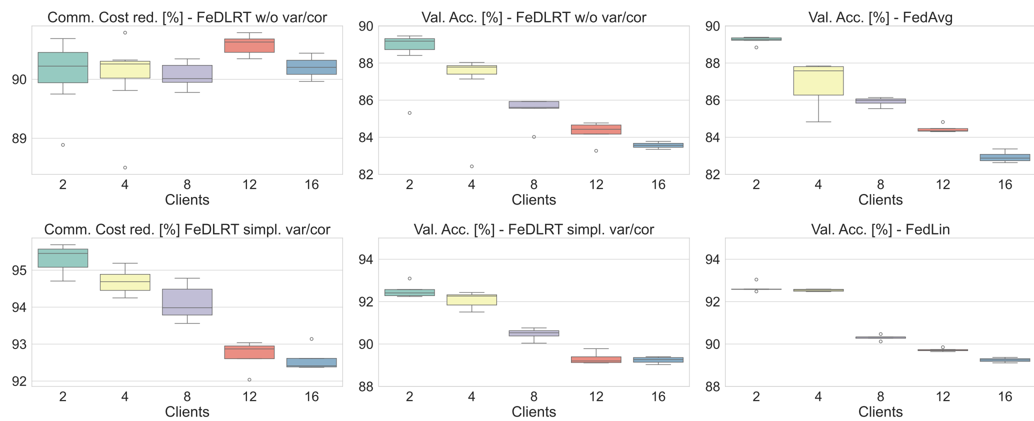

Figure 8: VGG16 CIFAR10 benchmark with $240/C$ local iterations for C clients with simplified (lower row) and without (upper row) variance correction. (Left panel) show the savings in communication cost corresponding to FedLin at final time. (Mid and right panel top row) compares the validation accuracy of FeDLRT and FedAvg, where we see that FeDLRT behaves similarly to FedAvg, where higher C correlates with a drop in accuracy. FeDLRT with variance correction mitigates this issue and achieves similar performance as FedLin, close to the non-federated baseline accuracy is 93.15%.

Tucker format, as described in appendix B. The full-connected head of the network is treated with the matrix low-rank method. The corresponding training results can be seen in Figure 9, and correspond well with the previous results for VGG16. The reduction of communication cost is slightly higher, due to the compression of the convolutions.

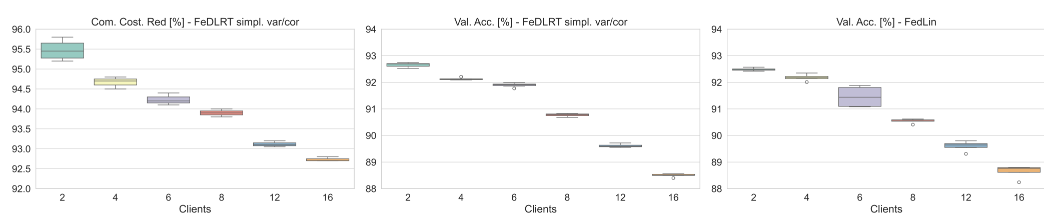

Figure 9: VGG16 CIFAR10, low-rank convolutional layers and low-rank fully connected layers. We report the communication cost savings and the validation accuracy of VGG16 with FeDLRT applied to training convolution and classifier layers. 2D convolutions are interpreted as an order-4 tensor and factorized in the Tucker format. The statistics over five random network initializations are reported using the training hyperparemeters of Table 2 of the main manuscript. The results are similar to Fig. 7 in the main manuscript, where only the classifier is compressed. Remark that here the classifier contains most of the network parameters.

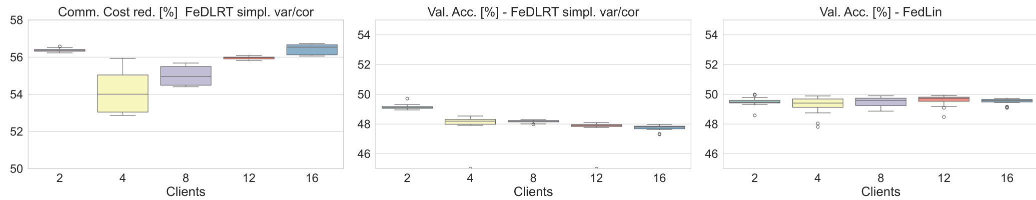

Figure 10: ViT CIFAR100 benchmark. (Left Panel) shows the savings in communication cost of variance corrected FeDLRT vs FedLin. (Mid and right panel) compares the validation accuracy of FeDLRT and FedLin, where we see that FeDLRT behaves similarly to FedLin and achieves accuracy levels near the non-federated baseline value of 50%, which is similar to literature results [46].

Table 2: Experimental setup object detection benchmarks. All test cases use a cosine annealing learning rate scheduler.

	Alexnet/Cifar10	ResNet18/Cifar10	VGG16/Cifar10	ViT/Cifar100
Batch size	128	128	128	256
Start Learningrate	1e$-$2	1e$-$3	1e$-$2	3e$-$4
End Learningrate	1e$-$5	5e$-$4	5e$-$4	1e$-$5
Aggregation Rounds	200	200	200	200
Local Iterations	100	$240/C$	$240/C$	$240/C$
Truncation tolerance τ	0.01	0.01	0.01	0.01
Momentum	0.0	0.9	0.1	n.a.
Weight Decay	1e$-$4	1e$-$3	1e$-$4	1e$-$2
Optimizer	SGD	SGD	SGD	Adam w/ std pytorch parameters

Vision Transformer on CIFAR100: We consider a small vision transformer for CIFAR100, with 6 attention layers with 2 heads each followed by a ResNet block and a drop-out layer, all with weight matrices of dimension 512×512. The tokenizer takes patches of size 8 with embedding dimension 512. Training hyperparameters are given in Table 2. Remark that we do not aim for SOTA performance, since transformer architectures are notoriously difficult to compress with low-rank approaches, but rather compare the performance of FedLin to FeDLRT for a given compute budget. We use $s_* = 240/C$ local iterations for C clients. Observe in Figure 10 that FeDLRT achieves similar performance as ViT with over 55% communication cost savings on average.

D Notation overview for the numerical analysis

We establish a set of notations to simplify the notation in the proofs

- $\mathcal{L}_c(W)$ denotes the local loss function based on dataset X_c at client c.
- $\mathcal{L}(W) = \frac{1}{C}\sum_{c=1}^{C} \mathcal{L}_c(W)$ is the global loss function.
- $F_c(W) = -\nabla_W \mathcal{L}_c(W)$ is the negate of local loss gradient.
- $F(W) = \frac{1}{C}\sum_{c=1}^{C} F_c(W)$ is the negate of global loss gradient.
- $\mathcal{M}_r = \{W \in \mathbb{R}^{n\times n} : \text{rank}(W) = r\}$ is a manifold of rank r matrices.
- $W_r = USV^\top \in \mathcal{M}_r$ is a rank-r approximation of a matrix W.
- $\mathcal{T}_{W_r}\mathcal{M}_r$ is the tangent space of $\mathcal{M}_r$ at W_r.
- $P(W_r)$ is the orthogonal projection onto $\mathcal{T}_{W_r}\mathcal{M}_r$.
- $P_U = UU^\top$ is the orthogonal projection onto the range of orthonormal $U \in \mathbb{R}^{n\times r}$.
- $P_V = VV^\top$ is the orthogonal projection onto the range of orthonormal $V \in \mathbb{R}^{n\times r}$.
- When applied to vectors, $\|\cdot\|$ denotes the Euclidean norm (ℓ_2-norm). When applied to matrices, $\|\cdot\|$ denotes the Frobenius norm.

E Efficient basis gradient dynamics for basis augmentation

We first consider the basis update & Galerkin splitting scheme of (5). The splitting performs a reparametrization of the form $K(t) = U(t)S(t)$ and $L(t) = V(t)S(t)^\top$. The basis update then reads

$$\begin{aligned} \dot{K} &= -\nabla_K \mathcal{L}(K(t)V_0^\top) \in \mathbb{R}^{n\times r}, \quad K(0) = U_0 S_0, \\ \dot{L} &= -\nabla_L \mathcal{L}(U_0 L(t)^\top) \in \mathbb{R}^{n\times r}, \quad L(0) = V_0 S_0^\top. \end{aligned} \tag{15}$$

Given the solution $K(t_1)$ and $L(t_1)$ at time t_1, the bases U_0 and V_0 are augmented by the orthonormalization of the new directions $K(t_1)$ and $L(t_1)$, i.e.

$$\begin{aligned} &\widetilde{U}R = \texttt{qr}([U_0 \mid K(t_1)]) \in \mathbb{R}^{n\times 2r}, \\ \text{and} \quad &\widetilde{V}R = \texttt{qr}([V_0 \mid L(t_1)]) \in \mathbb{R}^{n\times 2r}, \end{aligned} \tag{16}$$

where R is the right factor of the respective QR decomposition and can be discarded. The initial condition of the coefficient update is $S(t_0)$ projected onto the new bases, i.e.,

$$\dot{\widetilde{S}} = -\nabla_S \mathcal{L}(\widetilde{U}\widetilde{S}(t)\widetilde{V}^\top), \quad \widetilde{S}(0) = \widetilde{U}^\top U_0 \widetilde{S}(0) V_0^\top \widetilde{V}. \tag{17}$$

After the integration of the coefficient dynamics above, the redundant basis functions are typically truncated via an SVD of S ensuring that S is always full rank. In its continuous form above, the splitting yields a robust integrator for the projected gradient flow, without manifold dependent step-size restrictions:

Theorem 5. *([32]) Assume $\mathcal{L}$ is L-smooth with constant L, and locally bounded by B. Let $W_r(t)$ be the low-rank continuous time solution of (15) and (17) and let $W(t)$ be the full rank solution at $t = 0$. Assume the $K, L,$ and S equations are integrated exactly from time $t = 0$ to Δt. Assume that for any $Y \in \mathcal{M}_r$ sufficiently close to $W_r(t)$ the gradient $F(Y)$ is ϵ close to $\mathcal{M}_r$. Then*

$$\|W(\Delta t) - W_r(\Delta t)\| \leq d_1\epsilon + d_2\Delta t + d_3\frac{\vartheta}{\Delta t},$$

where d_1, d_2, d_3 depend only on L and B.

The theorem guarantees, that the low-rank representation does not imply any step-size restrictions on the optimization scheme. This is in stark contrast to a naive alternating descent optimization of the low-rank factors U, S, V.

To build an discretized numerical optimizer in a resource constrained federated scenario from the above continuous splitting equations, we avoid the reparametrization, which implies a 200% memory cost increase on the client side, since three versions of the low-rank layer need to be tracked.

Lemma 2. *Let $USV \in \mathcal{M}_r$ be a low rank factorization that follows the projected gradient (5) flow using the splitting scheme (15) with $K = US$ and $V = VS^\top$. Further, assume that equations for the*

K and L factors are solved by an explicit Euler time integration with learning rate λ, i.e.

$$\begin{aligned} K(t_1) &= K(0) - \lambda \nabla_K \mathcal{L}(K(0)V_0^\top), \quad K(0) = U_0 S_0, \\ L(t_1) &= L(0) - \lambda \nabla_L \mathcal{L}(U_0 L(0)^\top), \quad L(0) = V_0 S_0^\top. \end{aligned} \tag{18}$$

Then, the basis augmentation (16) can be expressed as

$$\begin{aligned} \widetilde{U}R &= \mathtt{qr}([U_0 \mid -\nabla_U \mathcal{L}\big(U_0 S_0 V_0^\top)]) \in \mathbb{R}^{n\times 2r}, \\ \text{and} \quad \widetilde{V}R &= \mathtt{qr}([V_0 \mid -\nabla_V \mathcal{L}\big(U_0 S_0 V_0^\top)]) \in \mathbb{R}^{n\times 2r}. \end{aligned} \tag{19}$$

and maintains the structure of the basis update and Galerkin operator split.

Proof. We consider the proof for the K equation and the U basis; the proof for L and V follows analogously.

Considering (16), we obtain with the explicit Euler discretization (18),

$$\begin{aligned} \operatorname{span}([U_0 \mid K(t_1)]) &= \operatorname{span}\big([U_0 \mid U_0 - \lambda \nabla_K \mathcal{L}(K(0)V_0^\top)]\big) \\ &= \operatorname{span}\big([U_0 \mid -\lambda \nabla_K \mathcal{L}(K(0)V_0^\top)]\big) \\ &= \operatorname{span}\big([U_0 \mid -\nabla_K \mathcal{L}(K(0)V_0^\top)]\big)\,. \end{aligned} \tag{20}$$

Next, consider the continuous time dynamics of $\dot{K}$, where we omit explicit time dependence on U, S, V and K for the sake of brevity, i.e.,

$$\begin{aligned} \dot{K} &= (\dot{US}) \\ &= \dot{U}S + U\dot{S} \\ &\overset{(5)}{=} -(I - UU^\top)\nabla_W \mathcal{L}(USV^\top)VS^{-1}S - UU^\top \nabla_W \mathcal{L}(USV^\top)V \\ &= -(I - P_U)\nabla_W \mathcal{L}(USV^\top)V - P_U \nabla_W \mathcal{L}(USV^\top)V \\ &= (P_U - I)\nabla_W \mathcal{L}(USV^\top)V - P_U \nabla_W \mathcal{L}(USV^\top)V \\ &= -\nabla_W \mathcal{L}(USV^\top)V \end{aligned} \tag{21}$$

Further, using the chain rule, we observe

$$\nabla_U \mathcal{L}(USV^\top) = \nabla_W \mathcal{L}(USV^\top)\nabla_U(USV^\top) = \nabla_W \mathcal{L}(USV^\top)VS^\top$$

Thus, $-\nabla_U \mathcal{L}(USV^\top)S^{-\top} = -\nabla_W \mathcal{L}(USV^\top)V = \dot{K}$. Full rankness of S and (21) yield that $\operatorname{span}(-\nabla_U \mathcal{L}(USV^\top)) = \operatorname{span}(\dot{K})$. Together with (20) this yields the proof. □

Lemma 2 adopts a more general result for Tucker tensors in an unpublished manuscript and simplifies the analysis for the matrix case considered here.

F Efficient basis and coefficient communication

Note that we have by orthogonality of the bases $\widetilde{U} = [U, \bar{U}]$ with $\bar{U} \in \mathbb{R}^{n\times r}$ and $\bar{U}^\top U = 0$ and $\widetilde{V} = [V, \bar{V}]$ with $\bar{V} \in \mathbb{R}^{n\times r}$ and $\bar{V}^\top V = 0$.

Proof. (Lemma 1) The basis augmented basis $[U, G_U]$ before orthonormalization already contains the orthonormal vectors given by the columns of U. A QR decomposition therefor only rearranges the columns of G_U such that $\widetilde{U} = [U, \bar{U}]$ with $\bar{U} \in \mathbb{R}^{n\times r}$ and $\bar{U}^\top U = 0$. The analogous result holds for $\widetilde{V} = [V, \bar{V}]$. The projection onto the augmented basis therefore reads

$$\widetilde{U}^\top U = \begin{bmatrix} U^\top U \\ \overline{U}^\top U \end{bmatrix} = \begin{bmatrix} I \\ 0 \end{bmatrix} \quad \text{and} \quad \widetilde{V}^\top V = \begin{bmatrix} V^\top V \\ \overline{V}^\top V \end{bmatrix} = \begin{bmatrix} I \\ 0 \end{bmatrix}. \tag{22}$$

Consequently, the augmented coefficient matrix takes the form

$$\widetilde{S} = \widetilde{U}^\top USV^\top \widetilde{V} = \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix}. \tag{23}$$

□

G Analysis for FeDLRT with full variance correction

In this section we establish bounds on the coefficient drift of the FeDLRT method with full variance correction. We use the established coefficient drift bound to derive a loss-descend guarantee. The strategy of our analysis follows the one of FedLin [24]. We first state an auxiliary lemma.

Lemma 3. *Let $U \in \mathbb{R}^{n\times r}$ and $V \in \mathbb{R}^{n\times r}$ be orthonormal matrices. Let F be an L-continuous function. Then, for $S_1, S_2 \in \mathbb{R}^{r\times r}$,*

$$\left\|P_U\left(F(US_1V^\top) - F(US_2V^\top)\right)P_V\right\| \leq L\left\|S_1 - S_2\right\| \tag{24}$$

and

$$\left\|U\left(F(US_1V^\top) - F(US_2V^\top)\right)V^\top\right\| \leq L\left\|S_1 - S_2\right\|, \tag{25}$$

where P_U and P_V are orthogonal projections defined in Appendix D.

Proof. For the first statement, consider

$$\begin{aligned}
&\left\|P_U\left(F(US_1V^\top) - F(US_2V^\top)\right)P_V\right\| \\
=&\left\|UU^\top\left(F(US_1V^\top) - F(US_2V^\top)\right)VV^\top\right\| \\
\overset{\text{(I)}}{\leq}&\left\|U\right\|\left\|U^\top\right\|\left\|F(US_1V^\top) - F(US_2V^\top)\right\|\left\|V\right\|\left\|V^\top\right\| \\
\overset{\text{(II)}}{=}&\left\|F(US_1V^\top) - F(US_2V^\top)\right\| \\
\overset{\text{(III)}}{\leq}&L\left\|US_1V^\top - US_2V^\top\right\| = L\left\|U(S_1 - S_2)V^\top\right\| \\
\overset{\text{(I)}}{\leq}&L\left\|U\right\|\left\|S_1 - S_2\right\|\left\|V^\top\right\| \\
\overset{\text{(II)}}{=}&L\left\|S_1 - S_2\right\|,
\end{aligned}$$

where we have used in (I) the operator norm inequality of the Frobenius norm, in (II) orthonormality of U, V, and in (III) L-continuity of F. The second statement is proven analogously. ☐

G.1 Coefficient drift bound for FeDLRT with full variance correction

We consider the FeDLRT method with variance correction, see Algorithm 1. Key difference to the FeDLRT method without variance correction is the modified coefficient update, incorporating global gradient information of the augmented coefficient matrix $\widetilde{S}$ and local, stale gradient information of the augmented coefficient matrix $\widetilde{S}_c$. The variance corrected local coefficient update (8) can be expressed in terms of the projected Riemannian gradient as

$$\widetilde{S}_c^{s+1} = \widetilde{S}_c^s + \lambda\widetilde{U}^\top\left(F_c(\widetilde{W}_{r,c}^s) - F_c(\widetilde{W}_r) + F(\widetilde{W}_r)\right)\widetilde{V}, \tag{26}$$

where $\widetilde{U}^\top F_c(\widetilde{W}_{r,c}^s)\widetilde{V} = \nabla_{\widetilde{S}_c}\mathcal{L}_c(\widetilde{U}\widetilde{S}_c^s\widetilde{V})$, $\widetilde{U}^\top F_c(\widetilde{W}_{r,c})\widetilde{V} = \nabla_{\widetilde{S}_c}\mathcal{L}_c(\widetilde{U}\widetilde{S}_c^{s=0}\widetilde{V})$ and $\widetilde{U}^\top F_c(\widetilde{W}_{r,c}^s)\widetilde{V} = \nabla_{\widetilde{S}_c}\mathcal{L}(\widetilde{U}\widetilde{S}_c^s\widetilde{V})$. Recall that $\widetilde{S} = \widetilde{S}_c$ for $s = 0$.

We provide proof for Theorem 1 to bound the drift term $\left\|\widetilde{S}_c^s - \widetilde{S}_c\right\|$. We restate this theorem to the Riemannian notation and restate it below.

Theorem 6. *(Restatement of Theorem 1) Given augmented basis and coefficient matrices $\widetilde{U}$, $\widetilde{V}$, and $\widetilde{S}$, and $\widetilde{W}_r = \widetilde{U}\widetilde{S}\widetilde{V}^\top$. If the local learning rate $0 < \lambda \leq \frac{1}{Ls_*}$ with $s_* \geq 1$ the number of local steps, for all clients c,*

$$\|\widetilde{S}_c^s - \widetilde{S}_c\| \leq \exp(1)s_*\lambda\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\|, \quad \text{for} \quad s = 1, \ldots, s^* - 1, \tag{27}$$

where $\widetilde{S}_c^s$ is the variance corrected coefficient as given in (8).

Proof. From the adjusted coefficient update in (26), we get

$$\begin{aligned}
\left\|\widetilde{S}_c^{s+1}-\widetilde{S}_c\right\| &= \left\|\widetilde{S}_c^{s}-\widetilde{S}_c+\lambda\widetilde{U}^\top\left(F_c(\widetilde{W}_{r,c}^s)-F_c(\widetilde{W}_r)+F(\widetilde{W}_r)\right)\widetilde{V}\right\| \\
&\le \left\|\widetilde{S}_c^{s}-\widetilde{S}_c\right\|+\lambda\left\|\widetilde{U}^\top\left(F_c(\widetilde{W}_{r,c}^s)-F_c(\widetilde{W}_r)\right)\widetilde{V}\right\|+\lambda\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\| \\
&\overset{(\mathrm{I})}{\le} \left\|\widetilde{S}_c^{s}-\widetilde{S}_c\right\|+\lambda L\left\|\widetilde{S}_c^{s}-\widetilde{S}\right\|+\lambda\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\| \\
&\le (1+\lambda L)\left\|\widetilde{S}_c^{s}-\widetilde{S}\right\|+\lambda\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\| \\
&\le \left(1+\frac{1}{s_*}\right)\left\|\widetilde{S}_c^{s}-\widetilde{S}\right\|+\lambda\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\|.
\end{aligned}$$

We use in (I) Lemma 3 Recursively plugging in the above inequality yields for $a=(1+\frac{1}{s_*})$

$$\begin{aligned}
\left\|\widetilde{S}_c^{s+1}-\widetilde{S}_c\right\| &\le a^{s+1}\left\|\widetilde{S}_c^{s=0}-\widetilde{S}\right\|+\left(\sum_{j=0}^{s}a^j\right)\lambda\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\| \\
&= \left(\sum_{j=0}^{s}a^j\right)\lambda\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\| \\
&= \frac{a^{s+1}-1}{a-1}\lambda\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\| \\
&\le \left(1+\frac{1}{s_*}\right)^{s+1}s_*\lambda\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\| \\
&\le \left(1+\frac{1}{s_*}\right)^{s_*}s_*\lambda\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\| \\
&\le \exp(1)s_*\lambda\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\|.
\end{aligned}$$

□

G.2 Global loss descend for FeDLRT with full variance correction

We first state a few auxiliary lemmas, which provide common inequalities that will be used in the following analysis.

Lemma 4. *([10, Lemma 5.2]) For any two matrices $Y_1, Y_2 \in \mathbb{R}^{n\times n}$ and an L-smooth $\mathcal{L}$ with constant L it holds*

$$\mathcal{L}(Y_1)-\mathcal{L}(Y_2)\le -\langle Y_1-Y_2, F(Y_2)\rangle+\frac{L}{2}\left\|Y_1-Y_2\right\|^2, \tag{28}$$

where $F(Y)=-\nabla_Y\mathcal{L}(Y)$.

Lemma 5. *([25, Lemma 5]) For two vectors $x_1, x_2\in\mathbb{R}^d$ it holds for $\gamma>0$*

$$\left\|x_1+x_2\right\|^2\le(1+\gamma)\left\|x_1\right\|^2+\left(1+\frac{1}{\gamma}\right)\left\|x_2\right\|^2. \tag{29}$$

Lemma 6. *([25, Lemma 6]) For C vectors $x_1,\dots,x_C\in\mathbb{R}^d$ the application of Jensen's inequality yields*

$$\left\|\sum_{c=1}^{C}x_c\right\|^2\le C\sum_{c=1}^{C}\left\|x_c\right\|^2. \tag{30}$$

First, we consider the loss function value at the augmentation step.

Lemma 7. *We have $\mathcal{L}(\widetilde{W}_r)=\mathcal{L}(W_r^t)$ for the loss before and after basis augmentation.*

Proof. Due to Lemma 1, $\widetilde{S} = \begin{bmatrix} S^t & 0 \\ 0 & 0 \end{bmatrix}$, thus $\widetilde{W}_r = \widetilde{U}\widetilde{S}\widetilde{V}^\top = USV^\top = W^t$. $\square$

We next bound the loss descent between the augmentation step and the truncation step - having performed the aggregation of the client updates.

Theorem 7. *Let $\widetilde{W}_r = \widetilde{U}\widetilde{S}\widetilde{V}^\top$ be the augmented factorization at global iteration t and let $\widetilde{W}_r^* = \widetilde{U}\widetilde{S}^*\widetilde{V}^\top$ be the aggregated solution after client iterations, i.e., $\widetilde{S}^* = \frac{1}{C}\sum_{c=1}^{C} \widetilde{S}_c^{s_*}$. Then the variance corrected coefficient update (26) yields the guarantee*

$$\begin{aligned}\mathcal{L}(\widetilde{W}_r^*) - \mathcal{L}(\widetilde{W}_r) \leq& -(s_*\lambda)(1-(s_*\lambda)L)\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\|^2 \\ &+ \left(\frac{L\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1}\left\|\widetilde{S}_c^s - \widetilde{S}\right\|\right)\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\| \\ &+ \frac{L^3\lambda^2 s_*}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1}\left\|\widetilde{S}_c^s - \widetilde{S}_c\right\|^2.\end{aligned} \tag{31}$$

Proof. From (8), $P_{\widetilde{U}} = \widetilde{U}\widetilde{U}^\top$, $P_{\widetilde{V}} = \widetilde{V}\widetilde{V}^\top$, and the fact that $\widetilde{W}_{r,c}^{s=0} = \widetilde{W}_r$ for all $c = 1, \dots, C$,

$$\begin{aligned}\widetilde{W}_{r,c}^{s_*} = \widetilde{U}\widetilde{S}_c^{s_*}\widetilde{V}^\top &= \widetilde{U}\widetilde{S}_c^{s=0}\widetilde{V}^\top + \widetilde{U}\widetilde{U}^\top\sum_{s=0}^{s_*-1}\lambda\left(F_c(\widetilde{W}_{r,c}^s) - F_c(\widetilde{W}_r) + F(\widetilde{W}_r)\right)\widetilde{V}\widetilde{V}^\top \\ &= \widetilde{W}_r - \lambda\sum_{s=0}^{s_*-1}P_{\widetilde{U}}F_c(\widetilde{W}_{r,c}^s)P_{\widetilde{V}} - \lambda P_{\widetilde{U}}\left(F(\widetilde{W}_r) - F_c(\widetilde{W}_r)\right)P_{\widetilde{V}}.\end{aligned}$$

Averaging across clients leads to

$$\begin{aligned}\widetilde{W}_r^* =& \frac{1}{C}\sum_{c=1}^{C}\widetilde{W}_{r,c}^{s_*} = \widetilde{W}_r - \frac{\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1}P_{\widetilde{U}}F_c(\widetilde{W}_{r,c}^s)P_{\widetilde{V}} - \frac{\lambda}{C}\sum_{c=1}^{C}P_{\widetilde{U}}\left(F(\widetilde{W}_r) - F_c(\widetilde{W}_r)\right)P_{\widetilde{V}} \\ =& \widetilde{W}_r - \frac{\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1}P_{\widetilde{U}}F_c(\widetilde{W}_{r,c}^s)P_{\widetilde{V}},\end{aligned} \tag{32}$$

where we have used the definition of the global and local gradient at $\widetilde{W}_r$, i.e., $\frac{1}{C}\sum_{c=1}^{C}F_c(\widetilde{W}_r) = F(\widetilde{W}_r)$. Based on L-continuity of F and F_c, (32), and Lemma 4, we obtain further

$$\begin{aligned}\mathcal{L}(\widetilde{W}_r^*) - \mathcal{L}(\widetilde{W}_r) &\leq \left\langle \widetilde{W}_r^* - \widetilde{W}_r, F(\widetilde{W}_r)\right\rangle + \frac{L}{2}\left\|\widetilde{W}_r^* - \widetilde{W}_r\right\|^2 \qquad (33)\\ &= -\left\langle \frac{\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1}P_{\widetilde{U}}F_c(\widetilde{W}_{r,c}^s)P_{\widetilde{V}}, F(\widetilde{W}_r)\right\rangle + \frac{L}{2}\left\|\frac{\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1}P_{\widetilde{U}}F_c(\widetilde{W}_{r,c}^s)P_{\widetilde{V}}\right\|^2.\end{aligned}$$

Next, we bound each of the two right-hand-side terms separately. We first express the first term as

$$
\begin{aligned}
& -\left\langle \frac{\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1} P_{\widetilde{U}} F_c(\widetilde{W}_{r,c}^s) P_{\widetilde{V}}, F(\widetilde{W}_r)\right\rangle \\
=& -\left\langle \frac{\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1} P_{\widetilde{U}} \left(F_c(\widetilde{W}_{r,c}^s) - F_c(\widetilde{W}_r)\right) P_{\widetilde{V}} + P_{\widetilde{U}}\left(\frac{\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1} F_c(\widetilde{W}_r)\right) P_{\widetilde{V}}, F(\widetilde{W}_r)\right\rangle \\
=& -\left\langle \frac{\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1} P_{\widetilde{U}} \left(F_c(\widetilde{W}_{r,c}^s) - F_c(\widetilde{W}_r)\right) P_{\widetilde{V}} + P_{\widetilde{U}}\frac{s_*\lambda}{C}\sum_{c=1}^{C} F_c(\widetilde{W}_r) P_{\widetilde{V}}, F(\widetilde{W}_r)\right\rangle \\
=& -\left\langle P_{\widetilde{U}}\left(\frac{\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1} F_c(\widetilde{W}_{r,c}^s) - F_c(\widetilde{W}_r)\right) P_{\widetilde{V}} + P_{\widetilde{U}} s_*\lambda F(\widetilde{W}_r) P_{\widetilde{V}}, F(\widetilde{W}_r)\right\rangle \\
=& -\left\langle \widetilde{U}^\top\left(\frac{\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1} F_c(\widetilde{W}_{r,c}^s) - F_c(\widetilde{W}_r)\right)\widetilde{V}, \widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}^\top\right\rangle - s_*\lambda\left\langle \widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}, \widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\rangle \\
=& -\left\langle \frac{\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1} \widetilde{U}^\top\left(F_c(\widetilde{W}_{r,c}^s) - F_c(\widetilde{W}_r)\right)\widetilde{V}, \widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\rangle - s_*\lambda\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\|^2,
\end{aligned}
$$

where the definitions of $P_{\widetilde{U}}$ and $P_{\widetilde{V}}$ are used. Following this, the first term then can be bounded by

$$
\begin{aligned}
& -\left\langle \frac{\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1} P_{\widetilde{U}} F_c(\widetilde{W}_{r,c}^s) P_{\widetilde{V}}, F(\widetilde{W}_r)\right\rangle \\
\leq & \frac{\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1}\left\|\widetilde{U}^\top\left(F_c(\widetilde{W}_{r,c}^s) - F_c(\widetilde{W}_r)\right)\widetilde{V}\right\|\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\| - s_*\lambda\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\|^2 \\
\leq & \frac{L\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1}\left\|\widetilde{S}_c^s - \widetilde{S}\right\|\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\| - s_*\lambda\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\|^2,
\end{aligned}
$$

where Lemma 3 is invoked in the last inequality. Following a similar approach, we express the second term as

$$
\frac{L}{2}\left\|\frac{\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1} P_{\widetilde{U}} F_c(\widetilde{W}_{r,c}^s) P_{\widetilde{V}}\right\|^2 = \frac{L}{2}\left\|\frac{\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1} P_{\widetilde{U}}\left(F_c(\widetilde{W}_{r,c}^s) - F_c(\widetilde{W}_r)\right) P_{\widetilde{V}} + s_*\lambda P_{\widetilde{U}} F(\widetilde{W}_r) P_{\widetilde{V}}\right\|^2,
$$

which can be bounded by

$$
\begin{aligned}
& \frac{L}{2}\left\|\frac{\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1} P_{\widetilde{U}} F_c(\widetilde{W}_{r,c}^s) P_{\widetilde{V}}\right\|^2 \\
\overset{\text{(I)}}{\leq} & L\left\|\frac{\lambda}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1} P_{\widetilde{U}}\left(F_c(\widetilde{W}_{r,c}^s) - F_c(\widetilde{W}_r)\right) P_{\widetilde{V}}\right\|^2 + (s_*\lambda)^2 L\left\|P_{\widetilde{U}} F(\widetilde{W}_r) P_{\widetilde{V}}\right\|^2 \\
\overset{\text{(II)}}{\leq} & \frac{L}{C}\sum_{c=1}^{C}\lambda^2 s_*\sum_{s=0}^{s_*-1}\left\|P_{\widetilde{U}}\left(F_c(\widetilde{W}_{r,c}^s) - F_c(\widetilde{W}_r)\right) P_{\widetilde{V}}\right\|^2 + (s_*\lambda)^2 L\left\|P_{\widetilde{U}} F(\widetilde{W}_r) P_{\widetilde{V}}\right\|^2 \\
\overset{\text{(III)}}{\leq} & \frac{L^3\lambda^2 s_*}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1}\left\|\widetilde{S}_c^s - \widetilde{S}_c\right\|^2 + (s_*\lambda)^2 L\left\|P_{\widetilde{U}} F(\widetilde{W}_r) P_{\widetilde{V}}\right\|^2 \\
\overset{\text{(IV)}}{\leq} & \frac{L^3\lambda^2 s_*}{C}\sum_{c=1}^{C}\sum_{s=0}^{s_*-1}\left\|\widetilde{S}_c^s - \widetilde{S}_c\right\|^2 + (s_*\lambda)^2 L\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\|^2,
\end{aligned}
$$

where Lemma 5 with $\gamma = 1$ is used in in (I), Jensen's inequality is used in (II), Lemma 3 is used in in (III), and (IV) follows from the Operator norm inequality of the Frobenius norm in combination with orthonormality of U and $V^\top$.

Plugging these two bounds into (33) gives

$$\begin{aligned}
\mathcal{L}(\widetilde{W}_r^*) - \mathcal{L}(\widetilde{W}_r) \leq & - \left\langle \frac{\lambda}{C} \sum_{c=1}^{C} \sum_{s=0}^{s_*-1} P_{\widetilde{U}} F_c(\widetilde{W}_{r,c}^s) P_{\widetilde{V}}, F(\widetilde{W}_r) \right\rangle + \frac{L}{2} \left\| \frac{\lambda}{C} \sum_{c=1}^{C} \sum_{s=0}^{s_*-1} P_{\widetilde{U}} F_c(\widetilde{W}_{r,c}^s) P_{\widetilde{V}} \right\|^2 \\
\leq & \frac{L\lambda}{C} \sum_{c=1}^{C} \sum_{s=0}^{s_*-1} \left\| \widetilde{S}_c^s - \widetilde{S} \right\| \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\| - s_* \lambda \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2 \\
& + \frac{L^3 \lambda^2 s_*}{C} \sum_{c=1}^{C} \sum_{s=0}^{s_*-1} \left\| \widetilde{S}_c^s - \widetilde{S}_c \right\|^2 + (s_*\lambda)^2 L \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2 \\
= & - (s_*\lambda)(1 - (s_*\lambda)L) \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2 \\
& + \left(\frac{L\lambda}{C} \sum_{c=1}^{C} \sum_{s=0}^{s_*-1} \left\| \widetilde{S}_c^s - \widetilde{S} \right\| \right) \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\| \\
& + \frac{L^3 \lambda^2 s_*}{C} \sum_{c=1}^{C} \sum_{s=0}^{s_*-1} \left\| \widetilde{S}_c^s - \widetilde{S}_c \right\|^2,
\end{aligned}$$

which concludes the proof. □

With this result, we next bound the loss descent between the augmentation and coefficient aggregation step in the following theorem.

Theorem 8. *Under the same assumptions as in Theorem 7. Let the local learning rate be $0 < \lambda \leq \frac{1}{12 L s_*}$ with number of local iterations $s_* \geq 1$. Then,*

$$\mathcal{L}(\widetilde{W}_r^*) - \mathcal{L}(\widetilde{W}_r) \leq -s_* \lambda (1 - 12 s_* \lambda L) \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2. \tag{34}$$

Proof. Applying the drift bound given in Theorem 1 to the loss descent bound given by Theorem 7 in (31) leads to

$$\begin{aligned}
& - (s_*\lambda)(1 - (s_*\lambda)L) \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2 \\
& + \left(\frac{L\lambda}{C} \sum_{c=1}^{C} \sum_{s=0}^{s_*-1} \left(\exp(1) s_* \lambda \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\| \right) \right) \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\| \\
& + \frac{L^3 \lambda^2 s_*}{C} \sum_{c=1}^{C} \sum_{s=0}^{s_*-1} \left(\exp(1) s_* \lambda \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\| \right)^2 \\
= & - (s_*\lambda)(1 - (s_*\lambda)L) \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2 + L \lambda^2 s_*^2 \exp(1) \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2 \\
& + L^3 \lambda^4 s_*^4 \exp(2) \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2 \\
= & - (s_*\lambda)(1 - (s_*\lambda)L - (s_*\lambda)L \exp(1) - (s_*\lambda)^3 L^2 \exp(2)) \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2 \\
\leq & - (s_*\lambda)(1 - (s_*\lambda)L(1 + \exp(1) + \exp(2))) \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2 \\
\leq & - (s_*\lambda)(1 - 12(s_*\lambda)L) \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2,
\end{aligned}$$

where we have used that $(s_*\lambda)L \leq 1$ and that $1 + \exp(1) + \exp(2) \approx 11.107 \leq 12$. □

We are now prepared to prove Theorem 2, which we restate in terms of Riemannian gradients as below.

Theorem 9. *(Restatement of Theorem 2) Let $U^t S^t V^{t,\top}$ and $U^{t+1} S^{t+1} V^{t+1,\top}$ be the factorization before and after iteration t of Algorithm 1 with variance correction and singular value truncation* *threshold ϑ. Let $\mathcal{L}_c$ and $\mathcal{L}$ be L-smooth with constant L, and let the local learning rate be $0 \leq \lambda \leq \frac{1}{12Ls_*}$. Then the global loss descent is bounded by*

$$\mathcal{L}(U^{t+1}S^{t+1}V^{t+1,\top}) - \mathcal{L}(U^tS^tV^{t,\top}) \leq -(s_*\lambda)(1-12(s_*\lambda)L)\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\|^2 + L\vartheta. \quad (35)$$

Proof. Consider $\mathcal{L}(W_r^{t+1})$ and $\mathcal{L}(\widetilde{W}_r^*)$, i.e., the loss values before and after the truncation step. By the mean value theorem, we obtain for some $h \in [0,1]$

$$\begin{aligned}\mathcal{L}(W_r^{t+1}) &= \mathcal{L}(\widetilde{W}_r^*) + \left\langle -F(hW_r^{t+1} + (1-h)\widetilde{W}_r^*), W_r^{t+1} - \widetilde{W}_r^*\right\rangle \\ &\leq \mathcal{L}(\widetilde{W}_r^*) + \left\|F(hW_r^{t+1} + (1-h)\widetilde{W}_r^*)\right\|\left\|W_r^{t+1} - \widetilde{W}_r^*\right\| \\ &\leq \mathcal{L}(\widetilde{W}_r^*) + L\vartheta\end{aligned} \quad (36)$$

where L-smoothness and the fact that $\vartheta \geq \left\|W_r^{t+1} - \widetilde{W}_r^*\right\|$ are used in (II), where the latter follows from the singular value truncation threshold. Combining the above arguments with Lemma 7 and Theorem 8 yields

$$\begin{aligned}\mathcal{L}(W_r^{t+1}) - \mathcal{L}(W_r^t) &= (\mathcal{L}(W_r^{t+1}) - \mathcal{L}(\widetilde{W}_r^*)) + (\mathcal{L}(\widetilde{W}_r^*) - \mathcal{L}(\widetilde{W}_r)) + (\mathcal{L}(\widetilde{W}_r) - \mathcal{L}(W_r^t)) \\ &\leq L\vartheta - (s_*\lambda)(1-12(s_*\lambda)L)\left\|\widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V}\right\|^2,\end{aligned}$$

which concludes the proof. □

G.3 Global convergence of FeDLRT with full variance correction

Theorem 10. *(Restatement of Theorem 3) Assume that $\mathcal{L}$ is L-smooth with constant L for all $c = 1, \ldots, C$. Let $\widetilde{U}^t\widetilde{S}^t\widetilde{V}^{t,\top}$ be the augmented representation at iteration t. Then Algorithm 1 guarantees for the learning rate $\lambda \leq \frac{1}{12Ls_*}$ and final iteration T*

$$\min_{t=1,\ldots,T}\left\|\nabla_{\widetilde{S}}\mathcal{L}(U^tS^tV^{t,\top})\right\|^2 \leq \frac{48L}{T}\left(\mathcal{L}({W_r}^{t=1}) - \mathcal{L}({W_r}^{t=T+1})\right) + 48L^2\vartheta. \quad (37)$$

Proof. Consider Theorem 2,

$$\mathcal{L}(W_r^{t+1}) - \mathcal{L}(W_r^t) \leq L\vartheta - (s_*\lambda)(1-12(s_*\lambda)L)\left\|\nabla_{\widetilde{S}}\mathcal{L}(U^tS^tV^{t,\top})\right\|^2, \quad (38)$$

and assume that $\lambda s_* = \frac{1}{24L}$, i.e. $\lambda = \frac{1}{24Ls_*} \leq \frac{1}{Ls_*}$, which obeys the learning rate requirement of Theorem 2. Plugging this learning rate into (38) gives

$$\left\|\nabla_{\widetilde{S}}\mathcal{L}(U^tS^tV^{t,\top})\right\|^2 \leq 48L\left(\mathcal{L}({W_r}^t) - \mathcal{L}({W_r}^{t+1}) + L\vartheta\right).$$

Averaging from $t=1$ to $t=T$ yields

$$\begin{aligned}\min_{t=1,\ldots,T}\left\|\nabla_{\widetilde{S}}\mathcal{L}(U^tS^tV^{t,\top})\right\|^2 &\leq \frac{1}{T}\sum_{t=1}^T\left\|\nabla_{\widetilde{S}}\mathcal{L}(U^tS^tV^{t,\top})\right\|^2 \\ &\leq \frac{48L}{T}\left(\mathcal{L}({W_r}^{t=1}) - \mathcal{L}({W_r}^{t=T+1})\right) + 48L^2\vartheta,\end{aligned}$$

which concludes the proof. □

We remark that for a general loss function, it is possible that a point with small gradient magnitude can be far from the stationary points. However, assuming that the loss function is locally strongly convex in a neighborhood of a stationary point, then the gradient magnitude can be used to bound the distance to this stationary point in the neighborhood. For further reference, we point to [**?** , Eq. (4.12)] for the estimate.

H Analysis for FeDLRT with simplified variance correction

We consider the FeDLRT method with simplified variance correction, see Algorithm 5. Key difference to the standard FeDLRT with full variance correction, see Algorithm 1 is the modified coefficient update, incorporating global gradient information of the non-augmented coefficient matrix S for the variance correction term, that is

$$\check{V}_c = \check{G}_{\widetilde{S}} - \check{G}_{\widetilde{S},c} = \begin{bmatrix} \nabla_S \mathcal{L}(U^t S^t V^{t,\top}) - \nabla_S \mathcal{L}_c(U^t S^t V^{t,\top}) & 0 \\ 0 & 0 \end{bmatrix}. \tag{39}$$

Using the Riemmanian gradient, we can equivalently write

$$\check{V}_c = \left[U^\top \,|\, 0\right] (F(\widetilde{W}_r) - F_c(\widetilde{W}_r)) \begin{bmatrix} V \\ 0 \end{bmatrix} = \widetilde{U}^\top \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} (F_c(\widetilde{W}_r) - F(\widetilde{W}_r)) \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \widetilde{V}.$$

Remember the simplified variance corrected local coefficient update, given by

$$\begin{aligned} \widetilde{S}_c^{s+1} &= \widetilde{S}_c^s + \lambda \widetilde{U}^\top \left(F_c(\widetilde{W}_{r,c}^s) + \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} (F_C(\widetilde{W}_r) - F(\widetilde{W}_r)) \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} \right) \widetilde{V} \\ &= \widetilde{S}_c^s + \lambda \widetilde{U}^\top \left(F_c(\widetilde{W}_{r,c}^s) \right) \widetilde{V} + \check{V}_c. \end{aligned} \tag{40}$$

H.1 Global loss descent for FeDLRT with simplified variance correction

In the following we provide proof for a global loss descent for Algorithm 5, i.e. using the local coefficient update with variance correction (40).

Theorem 11. *(Restatement of Theorem 4) Under Assumption 1, if the local learning rate* $0 < \lambda \leq \frac{1}{12 L s_*}$*, then Algorithm 5 leads to the global loss descent*

$$\mathcal{L}({W_r}^{t+1}) - \mathcal{L}({W_r}^t) \leq -s_* \lambda (1 - \delta^2 - 12 s_* \lambda L + \delta^2 s_* \lambda) \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2 + L\vartheta, \tag{41}$$

with ${W_r}^t = U^t S^t V^{t,\top}$ *and* ${W_r}^{t+1} = U^{t+1} S^{t+1} V^{t+1,\top}$.

Proof. We split the adjusted coefficient update in (40) into the non-augmented $r \times r$ matrix S and the tree off-diagonal blocks given by the augmentation $\widehat{S}$:

$$\widehat{S} = \widetilde{S} - \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix}. \tag{42}$$

Analogously to the proof of Theorem 2, we consider

$$\begin{aligned} \mathcal{L}(\widetilde{W}_r^*) - \mathcal{L}(\widetilde{W}_r) &\leq \left\langle \widetilde{W}_r^* - \widetilde{W}_r, F(\widetilde{W}_r) \right\rangle + \frac{L}{2} \left\| \widetilde{W}_r^* - \widetilde{W}_r \right\|^2 \\ &= \left\langle \widetilde{U} \widetilde{S}^* \widetilde{V}^\top - \widetilde{U} \widetilde{S} \widetilde{V}^\top, F(\widetilde{W}_r) \right\rangle + \frac{L}{2} \left\| \widetilde{U} \widetilde{S}^* \widetilde{V}^\top - \widetilde{U} \widetilde{S} \widetilde{V}^\top \right\|^2 \\ &= \left\langle \widetilde{S}^* - \widetilde{S}, \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\rangle + \frac{L}{2} \left\| \widetilde{S}^* - \widetilde{S} \right\|^2 \\ &= \left\langle \widetilde{S}^* - \widetilde{S}, -\nabla_{\widetilde{S}} \mathcal{L}(\widetilde{W}_r) \right\rangle + \frac{L}{2} \left\| \widetilde{S}^* - \widetilde{S} \right\|^2, \end{aligned}$$

where the transformation uses orthonormality of $\widetilde{U}$ and $\widetilde{V}$ and definition of the projected gradient. We split the right hand side in terms corresponding to augmented terms $\widehat{S}$ and non-augmented terms S according to (42), i.e.,

$$\left\langle S^* - S, -\nabla_S \mathcal{L}(\widetilde{W}_r) \right\rangle + \frac{L}{2} \left\| S^* - S \right\|^2, \tag{43}$$

which is treated exactly as in the proof of Theorem 2, and the augmented terms

$$\left\langle \widehat{S}^* - \widehat{S}, -\nabla_{\widehat{S}} \mathcal{L}(\widetilde{W}_r) \right\rangle + \frac{L}{2} \left\| \widehat{S}^* - \widehat{S} \right\|^2. \tag{44}$$

First we bound the term (43). Remember that $\widehat{S} = 0$ at the start of the local iterations due to orthonormality of $\widetilde{U}, \widetilde{V}$. The coefficient update (40) for S reads

$$S_c^{s+1} = S_c^s + \lambda U^\top \left(F_c(\widetilde{W}_{r,c}^s) - F_c(\widetilde{W}_r) + F(\widetilde{W}_r) \right) V. \tag{45}$$

Then we can readily apply Theorem 2 to obtain the bound

$$\left\langle S^* - S, -\nabla_S \mathcal{L}(\widetilde{W}_r) \right\rangle + \frac{L}{2} \left\| S^* - S \right\|^2 \leq -(s_*\lambda)(1 - 12(s_*\lambda)L) \left\| U^\top F(\widetilde{W}_r) V \right\|^2. \tag{46}$$

Next, we bound (44), starting with the first term:

$$\begin{aligned}
\left\langle \widehat{S}^* - \widehat{S}, -\nabla_{\widehat{S}} \mathcal{L}(\widetilde{W}_r) \right\rangle &\overset{\text{(I)}}{=} \left\langle \widehat{S}^* - 0, -\nabla_{\widehat{S}} \mathcal{L}(\widetilde{W}_r) \right\rangle \\
&= \left\langle -\frac{\lambda}{C} \sum_{c=1}^{C} \sum_{s=0}^{s_*-1} \nabla_{\widehat{S}} \mathcal{L}_c(\widetilde{W}_{r,c}^s), -\nabla_{\widehat{S}} \mathcal{L}(\widetilde{W}_r) \right\rangle \\
&= \frac{\lambda}{C} \sum_{c=1}^{C} \sum_{s=0}^{s_*-1} \left\langle \nabla_{\widehat{S}} \mathcal{L}_c(\widetilde{W}_{r,c}^s), \nabla_{\widehat{S}} \mathcal{L}(\widetilde{W}_r) \right\rangle \\
&\leq \frac{\lambda}{C} \sum_{c=1}^{C} \sum_{s=0}^{s_*-1} \left\| \nabla_{\widehat{S}} \mathcal{L}_c(\widetilde{W}_{r,c}^s) \right\| \left\| \nabla_{\widehat{S}} \mathcal{L}(\widetilde{W}_r) \right\| \\
&\overset{\text{(II)}}{\leq} \frac{\lambda}{C} \sum_{c=1}^{C} \sum_{s=0}^{s_*-1} \delta^2 \left\| \nabla_{\widetilde{S}} \mathcal{L}(\widetilde{W}_r) \right\| \left\| \nabla_{\widetilde{S}} \mathcal{L}(\widetilde{W}_r) \right\| \\
&= \delta^2 s_* \lambda \left\| \nabla_{\widetilde{S}} \mathcal{L}(\widetilde{W}_r) \right\|^2 = \delta^2 s_* \lambda \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2,
\end{aligned}$$

where we use $\widehat{S} = 0$ in (I), and Assumption 1 in (II). Next, we bound the second term

$$\begin{aligned}
\frac{L}{2} \left\| \widehat{S}^* - \widehat{S} \right\|^2 =& \frac{L}{2} \left\| -\frac{\lambda}{C} \sum_{c=1}^{C} \sum_{s=0}^{s_*-1} \nabla_{\widehat{S}} \mathcal{L}(\widetilde{W}_{r,c}^S) \right\|^2 \\
\overset{\text{(I)}}{\leq}& \frac{L}{2} \lambda^2 \frac{1}{C} \sum_{c=1}^{C} \left\| \sum_{s=0}^{s_*-1} \nabla_{\widehat{S}} \mathcal{L}(\widetilde{W}_{r,c}^S) \right\|^2 \\
\overset{\text{(I)}}{\leq}& \frac{L}{2} s_* \lambda^2 \frac{1}{C} \sum_{c=1}^{C} \sum_{s=0}^{s_*-1} \left\| \nabla_{\widehat{S}} \mathcal{L}(\widetilde{W}_{r,c}^S) \right\|^2 \\
\leq& s_* \frac{L}{2} \delta^2 \lambda^2 \frac{1}{C} \sum_{c=1}^{C} \sum_{s=0}^{s_*-1} \left\| \nabla_{\widetilde{S}} \mathcal{L}(\widetilde{W}_r) \right\|^2 \\
\leq& \frac{L}{2} \delta^2 (s_*\lambda)^2 \left\| \nabla_{\widetilde{S}} \mathcal{L}(\widetilde{W}_r) \right\|^2 = \frac{L}{2} \delta^2 (s_*\lambda)^2 \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2,
\end{aligned}$$

where we used Jensen's inequality in (I) again Assumption 1. We combine the bound on the non-augmented terms (46) and the two bounds above for the augmented terms to

$$\begin{aligned}
&\mathcal{L}(\widetilde{W}_r^*) - \mathcal{L}(\widetilde{W}_r) \leq \left\langle \widetilde{W}_r^* - \widetilde{W}_r, F(\widetilde{W}_r) \right\rangle + \frac{L}{2} \left\| \widetilde{W}_r^* - \widetilde{W}_r \right\|^2 \\
&\leq -(s_*\lambda)(1 - 12(s_*\lambda)L) \left\| U^\top F(\widetilde{W}_r) V \right\|^2 + \delta s_* \lambda \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2 + \delta (s_*\lambda)^2 \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2 \\
&\overset{\text{(I)}}{\leq} -(s_*\lambda)(1 - 12(s_*\lambda)L) \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2 + \delta s_* \lambda \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2 + \delta (s_*\lambda)^2 \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2 \\
&= -(s_*\lambda)(1 - \delta^2 - 12(s_*\lambda)L + \delta^2(s_*\lambda)) \left\| \widetilde{U}^\top F(\widetilde{W}_r) \widetilde{V} \right\|^2,
\end{aligned}$$

where we use in (I) $\left\| U^\top F(\widetilde{W}_r)V \right\| \leq \left\| \widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V} \right\|$. Using Equation (36), we can conclude the proof:

$$\begin{aligned}
&\mathcal{L}(U^{t+1}S^{t+1}V^{t+1,\top}) - \mathcal{L}(U^tS^tV^{t,\top}) \\
\leq &-(s_*\lambda)(1-\delta^2-12(s_*\lambda)L+\delta^2(s_*\lambda)) \left\| \widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V} \right\|^2 + L\vartheta.
\end{aligned}$$

□

H.2 Global convergence of FeDLRT with simplified variance correction

Corollary 2. *(Restatement of Corollary 1) Under Assumption 1, Algorithm 5 guarantees for the learning rate* $\lambda \leq \frac{1}{s_*(12L+\delta^2)}$

$$\min_{t=1,\dots,T} \left\| \nabla_{\widetilde{S}} \mathcal{L}({W_r}^t) \right\|^2 \leq \frac{96L}{T}\left(\mathcal{L}({W_r}^1) - \mathcal{L}({W_r}^{T+1})\right) + 96L^2\vartheta, \tag{47}$$

with ${W_r}^t = U^tS^tV^{t,\top}$, ${W_r}^1 = U^1S^1V^{1,\top}$. *and* ${W_r}^{T+1} = U^{T+1}S^{T+1}V^{T+1,\top}$.

Proof. Consider Theorem 4,

$$\mathcal{L}({W_r}^{t+1}) - \mathcal{L}({W_r}^t) \leq -(s_*\lambda)(1-\delta^2-12(s_*\lambda)L+\delta^2(s_*\lambda)) \left\| \widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V} \right\|^2 + L\vartheta$$

and assume that $\lambda s_* = \frac{1}{(12L+\delta^2)}$, i.e. $\lambda = \frac{1}{s_*(12L+\delta^2)} \leq \frac{1}{Ls_*}$, which obeys the learning rate requirement of Theorem 2. Plugging this learning rate into (38) gives

$$\left\| \nabla_{\widetilde{S}} \mathcal{L}({W_r}^t) \right\|^2 \leq 96L\left(\mathcal{L}({W_r}^t) - \mathcal{L}({W_r}^{t+1}) + L\vartheta\right),$$

where we use $(\frac{1}{4}-\delta^2) \leq \frac{1}{4}$ and $\frac{1}{(12L+\delta^2)} \leq \frac{1}{12L}$ Averaging from $t=1$ to $t=T$ yields

$$\begin{aligned}
\min_{t=1,\dots,T} \left\| \nabla_{\widetilde{S}} \mathcal{L}({W_r}^t) \right\|^2 \leq &\frac{1}{T}\sum_{t=1}^{T} \left\| \widetilde{U}^\top F(\widetilde{W}_r)\widetilde{V} \right\|^2 \\
\leq &\frac{96L}{T}\left(\mathcal{L}({W_r}^{t=1}) - \mathcal{L}({W_r}^{t=T+1})\right) + 96L^2\vartheta,
\end{aligned}$$

which concludes the proof. □

I NeurIPS review

I.1 Paper Decision

While the reviewers agreed that the work has interesting contributions and found merits in them, they raised several issues that are worth addressing in a careful and thorough manner. These include the exposition of the manuscript, scope of the numerical experiments and presentation of the numerical results, and possible extensions of the proposed approach to other settings. As the revision will be extensive and thus requires another round of review, and in view of the fact that NeurIPS can only accommodate one round of review, I regrettably have to reject the manuscript at this point.

I.2 Official Comment by the Authors

Wrap up statement of the discussion period

As the discussion period approaches its final day, we would like to thank the reviewers again for their feedback and hope we have clarified their remarks.

Overall, the reviewers pointed out that the submission proposes a sound algorithm solving the increasingly relevant and important problem of automatic compression for distributed and edge computing, while providing a robust theoretical foundation with global convergence proofs. The algorithm is evaluated on multiple datasets and network architectures (convolutional layers, transformers, and fully connected layers) and test problems. The reviewers found the paper to be well written.

We received valuable, constructive feedback by the reviewers and summarize the rebuttal and discussions in the following:

- We are happy to have resolved some unclear statements in the test-case descriptions, and contribution statements.
- Prompted by reviewer VUf8, we have extended the algorithm description for Tensor valued layers, prominently featured in convolutional neural networks, where we apply the proposed method to Tucker-Factorized tensors, and demonstrate their viability in the general answer PDF.
- We have provided more convergence plots to illustrate the effect of variance correction in neural network training, in addition to the plots for least squares regression in the paper.
- We have explained the mechanics of the basis augmentation, prompted by reviewer rsLZ, and how the basis augmentation, which does not induce any approximation errors, provides a key ingredient for the analysis. We further clarified that indeed, the coefficient update step resembles an optimization step on the low-rank manifold.
- We have clarified the consequences of the convergence guarantee on the distance of the trained solution to the stationary point in the fruitful discussion with reviewer hYpR.
- In the fruitful discussion with reviewer VUf8, we have clarified some limitations of the proposed method in context of heterogeneous data, and partial participation. In summary, we have seen that the proposed method works well in the setting
  - homogeneous data, deterministic gradient (Main paper, Section 4.1 Homogeneous test; supplemented by Figure 4)
  - homogeneous data, stochastic (mini-batch) gradient (Main paper, Section 4.2 and appendix; supplemented by Figure 5-8)
  - heterogeneous data, deterministic gradient (Main paper, Section 4.1 Heterogeneous test; supplemented by Figure 1)

  Preliminary tests during the discussion period showed that more research is required to provide good results for heterogeneous data, stochastic (mini-batch) gradient.
- Finally, we point out how the method can be extended to a partial participation scenario, where not all clients are active at the same time.

We hope that our answers have satisfied the reviewers, and we thank them again for their feedback.

Kind regards,

Authors

I.3 Review 1 - hYpR

Summary: This paper proposes a low-rank scheme to reduce communication and computation cost in FL, while also reducing client drift.

Soundness: 3: good Presentation: 3: good Contribution: 3: good Strengths: The paper is well-written and seems to be solving a relevant and important problem.

Weaknesses: see below

Questions:

Section 2: In Fig 1, how is the initial trajectory of FedAvg and FedLin identical till FedAvg settles?

Section 3: lines 126-7: just my curiosity, but why are SVD and QR decomposition not GPU friendly? In Fig 3 caption, the sentence about cost drop after is unclear. The description following Theorem 3 connects the non-zero bias in (12-13) with Fig 1. However, Fig 1 shows distance to solution, while in theorem 2, 3, these are gradient norms. Can we really say anything much about the convergence based on bias in gradient norm, since in the worst case, we can be arbitrarily far from any stationary point?

Section 4: heterogeneous test case - why do all clients have access to all the training points? Shouldn't the data be distributed across clients as in the homogeneous case?

Limitations: n/a

Flag For Ethics Review: No ethics review needed.

Rating: 6: Weak Accept: Technically solid, moderate-to-high impact paper, with no major concerns with respect to evaluation, resources, reproducibility, ethical considerations.

Confidence: 3: You are fairly confident in your assessment. It is possible that you did not understand some parts of the submission or that you are unfamiliar with some pieces of related work. Math/other details were not carefully checked.

Code Of Conduct: Yes

I.3.1 Rebuttal by authors

Rebuttal: We thank the reviewer for their review. Each of the questions is addressed below.

Regarding the trajectories in Fig.1: We would like to clarify that, in the early stage, the trajectories of FedAvg and FedLin reported in Fig. 1 are very close but not identical. The similarity of these two trajectories is due to the fact that the variance correction term, which is the distinguishing factor between FedAvg and FedLin, see Eq. (4), being insignificant in the early stage of training for this problem. The variance correction term corrects the local gradient directions of the clients to prevent stalling of the convergence process. In this problem, the distances between model parameters at early stage and the the local optima are much longer than the distance between local (client) optima and the global optimum of the federated problem. In this case, the local gradients are good estimates of the global federated gradients and thus the variance correction effect is insignificant, which leads to similar behavior of FedAvg and FedLin. As the model parameters approach the local/global optimum, the local gradients are no longer good estimates of the global gradient due to data heterogeneity, and the variance correction term eventually results in better convergence behavior of FedLin. The same argument holds for the proposed low-rank methods with and without variance correction.

Regarding GPU friendliness: We consider SVD and QR not as GPU friendly as other parts of the proposed algorithm. They are less GPU friendly because the underlying SVD and QR algorithms are inherently sequential. For example, in a QR decomposition, the orthogonal space is build by rotating each column vector of a matrix onto the orthogonal complement of the subspace spanned by existing vectors. This sequential iterative procedure makes massively parallel implementation nontrivial, as opposed to, e.g., batchwise network evaluations.

Regarding the caption of Fig. 3: As for the sentence in question in the caption of Fig. 3, we meant to state that, when the rank is below 200, the communication, computation, and memory costs of the FeDLRT are lower than the costs of the full rank FedLin method. Thank you for pointing our the potential confusion. We will clarify this in a revised version.

Regarding the description following Theorem 3: Thank you for this remark. The result in Theorem 3 describes convergence to a stationary point by providing upper bounds on the norm of the loss function gradient. For a general loss function, it is indeed possible that a point with small gradient magnitude can be far from the stationary points. However, if we assume that the loss function is locally strongly convex in a neighborhood of a stationary point, then the gradient magnitude can be used to bound the distance to this stationary point in the neighborhood. Please see, for example, Eq. (4.12) in Bottou, Léon, Frank E. Curtis, and Jorge Nocedal, "Optimization methods for large-scale machine learning." SIAM review 60, no. 2 (2018): 223-311, for the estimate and Appendix B therein for the proof.

Regarding the heterogeneous data test case: Thank you for pointing out the ambiguity in the problem description. In the heterogeneous linear regression test case, each client performs regression to a different target function. Therefore, even though they share the same 10,000 locations sampled on , the local objective functions are defined with different target functions . We will clarify the problem configuration in a revised version.

I.3.2 Comment by reviewer

Thanks to the authors for their response. I maintain my score. All the best!

I.4 Review 2 - VUf8

Summary: The paper introduces FeDLRT, a federated algorithm to train and truncate low-rank weights automatically. The algorithm is based on a distributed version of the dynamic low-rank training. This requires multiple communication rounds (3 at worst) between the server and all clients, where first the U and V basis are augmented on the server after the basis gradients are sent from the clients and aggregated by the server. Then the clients learn the coefficients S and eventually correct the variance. The server then aggregates S, compress, and update the basis. The algorithm has theoretical guarantees of global convergence.

Soundness: 3: good Presentation: 2: fair Contribution: 3: good Strengths: The algorithm is sound and has theoretical guarantees of global convergence. Automatic compression is an increasingly important research topic, especially for edge and distributed training.

Weaknesses: I personally found the paper hard to follow and to distinguish between the actual contributions and what is instead based on the existing literature. The appendix is helpful, but I suggest the authors restructure section 3 and divide it into the background for dynamical low-rank training and their actual contributions. The algorithm seems a federated porting of the DLRT algorithm, which in order to have guarantees requires at least double communication per round to have shared augmented bases. Also, a clear contributions section would be helpful.

The way the CIFAR10 dataset has been split across clients is quite naive (only a few clients) and homogeneous - this is not a standard practice in FL where the algorithms are generally tested in heterogeneous non-iid settings, for instance splitting data among clients, based on a Dirichlet distribution (see for instance https://arxiv.org/abs/1909.06335 and https://arxiv.org/abs/2003.00295).

The algorithm seems to be working only in full participation mode (at each round it needs to communicate with all the clients), so it is mainly made for cross-silo settings with a few always available clients rather than cross-device. Indeed it requires 2 (or even 3 in the worst case) communication rounds (broadcast and aggregate operations)

Questions: Experiments on computer vision datasets: in the main paper the authors present experiments by training only the classifier using their proposed method. It is unclear if the method can be extended to all layers to train them and automatically compress them to their optimal rank. It is unclear if the method can be extended to convolutional layers.

It would be interesting to see plots of the loss and accuracy on the CIFAR dataset (with heterogeneity) to check the actual speed of convergence of the method against baselines. Something similar to Figure 4, but at least for the CIFAR10 dataset and against baselines such as FedAVG, FedLin, and potentially also something more recent to tackle heterogeneity. Indeed, while the method proposed has a variance reduction correction, apparently for mitigating client-drift, it is unclear if it can handle and mitigate the effect of heterogeneity.

Could the algorithm be extended to avoid communicating twice, hence to work in, cross-device, realistic, and partial participation settings?

Limitations: The authors should dedicate more space to the limitations of their approach as they are not clearly expressed and, while sound, the work seems not ready to be a practical algorithm yet.

Flag For Ethics Review: No ethics review needed.

Rating: 6: Weak Accept: Technically solid, moderate-to-high impact paper, with no major concerns with respect to evaluation, resources, reproducibility, ethical considerations.

Confidence: 3: You are fairly confident in your assessment. It is possible that you did not understand some parts of the submission or that you are unfamiliar with some pieces of related work. Math/other details were not carefully checked.

Code Of Conduct: Yes

I.4.1 Rebuttal by authors

Rebuttal: We thank the reviewer for their review. To improve the presentation, we propose to restructure Section 2 and 3 by moving the description of the (non-federated) dynamical low-rank training from Section 3 to Section 2 as part of the background. The new Section 2 will include subsections on background for federated learning and variance correction, background for low-rank and dynamical low-rank training, as well as a standalone subsection on the contribution, which will be derived from the last paragraph of Section 2 in the current version. After this, the entire Section 3 is dedicated to the proposed method and analysis.

We address each of the questions below.

Regarding compressing convolutions: We focused on the classifier since these layers are matrix-valued, and thus the proposed algorithm is directly applicable. We have extended the implementation of FeDLRT to train convolutional layers in a low-rank fashion as well. The results of FeDLRT applied to all layers (convolutions and classifiers) of VGG16 on CIFAR10 are reported in Fig. 2 in the general response PDF file. These results resemble the the ones in Fig. 7 with slightly different compression ratios, since more layers are now low rank. In the following paragraph, we give technical details in the extension of FeDLRT to compress convolutional layers.

To extend FeDLRT to convolutional layers, we follow the approach considered in, e.g., (https://arxiv.org/abs/2305.19059) for (non-federated) DLRT, where a 2D convolution is interpreted as an order-4 tensor and factorized by using the Tucker decomposition. To this end, the Tucker bases $U_i \in \mathbb{R}^{n_i \times r_i}$ for $i = 1, \ldots, 4$. replace the U and V bases in the matrix case, and the Tucker core tensor $C \in \mathbb{R} \in \mathbb{R}^{r_1, \times \cdots \times r_4}$ replaces the coefficient matrix S , to which the variance correction is applied. The analysis holds for the Tucker Tensor case, since Tucker Tensors have a manifold structure. In the proofs, we need to project onto all bases U_i . The compression step is performed with an truncated Tucker decomposition of the core tensor , instead of an SVD of the coefficient matrix . For intuition, one can also refer to the matrix case as the order-2 Tucker Tensor case. Remark that the bases are all updated simultaneously, thus the adaption to the tensor case does not require more communication rounds.

Regarding accuracy plots: We thank the reviewer for the constructive question. First, we provide in the general response PDF file, Fig. 1, a convergence plot for Resnet18 on CIFAR10 for the (homogeneous) test case reported in Fig. 5 of the main manuscript. One can see that the benefit of the variance correction term (in FeDLRT w/ var/cor and FedLin) mitigates the stalling of the convergence seen in the non-variance-corrected methods (FeDLRT w/o var/cor and FedAvg).

Regarding heterogeneous test cases: Prompted by this question, we conducted a preliminary study for a federated scenario with heterogeneous data on the client devices, drawn from a Dirichlet distribution. We found that the variance correction does not provide significant performance increase in scenarios with stochastic multi-batch gradient descent on clients and strong heterogeneity. Given the positive results for homogeneous data, we consider this challenge a relevant future research direction and will investigate the potential incorporation of more recent techniques into FeDLRT to tackle strongly heterogeneous data.

Regarding a modification to reduce the communication rounds: The FeDLRT algorithm and the convergence analysis require communication of the basis and optionally the variance correction term prior to the client coefficient updates, therefore, both communication rounds are necessary. The variance corrected baseline, FedLin, considered in this work also requires two communication rounds. Moreover, we would like to emphasize that the total communication cost (including all communication rounds) per aggregation round of FeDLRT in practice is nearly an order of magnitude smaller than the full-rank baselines, e.g. FedLin or FedAvg, because FeDLRT only communicates part of the factors each round. See Fig. 3 of the manuscript and results in e.g. Fig. 5. We also remark that the variance correction benefits the convergence behavior, see, e.g. Fig. 1, and the two right panels of Fig. 4 in the main manuscript, as well as Fig. 1 in the general response PDF file. The superior convergence behavior implies that the proposed method reaches the target accuracy in fewer aggregation rounds, thus requiring fewer overall communication rounds.

A potential limitation of having two communication rounds, instead of one, is that latency differences of clients are more pronounced during hand-shakes. However, even for basic methods with single communication round, e.g. FedAvg, latency differences still pose a problem. To fully address this issue, one may need to extend the method non-trivially to accommodate asynchronous communication scenarios, which we find relevant as a future research direction.

On the other hand, allowing for partial participation is certainly possible in FeDLRT, as long as the active clients are consistent in all communication rounds within the same aggregation round. However, we have not been able to experiment in the partial participation configuration with many (> 100) clients training relevant network architectures such as Resnet18 or VGG16, due to the constraint on the computation resources and the current implementation of FeDLRT. We agree that this is an important research direction and will make attempts to scale up the FeDLRT method.

I.4.2 Comment by reviewer

Thank you for answering my questions and concerns.

Could you clarify (please be specific) how the algorithm could work in the partial participation case and if errors could arise (if the algorithm's guarantees are broken), especially in the heterogeneous case?

I.4.3 Comment by authors

We start our answer with a description of FeDLRT without variance correction for the partial participation case and then discuss a potential direction for extending FeDLRT to handle data heterogeneity in the partial participation scenario.

When variance correction is turned off, FeDLRT can be applied to partial active clients by considering only a (potentially random) subset $C^t \subset 1, \ldots, C$ of clients within a global aggregation round in Algorithm 1 in the original manuscript. Specifically, at aggregation round t, only the clients C^t are taken into account in the broadcasting in lines 2 and 6, client operations in lines 3, 7, 8, 15, and the aggregations in lines 4 and 16.

Due to the partial participation, the gradients computed in line 4 are no longer the global loss gradients with respect to U and V, respectively. However, this does not break the mechanism and analysis of FeDLRT since the augmented basis is not required to come from augmenting the global gradients.

The set of active clients can vary for different aggregation rounds, but, for FeDLRT, C^t needs to remain constant within an aggregation round. This restriction is consistent to the scenario considered in most existing work on federated learning with partial participation.

As for the performance, we expect that the FeDLRT w/o variance correction described above to perform similarly as FedAvg in terms of final accuracy, but at a much lower communication and memory cost due to the low rank technique.

Based on the preliminary results on heterogeneous data discussed in the rebuttal, we do not expect the current variance correction scheme to provide significant advantages in the partial participation case with heterogeneous data. A potential approach to address this issue is to incorporate in the FeDLRT algorithm an advanced variance correction scheme, such as the FedVARP scheme proposed in https://openreview.net/forum?id=HlWLLdUocx5 , which is tailored to the partial participation case with heterogeneous data.

We are happy to provide more details if there are further questions.

I.4.4 Comment by reviewer

Thank you for your responses. In its current state, I still believe the paper is borderline, as it does not seem intended for general heterogeneous federated learning. That said, this could be a bias on my part, as this is one of my main areas of expertise. The rest of the paper and the authors' responses are convincing, but I believe the authors should incorporate their explanations and additional details about my concerns into the main paper. I have the impression that the algorithm is more suited for distributed learning, where heterogeneity and partial participation are less of an issue, though compression could still be beneficial due to communication constraints.

I am raising my score, and I hope the authors will consider my concerns and suggestions in the final version of the paper as well as in future work.

I.5 Review 3 - rsLZ

Summary: This paper introduces FeDLRT (Federated Dynamical Low-Rank Training), an innovative method designed to enhance federated learning by incorporating a low-rank client optimization step and an optional variance correction mechanism. FeDLRT builds upon the dynamic low-rank approximation (DLRA) method, extending it to neural network training in a federated learning context. The key contributions of this work include the development of a basis update and Galerkin (BUG) splitting scheme that allows for the efficient and dynamic adjustment of the rank, ensuring client-wide manifold consistency, and minimizing communication costs.

Soundness: 2: fair Presentation: 3: good Contribution: 3: good Strengths: The paper presents a robust theoretical framework by building upon the dynamic low-rank approximation (DLRA) method and extending it to the federated learning context.

The dynamic adjustment of the rank through the BUG splitting scheme is an innovation. This approach not only ensures client-wide manifold consistency but also enables efficient basis augmentation and coefficient updates, leading to better utilization of communication resources. The optional variance correction mechanism adds another layer of robustness, addressing potential discrepancies in local updates and ensuring convergence.

The extensive evaluation on real datasets demonstrates the effectiveness of the proposed approach for federated dynamical low-rank training.

Weaknesses: The method requires two communication rounds—one for aggregating global basis gradients and another for locally updated coefficients, which might still be considered high in some federated learning scenarios. When the number of clients is large, each gradient and basis update can result in additional communication overhead. Can everything be done in one communication round?

The experiments are limited to ResNet18 on CIFAR-10. This method involves gradient calculation and local optimization on the client side, as well as incremental basis update and QR decomposition on the server side. Whether the model is valid when applied to higher-dimensional data or larger models such as RoBERTa or LLaMA, and large datasets like SST-2.

When updating the basis U and V, the effect of the upper triangular matrix R is ignored in the new incremental basis obtained by using QR decomposition. Will this affect the performance of the model? What is the error range caused by updating the coefficient matrix S with the new incremental basis?

When updating the incremental coefficient matrix S in this paper, using an update method similar to SGD will lead to the original parameter not being on the manifold after updating.

It is better to conduct the experiments with baselines. Otherwise it is difficult to justify the effectiveness of the proposed method.

Questions: Weaknesses

Limitations: No.

Flag For Ethics Review: No ethics review needed.

Rating: 3: Reject: For instance, a paper with technical flaws, weak evaluation, inadequate reproducibility and/or incompletely addressed ethical considerations.

Confidence: 4: You are confident in your assessment, but not absolutely certain. It is unlikely, but not impossible, that you did not understand some parts of the submission or that you are unfamiliar with some pieces of related work.

Code Of Conduct: Yes

I.5.1 Rebuttal by Authors

We thank the reviewer for this review. Please find the answer to the questions below.

1. Regarding communication cost: The method requires two communication rounds, since the basis update and variance correction term need to be available to each (active) client before the client update starts. This being said, the proposed method communicates only parts of the weight matrix factors during each communication round, i.e. its total communication cost is significantly reduced compared to baseline methods, such as FedAvg and FedLin. Further, FedLin, the baseline with variance correction, also requires two communication rounds. We remark that the variance correction also benefits the convergence behavior, see, e.g. Fig. 1, and the two right panels of Fig. 4 in the main manuscript, as well as Fig. 1 in the general response PDF file. The superior convergence behavior implies that the proposed method reaches the target accuracy in fewer aggregation rounds, thus requiring fewer overall communication rounds. In conclusion, we argue that the total number of communicated floating point numbers is significantly reduced in FeDRLT, compared to the mentioned baselines.
2. Regarding experiments: In addition to ResNet18 on CIFAR10, we provide numerical results for two convex test problems in the main manuscript, as well as AlexNet on CIFAR10, VGG16 on CIFAR10, and a Vision Transformer on CIFAR100 in the appendix, thus discussing performance on convolutional networks and transformers, two of the most widely used network architectures.

   The QR decomposition required in the basis augmentation step acts on a tall, but skinny $n \times 2r$ matrix, thus requiring $(2r)^2$ computational cost (typically $n \gg r$), which still smaller than the n^2 cost of multiplying a full-rank weight matrix with an input vector required in the full-rank baseline methods, such as FedLin and FedAvg. Further, the QR decomposition is performed once per aggregation round on the server, which has typically more compute resources than the clients. We stress that the method aims to minimize total communication and client compute costs, combined with preferred convergence behavior. Considering Table 1, we remark that FeDLRT is (to the best of our knowledge) the only one with linear dependence of the client compute cost on the layer dimensions.
3. Regarding the basis update: The basis update extends the old basis by the span of the gradient dynamics. Thus, the spans of the augmented bases obtained in the basis augmentation step also contain the spans of original bases. Consequently, **no error is introduced by augmenting the basis, and the training loss does not increase**. Intuitively the basis augmentation can be seen as a conservative extension of the search space of the neural network training: we allow to search for new coefficients in a manifold of twice the rank.

   In further detail, we refer to line 5 of Algorithm 1 (using Eq. (6)), where the basis update of U and V is performed. Due to the QR decomposition, we have span$(\widetilde{U}) =$ span$([U^t, G_U])$. **The R matrix is not relevant to the construction of the new basis and thus can be discarded in the algorithm**. However, since U^t is already orthonormal by construction, we further have $\widetilde{U} = [U^t, \bar{U}]$ with $U^t \perp \bar{U}$, which implies that the upper half of R is a unit matrix. This is indeed important since it yields the explicit expression of $\widetilde{S}$ in Lemma 1. As a consequence, the augmented low-rank representation $\widetilde{U}\widetilde{S}\widetilde{V}^\top$ is consistent with the non-augmented representation $USV^\top$, i.e. $||\widetilde{U}\widetilde{S}\widetilde{V}^\top - USV^\top||_F = 0$, which is a requirement in the proof of Theorems 2 and 4.
4. Regarding coefficient updates: The method is carefully constructed to so that the coefficient matrix update is an update within the manifold of rank $2r$ matrices, because the bases $\widetilde{U}$ and $\widetilde{V}$ remain constant in the client update steps. This not only implies that the updates stay on the manifold, but that the proposed method is robust with respect to the curvature of the low-rank manifold. We refer to Appendix D and specifically Theorem 5 in the manuscript for a technical discussion of the robust optimization method

that forms the foundation of this federated scheme. For further reading on why the BUG scheme is a robust optimization method on manifolds, we would like to refer to [https://arxiv.org/pdf/2205.13571, Section 4]. For a well-written geometric interpretation of the method, we refer to (https://arxiv.org/abs/1705.08521).

5. Baselines of experiments: We compare FeDLRT to the full-rank baselines, FedAvg and FedLin, in all numerical experiments. We show that across all test cases, the FeDLRT method confidently mirrors the convergence behavior of its full-rank counterpart, just as estimated in Theorem 5. Meanwhile, FeDLRT dynamically compresses the model to reduce communication bandwidth and the computational cost.