TRAINING A FEW ANCHORS FOR A DENSE UPDATE

Anonymous authors
Paper under double-blind review

ABSTRACT

Parameter-efficient fine-tuning reduces the number of optimized variables, but most methods impose a parameterization without considering the geometry of the downstream objective. We GATA: Geometry Aware Task Alignment, which selects a task-conditioned adaptation space and represents it using physical parameter anchors. Candidate Hessian modes are ranked by their predicted improvement under a damped quadratic model, $(\boldsymbol{v}_i^\top \boldsymbol{g})^2/(\lambda_i + \gamma)$. Q-DEIM then selects k parameter coordinates, and a fixed affine map reconstructs their updates over all N adapted parameters. AdamW therefore optimizes only k anchor variables while producing a dense update.

Across five-seed classification-head experiments, task alignment consistently outperforms curvature-only and dimension-matched random spaces. On ViT-B/16/CIFAR-10, it improves accuracy at $k = 20$ from $30 \pm 6.6\%$ for leading-curvature modes to $83 \pm 1.8\%$, while its Q-DEIM representation retains $84 \pm 1.8\%$. At $k = 150$, task-aligned Q-DEIM reaches approximately 92% accuracy using 150 optimized variables, comparable to rank-4 LoRA using 3,112. On ResNet-18/CIFAR-10, the task-aligned space reaches $78.65 \pm 0.85\%$ at $k = 250$, compared with $60.66 \pm 1.45\%$ for a matched random space. Complete Hessian spectra further show that curvature rank grows with model size before saturating at a task-dependent level, while GPT MLP measurements exhibit sublinear growth of spectral-energy rank. These results support curvature dimension as an interpretable adaptation budget and physical anchors as a compact interface for optimization and communication.

1 INTRODUCTION

Modern neural networks are adapted to many downstream tasks. Dense fine-tuning updates every parameter and requires correspondingly large optimizer states, memory, storage, and communication. These costs grow with model size, even when the downstream task requires only a small change in behavior.

Parameter-efficient fine-tuning (PEFT) reduces adaptation cost by constraining the update. LoRA and QLoRA, for example, represent changes in the weight-matrix through low-rank factors (Hu et al., 2022; Dettmers et al., 2023). This reduces the number of trainable variables, but the chosen matrix rank does not directly reveal the number or orientation of update directions needed for a particular downstream task. We suggest that the local geometry of the downstream loss can be used to identify candidate directions. This provides a principled basis for choosing both the update space and its dimension.

The local loss landscape provides a task-dependent view of adaptation. Around a reference model $\boldsymbol{\theta}_0$,

$$\mathcal{L}(\boldsymbol{\theta}_0 + \Delta\boldsymbol{\theta}) \approx \mathcal{L}(\boldsymbol{\theta}_0) + \boldsymbol{g}^\top \Delta\boldsymbol{\theta} + \frac{1}{2}\Delta\boldsymbol{\theta}^\top H \Delta\boldsymbol{\theta},$$

where $\boldsymbol{g}$ is the task gradient and H is the Hessian. The Hessian describes the local curvature of the downstream loss. Empirical studies suggest that, in large neural models, substantial curvature can be concentrated in a subspace much smaller than the parameter space (Li et al., 2018; Aghajanyan et al., 2021). This observation motivates our central question: can directions identified from that compact geometry support effective fine-tuning, and how should a downstream task select among them? If so, the number of coordinates needed for adaptation may grow much more slowly than the model parameter count. The method and experiments that follow examine this possibility.

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

Curvature alone does not determine task utility. A highly curved direction may have little alignment with the desired update. We therefore combine Hessian curvature with the downstream gradient to select directions that are both geometrically significant and useful for reducing the task loss.

The selected directions are dense combinations of model parameters. To obtain a practical representation, we use the QR-based Discrete Empirical Interpolation Method (Q-DEIM) (Chaturantabut & Sorensen, 2010; Drmač & Gugercin, 2016) to identify k physical parameter coordinates. These *structural anchors* determine an update within the selected space. AdamW maintains and updates only these anchors, while a fixed affine map reconstructs the corresponding dense update over the other $N - k$ adapted parameters.

We call this approach GATA: Geometry Aware Task Alignment. It provides an alternative to dense fine-tuning and low-rank adapters. The Hessian defines the local geometry, the task gradient selects useful directions, and Q-DEIM exposes those directions through physical coordinates. The resulting dimension k is an interpretable measure of the adaptation required by the task.

Our contributions are:

- We introduce GATA and a Q-DEIM parameterization in which k ($\ll N$) physical anchors control a dense N-parameter update.
- We show strong accuracy at small budgets. On ViT-B/16/CIFAR-10, 20 anchors achieve approximately 84% accuracy, while 150 anchors reach approximately 92%, comparable to rank-4 LoRA with 3,112 optimized scalars. GATA also improves ResNet-18 accuracy from approximately 61% for a matched random space to 79% at $k = 250$.
- We characterize the scaling of the curvature dimension. Exact Hessians show linear growth followed by task-dependent saturation, while GPT MLP matrices show sublinear growth in spectral-energy rank.

2 Related Work

Parameter-efficient fine-tuning. Parameter-efficient fine-tuning reduces adaptation cost by imposing structure on the trainable update. LoRA represents matrix updates through low-rank factors, while QLoRA combines low-rank adapters with quantized frozen weights (Hu et al., 2022; Dettmers et al., 2023). AdaLoRA allocates rank adaptively across layers, and DoRA separates changes in weight magnitude from changes in direction (Zhang et al., 2023; Liu et al., 2024). These approaches specify the algebraic form of an update independently of the local task geometry. Our method instead derives an update space from the downstream Hessian and gradient, then represents that space through selected physical coordinates. Its dimension k therefore denotes the number of selected subspace coordinates, which is geometry-inspired and is distinct from the matrix rank r used by LoRA-based methods.

Low-dimensional optimization geometry. Neural-network training can succeed in parameter spaces much smaller than the ambient model dimension (Li et al., 2018; Aghajanyan et al., 2021). Optimization trajectories have also been related to low-dimensional, Hessian-associated spaces (Gur-Ari et al., 2018). These findings support the existence of compact adaptation spaces, but the magnitude of the curvature alone does not determine which directions are useful for a particular downstream objective. We explicitly combine curvature with the task gradient, selecting modes according to their predicted local loss reduction. Dimension-matched random spaces and leading-curvature spaces distinguish the contribution of task alignment from the effect of dimensionality alone.

Empirical interpolation and physical coordinates. DEIM reconstructs vectors in a known low-dimensional space from a subset of their entries, and Q-DEIM uses pivoted QR to select a stable set of interpolation coordinates (Chaturantabut & Sorensen, 2010; Drmač & Gugercin, 2016). We apply Q-DEIM to a task-aligned parameter-update space, allowing k physical weights to control an update over N parameters. This differs from the sparse-subnetwork and lottery-ticket methods, which identify a subset of parameters that can be trained as a sparse model (Frankle & Carbin, 2019). Our anchors are sparse coordinates, but the reconstructed update is dense.

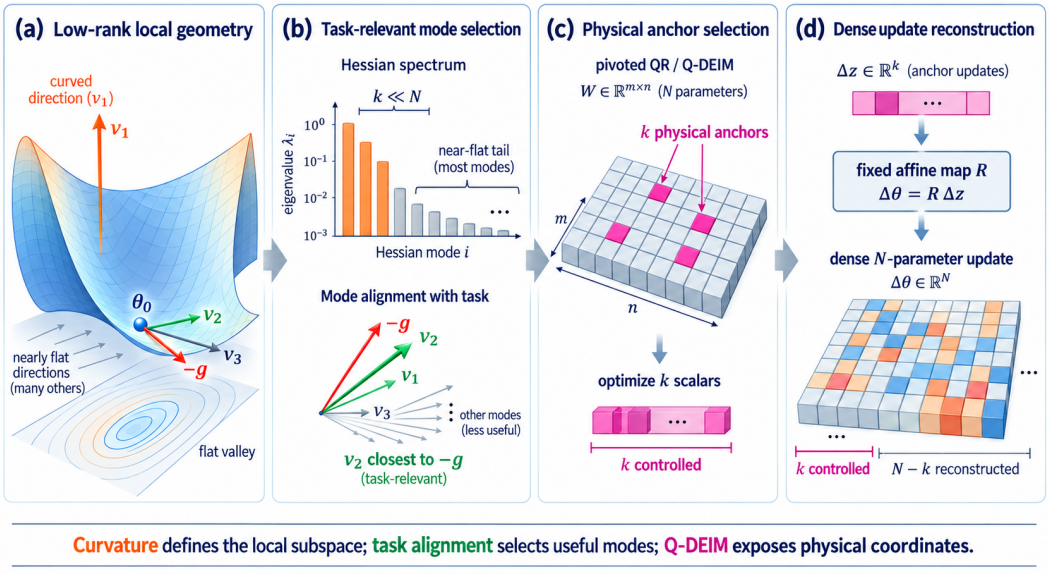


Figure 1: **Task-aligned Hessian adaptation in physical coordinates.** **(a)** Curvature is concentrated in a small number of local Hessian modes. **(b)** Task alignment prioritizes modes according to their predicted utility for the downstream objective. **(c)** Pivoted QR selects k physical anchors that provide coordinates for the chosen space. **(d)** Optimizing the anchors produces a dense N-parameter update through the fixed reconstruction map.

3 METHOD

Figure 1 summarizes the complete construction of GATA: curvature identifies a compact local space, task alignment selects useful modes, and Q-DEIM exposes that space through physical anchor parameters.

3.1 SETUP AND APPROACH

Let $f_{\boldsymbol{\theta}}$ be a pretrained model with N selected parameters $\boldsymbol{\theta} \in \mathbb{R}^N$, and let $\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta})$ be the loss for downstream task $\mathcal{T}$. Parameters outside the selected set remain fixed. Starting from a checkpoint $\boldsymbol{\theta}_0$, we construct an update controlled by $k \ll N$ trainable scalars.

We estimate a local Hessian eigenspace, select k physical parameter coordinates that represent that space, and reconstruct the remaining entries of each update. We also test a strategy where choice of Hessian modes is based on their alignment with the downstream gradient. A direct parameterization of the same eigenspace serves as a control. The method trains k anchor parameters but stores an $O(Nk)$ reconstruction map. The complete calibration and training sequence is given in Appendix Section B.

3.2 ESTIMATING THE CURVATURE SPACE

At the checkpoint, define

$$\boldsymbol{g} = \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}_0), \qquad H = \nabla^2_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}_0). \tag{1}$$

Lanczos iteration Lanczos (1950) uses Hessian–vector products from automatic differentiation to estimate a pool of p eigenpairs $(\lambda_i, \boldsymbol{v}_i)$ without fully forming the $N \times N$ Hessian or loss of accuracy, see Figures 7 and 8 for evidence. We orthonormalize the returned vectors and check each eigenpair residual $\|H\boldsymbol{v}_i - \lambda_i \boldsymbol{v}_i\|_2$. If blocks are analyzed independently, we use a block-diagonal Hessian approximation and label those experiments accordingly. Spectral and block diagnostics are defined in the Appendix B.1.

3.3 FROM LOCAL CURVATURE TO SELECTED COORDINATES

The gradient-flow force is $\boldsymbol{F}(\boldsymbol{\theta}) = -\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta})$, with Jacobian $D\boldsymbol{F} = -H$. Thus a rank-k Hessian constrains the first-order change in the force only in a k-dimensional space. For a rank-k approximation H_k and k independent selected rows S, the other force components admit a local affine approximation $\boldsymbol{F}_{S^c} \simeq A_0\boldsymbol{F}_S + \boldsymbol{c}_0$. The local affine-force approximation and its one-step error bound, including the effects of discarded eigenvalues and Hessian variation, are derived in Appendix Section A.

Our training rule uses a separate, exact fact: any update restricted to a chosen k-dimensional eigenspace can be reconstructed from suitably selected physical coordinates. The force approximation motivates the choice of space; the following coordinate construction defines the restricted updates used in training.

3.4 PHYSICAL-COORDINATE RECONSTRUCTION

Let $V_k \in \mathbb{R}^{N\times k}$ contain orthonormal selected modes. We apply column-pivoted QR to V_k^{T} to select k physical coordinates S (Q-DEIM Chaturantabut & Sorensen (2010); Drmač & Gugercin (2016)). Let $P_S \in \mathbb{R}^{N\times k}$ collect their standard basis vectors. Provided $P_S^{\mathsf{T}}V_k$ is nonsingular, set

$$B = V_k(P_S^{\mathsf{T}}V_k)^{-1}, \qquad \boldsymbol{\theta}(\boldsymbol{z}) = \boldsymbol{\theta}_0 + B\boldsymbol{z}, \quad \boldsymbol{z} \in \mathbb{R}^k. \tag{2}$$

Since $P_S^{\mathsf{T}}B = I_k$, the entries of $\boldsymbol{z}$ are exactly the updates at the selected coordinates. The remaining updates follow from

$$\Delta\boldsymbol{\theta}_S = \boldsymbol{z}, \qquad \Delta\boldsymbol{\theta}_{S^c} = M\boldsymbol{z}, \qquad M = V_k[S^c,:]\big(V_k[S,:]\big)^{-1}. \tag{3}$$

This construction preserves the selected eigenspace; the proof appears in the Appendix B.4. The gradient used to train $\boldsymbol{z}$ is

$$\nabla_{\boldsymbol{z}}\mathcal{L}_{\mathcal{T}} = B^{\mathsf{T}}\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{T}}. \tag{4}$$

This expression generally requires gradient information from unselected coordinates as well. For comparison, direct training within the same span uses $\boldsymbol{\theta}(\boldsymbol{a}) = \boldsymbol{\theta}_0 + V_k\boldsymbol{a}$ with $\nabla_{\boldsymbol{a}}\mathcal{L}_{\mathcal{T}} = V_k^{\mathsf{T}}\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{T}}$. Although the two parameterizations represent the same updates, gradient descent in their coordinates can follow different paths.

3.5 SELECTING MODES FOR THE DOWNSTREAM TASK

We choose V_k from the extracted eigenpair pool. A curvature-only control selects leading positive-curvature modes. Our GATA (task-conditioned) rule also uses the downstream gradient:

$$s_i = \frac{(\boldsymbol{v}_i^{\mathsf{T}}\boldsymbol{g})^2}{\lambda_i + \gamma}, \qquad \lambda_i + \gamma > 0 \quad \text{for every candidate mode.} \tag{5}$$

Here γ is chosen such that $\lambda_i + \gamma > 0$. This score (s_i) is proportional to the predicted decrease along mode i in a damped local quadratic model. We select the k highest-scoring modes and apply the same Q-DEIM reconstruction. The quadratic derivation and the positive-curvature, gradient-only, and random-space controls are detailed in the Appendix B.5. This selection uses only the extracted eigenpair pool and the gradient at the reference checkpoint.

4 EXPERIMENTAL PROTOCOL

Models and data. We adapt bias-free CIFAR-10 classification heads (Krizhevsky, 2009) on frozen ImageNet-pretrained ResNet-18 (He et al., 2016; Russakovsky et al., 2015) and ViT-B/16 (Dosovitskiy et al., 2021) backbones. Their cached 512- and 768-dimensional features produce heads with $N = 5{,}120$ and $N = 7{,}680$ parameters. The primary experiments use 10,000 training images, including a 15% development split, and 5,000 test images. Test labels are used only for final evaluation.

Mode and anchor selection. We estimate the head Hessian with Hessian–vector products and Lanczos iteration (Lanczos, 1950), using 4,096 calibration examples, 256 candidate eigenpairs, and 64 Hutchinson probes. Candidate modes are ordered by curvature λ_i, gradient alignment $(\boldsymbol{v}_i^\top \boldsymbol{g})^2$, or the task-aligned score

$$s_i = \frac{(\boldsymbol{v}_i^\top \boldsymbol{g})^2}{\lambda_i + \gamma}, \qquad \gamma = 10^{-3}.$$

We evaluate $k \in \{5, 20, 50, 100, 150, 250\}$. For each selected basis V_k, pivoted QR chooses k physical anchors, and exact Q-DEIM (Chaturantabut & Sorensen, 2010; Drmač & Gugercin, 2016) reconstructs the dense update:

$$B = V_k (P^\top V_k)^{-1}, \qquad \Delta\boldsymbol{\theta} = B \Delta \boldsymbol{a}.$$

We verify anchor interpolation, span preservation, and the reduced-gradient identity $\nabla_{\boldsymbol{a}} \mathcal{L} = B^\top \nabla_{\boldsymbol{\theta}} \mathcal{L}$.

Primary comparison. The five-seed study compares curvature, gradient-aligned, and task-aligned spaces with dimension-matched random spaces, random coordinates, and dense head adaptation. All methods share features, initialization, splits, and evaluation sets. AdamW with zero weight decay is run for 30 epochs with batch size 256. Learning rates are selected independently on the development split from $\{3\times10^{-4}, 10^{-3}, 3\times10^{-3}, 10^{-2}\}$. We report mean test accuracy and sample standard deviation over seeds 0–4. Dense AdamW optimizes all N parameters; each subspace method optimizes exactly k variables.

PEFT and distribution comparisons. The PEFT study compares task-aligned and random anchors with LoRA, QLoRA, DoRA, AdaLoRA, IA^3, and dense AdamW. It uses the complete CIFAR-10 split, reserving 10% for development, and trains for at most 15 epochs with batch size 128 and early stopping. Methods are compared by their actual number of optimized scalars, while frozen storage and reconstruction costs are reported separately. The weight-distribution analysis uses $k = 150$ and rank-1 LoRA over five matched seeds. Runs use up to 50 epochs, and curves show the mean density and standard error across seeds.

Rank scaling. We compute complete float64 classification-head Hessians for controlled 2-, 5-, and 10-class MNIST tasks with feature widths from 8 to 256, 4,800 balanced training examples, and 120 calibration examples. Numerical rank uses $|\lambda_i| > 10^{-8} \max_j |\lambda_j|$ and is compared with

$$\operatorname{rank}(H) \le \min\{(C-1)w,\ M(C-1)\}.$$

A complementary GPT analysis reports the ranks required to capture 90%, 95%, and 99% of the spectral energy of selected MLP matrices; these structural ranks are reported separately from Hessian rank.

5 Results and Analysis

5.1 Task alignment matters most under tight budgets

Tables 1a and 1b present the key results and establish a consistent ordering across architectures. GATA is substantially more effective than dimension-matched random spaces or choosing spaces purely based on curvature, showing that their performance can only be explained by a combination of low dimensionality and task-guidance. Curvature-only spaces also improve over the random control as k grows, indicating that the Hessian provides a meaningful geometric restriction. The effect of various strategies can be seen clearly in comparing the data in Table 1 (See Figure 6 in the Appendix for a graphical representation). At small k, selecting modes only by the magnitude of the curvature (eigenvalue) leaves a large precision gap, whereas task alignment recovers most of the attainable performance. The gap contracts as the anchor count increases. This is because the curvature space included both task-relevant and task-irrelevant directions. At the largest tested anchor budgets, curvature and task-aligned selection therefore converge. Task alignment should consequently be understood as a mechanism for focusing a limited anchor budget on the task at hand, rather than as a replacement for curvature.

Table 1: **Classification-head adaptation on CIFAR-10.** Test accuracy (%) for leading-curvature, GATA, and dimension-matched random spaces as the number of anchors k increases. Each subspace method optimizes exactly k scalars. Dense AdamW optimizes the complete classification head. Results are means $\pm$ sample standard deviations over five seeds.

(a) ResNet-18

k	Curvature	GATA	Random
5	11 ± 1.8	$\mathbf{33 \pm 3.5}$	11 ± 3.8
20	34 ± 4.4	$\mathbf{54 \pm 4.2}$	18 ± 6.6
50	57 ± 3.6	$\mathbf{69 \pm 2.3}$	30 ± 3.0
100	69 ± 3.2	$\mathbf{75 \pm 0.6}$	40 ± 1.9
150	75 ± 0.7	$\mathbf{77 \pm 0.7}$	51 ± 1.7
250	78 ± 0.9	$\mathbf{79 \pm 0.9}$	61 ± 1.5
Dense AdamW ($N = 5{,}120$): 85 ± 0.3			

(b) ViT-B/16

k	Curvature	GATA	Random
5	10 ± 4.4	$\mathbf{43 \pm 6.9}$	10 ± 3.5
20	30 ± 6.6	$\mathbf{83 \pm 1.8}$	21 ± 4.8
50	81 ± 3.1	$\mathbf{90 \pm 0.2}$	35 ± 4.5
100	91 ± 0.3	$\mathbf{92 \pm 0.3}$	54 ± 2.7
150	92 ± 0.2	$\mathbf{93 \pm 0.1}$	67 ± 2.5
250	$\mathbf{93 \pm 0.2}$	$\mathbf{93 \pm 0.2}$	75 ± 1.6
Dense AdamW ($N = 7{,}680$): 94 ± 0.2			

The results also separate subspace selection from coordinate representation. GATA and its Q-DEIM parameterization follow closely across the tested ViT budgets. Q-DEIM therefore exposes a useful subspace through physical weights without requiring the update itself to be sparse. The k selected anchors act as coordinates, while the affine reconstruction map propagates their changes across the remaining parameters.

Applying the same coordinate construction to the curvature-only space does not produce the task-aligned result. This distinction is central: Q-DEIM determines how a chosen space is controlled, whereas task alignment determines which space is chosen. The close agreement between direct and Q-DEIM forms supports the use of physical anchors as an implementation of the selected geometry rather than as an independent source of predictive power.

5.2 Anchor efficiency relative to PEFT baselines

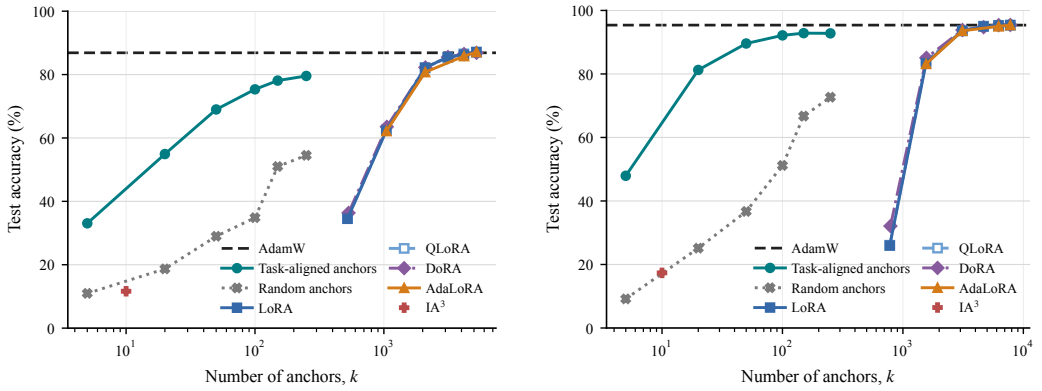
Figure 2 places physical anchors and rank-based PEFT methods on a common optimized-scalar axis. The main difference is obtained by the horizontal spacing between the curves. Task-aligned anchors obtain useful adaptation with tens or hundreds of optimizer-maintained values, while the LoRA-family methods begin at several hundred values and approach the AdamW ceiling as their ranks increase. Random anchors improve much more slowly, again demonstrating that the gain comes from selection rather than coordinate count.

The pattern is reproduced by both ResNet-18 and ViT-B/16 when trained from the same seed. ViT benefits particularly strongly from task-aligned anchors at small budgets, while ResNet exhibits a more gradual accuracy increase. In both cases, the task-aligned curve occupies the favorable low-budget region of the plot. Rank-based methods eventually reach higher absolute accuracy as their optimized-scalar counts approach the dense-head budget, but they require substantially more optimizer variables to do so.

This comparison concerns the dimension of the optimized state. Q-DEIM also requires the construction and storage of a basis and reconstruction map, whereas LoRA stores factorized adapters. The anchor results therefore demonstrate optimizer-state and communication efficiency; total memory and computation may depend on how the reconstruction map is represented and applied.

5.3 The restricted coordinates produce dense, structured updates

Figure 3 is a plot of the probability density function of the weights in a ViT experiment (See Figure 5 in the Appendix for a similar result on ResNET). These plots show that the different parameterizations do not merely reach different accuracies; they produce different but consistent families of updates. The QR-based methods concentrate final weights and signed updates more strongly around zero than AdamW. Their update-magnitude distributions are nevertheless spread across the head, confirming that optimizing a small number of anchors does not restrict learning to those physical entries; the affine map serves the purpose of propagating that anchor update information to the dense head.



(a) ResNet-18 on CIFAR-10.

(b) ViT-B/16 on CIFAR-10.

Figure 2: **Test accuracy versus number of anchors.** Task-aligned physical anchors are compared with random anchors and rank-based PEFT methods. The horizontal coordinate gives the physical-anchor count for the anchor methods; PEFT baselines are positioned according to their optimized-variable count. The dashed line denotes dense classification-head tuning with AdamW.

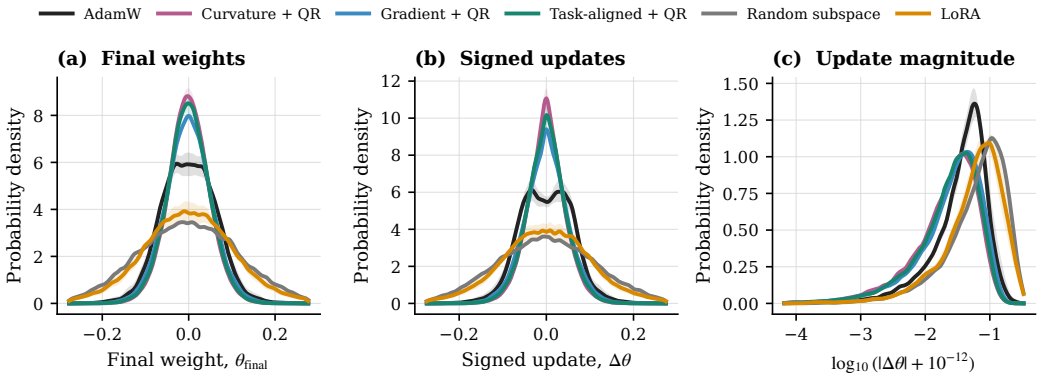


Figure 3: **ViT-B/16/CIFAR-10 weight and update distributions.** The panels show the distributions of final classification-head weights, signed changes from initialization, and logarithmic update magnitudes. The subspace methods use $k = 150$ anchors, while LoRA uses rank 1. Curves show the mean across five seeds, and shaded regions denote the standard error of the mean. Task-aligned, curvature, and gradient-based QR parameterizations produce dense reconstructed updates whose distributions differ from AdamW, a random subspace, and LoRA.

This observation clarifies the role of the affine reconstruction. A sparse set of controlled anchors generates a dense parameter update whose structure is determined by the selected basis. Random spaces and LoRA produce broader distributions because they impose different relationships among parameter changes. The similarity of the task-aligned distributions across multiple model architectures also suggests that this behavior follows from the parameterization rather than from a single backbone.

6 Scaling of the Curvature Dimension

Figure 4 examines how the dimension of the curved subspace changes with model size and task complexity. The two experiments use slightly different rank definitions. The GPT experiment measures the number of leading modes required to retain a specified fraction of the extracted spectral energy. The MNIST experiment computes the complete Hessian spectrum and counts eigenvalues above a relative threshold of 10^{-8}. Together, they show that curvature dimension does not grow in proportion to the model parameter dimension.

Sublinear scaling in GPT MLP matrices. Figure 4a reports the spectral-energy rank of the first MLP projection in Nano-GPT, Micro-GPT, and Small-GPT. These are scaled configurations based on the GPT-2 architecture (Radford et al., 2019) and are evaluated on WikiText-2 (Merity et al., 2017). The matrix dimension increases from approximately 6.6×10^4 to 2.4×10^6, a factor of about 36. Over the same range, the ranks required to retain 90%, 95%, and 99% of the extracted spectral energy increase much more slowly. The 99% rank is already close to saturation at the smallest scale, while the lower energy thresholds rise gradually before flattening.

This behavior is consistent with substantial overparameterization of the MLP blocks for the language-modeling objective. Increasing matrix size introduces many additional parameter directions, but only a smaller subset carries substantial curvature for the observed task and data distribution. Consequently, the model dimension grows rapidly while the dominant curvature dimension grows sublinearly and saturates.

Transition from linear growth to saturation. Figure 4b isolates this transition using complete float64 Hessians. We vary feature width while holding the dataset, calibration-set size, and optimization protocol fixed. Separate curves are reported for nested 2-, 5-, and 10-class tasks constructed from MNIST (LeCun et al., 1998).

At small widths, almost every identifiable direction contributes to the Hessian rank. If the classifier contains $N = Cw$ parameters for C classes and feature width w, the observed initial behavior is

$$k \simeq (C-1)w = \frac{C-1}{C}N.$$

The initial log–log slope is therefore approximately one. As the feature width increases, each curve bends away from linear growth and approaches a class and sample dependent plateau. With M calibration examples, the rank is bounded by

$$k \leq \min\{(C-1)w,\ M(C-1)\}.$$

The measured curves closely follow this linear-to-saturating structure. Additional model parameters initially introduce new curvature directions, but after the task-dependent threshold is reached, further width primarily adds locally redundant directions. As an aside, it is possible that such redundant directions may also serve a secondary purpose during training, for the training trajectory guided by gradient descent to find its way to the global optimum.

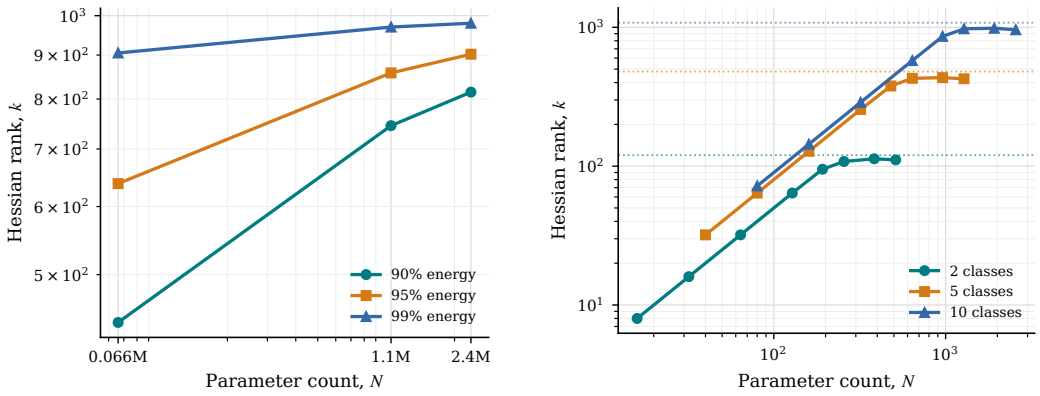
Dependence on task complexity. The saturation level increases systematically from the 2-class task to the 5-class and 10-class tasks. A larger number of classes introduces more independent output contrasts and therefore supports a higher-dimensional curvature space. Model size controls whether the available task structure can be represented, while task complexity controls the level at which the rank saturates. The combined behavior can be summarized by the approximation

$$k(N, C) \approx \min\left\{\frac{C-1}{C}N,\ M(C-1)\right\}.$$

This result gives a direct interpretation of low curvature rank in overparameterized models. The rank is not required to scale indefinitely with the number of parameters. Once the model is sufficiently wide for a fixed task, the curvature dimension is limited primarily by the information and independent constraints supplied by that task.

7 Discussion

The results presented above point to two roles of loss geometry. Hessian curvature supplies a compact set of candidate directions, while the task gradient identifies those most useful when k is small. Curvature-only and task-aligned spaces approach similar accuracy as the budget grows. Q-DEIM then represents the chosen space through k physical anchors: it preserves the space and produces a generally dense update through a linear affine map. The close agreement between direct subspace



(a) Spectral-energy rank of the first MLP projection in Nano-GPT, Micro-GPT, and Small-GPT.

(b) Complete Hessian rank for 2-, 5-, and 10-class MNIST tasks as feature width increases.

Figure 4: **Curvature dimension across model scales and tasks. (a)** The parameter dimension of the GPT MLP matrices grows much faster than the rank required to retain a fixed fraction of their extracted curvature energy. **(b)** Exact MNIST Hessians reveal the corresponding linear-to-saturating transition. Rank initially increases with model size and then approaches a task-dependent ceiling that rises with the number of classes.

optimization and anchor-based training shows that physical coordinates can retain the utility of the selected space.

The selected geometry offers a way to adapt a large model through a small set of task-relevant coordinates. As training moves away from the initial checkpoint, recalibration could keep those coordinates aligned with the changing loss landscape. The same anchors could provide a compact interface for federated learning: clients would exchange only k update values while a shared affine map reconstructs the remainder dense update. Realizing this at model scale will require efficient curvature estimation and reconstruction, as well as evidence that recalibration improves longer runs. Appendix Section E develops this federated learning application.

8 Conclusion

We introduced GATA, a method for selecting a compact, task-relevant adaptation space from the local geometry of the loss. The Hessian was shown to be of a rank much less than its dimension and therefore it is clear that not all parameters are linearly independent. This process can then be used to identify directions of curvature that are independent, and the downstream gradient helps downselect directions useful for adaptation. Q-DEIM represents the selected space using k physical parameter coordinates, or anchors. A linear map turns changes to these anchors into dense parameter updates.

In classification-head experiments with ResNet-18 and ViT-B/16, GATA outperforms dimension-matched random spaces, especially at small coordinate budgets. Its anchor representation closely matches direct training in the selected space. Comparisons with LoRA, QLoRA, DoRA, AdaLoRA, and IA^3 show competitive accuracy with a compact optimizer state.

These results suggest that local loss geometry and task information can guide the choice of an adaptation space. They also motivate extending GATA to multiple layers, updating the selected space during training, and reducing the cost of storing and applying the reconstruction map. Physical anchors may provide a compact interface for federated fine-tuning when clients share that map.

Ethics Statement

This work studies optimization and communication methods using established image-classification datasets and pretrained models. It involves no human subjects, sensitive personal data, or domain-specific deployment. Reduced communication in federated learning does not itself provide privacy; any privacy guarantee would require a separate threat model and dedicated analysis.

Reproducibility Statement

The appendix specifies the datasets, splits, calibration sets, Hessian construction, eigensolver settings, optimization grids, stopping criteria, and seed aggregation used in the experiments. The supplementary code reproduces the experiments and figures and records raw per-seed results in CSV, JSON, and parameter-array formats. All reported figures and tables are generated from these experimental records.

AI Use Statement

Generative AI tools assisted with code generation and debugging, figure preparation, and language editing. The authors were the original idea generators and planned executed the experiments, verified numerical results against the raw outputs, and reviewed all AI-assisted code, mathematical statements, figures, citations, and manuscript text. The authors take responsibility for the complete content of this work.

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2021.

Saifon Chaturantabut and Danny C. Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 2010. doi: 10.1137/090766498.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Zlatko Drmač and Serkan Gugercin. A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions. *SIAM Journal on Scientific Computing*, 2016. doi: 10.1137/15M1019271.

Zlatko Drmač and Serkan Gugercin. A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions. *SIAM Journal on Scientific Computing*, 38(2):A631–A648, 2016.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

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

Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4): 255–282, 1950. doi: 10.6028/jres.045.026.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*, 2018.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *International Conference on Machine Learning*, 2024.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115 (3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*, 2023.

A Error bound for thresholded low-rank Hessian updates

We use the spectral (operator) norm for matrices and Euclidean norm for vectors. Let $\mathcal{L} : \mathbb{R}^N \to \mathbb{R}$ be twice continuously differentiable on the closed ball $B_r(\mathbf{w}_0) = \{\mathbf{w} : \|\mathbf{w} - \mathbf{w}_0\|_2 \le r\}$, and let $\mathbf{F}(\mathbf{w}) = -\nabla\mathcal{L}(\mathbf{w})$ and $\mathbf{H}(\mathbf{w}) = \nabla^2\mathcal{L}(\mathbf{w})$. Assume the Hessian is Lipschitz on this ball:

$$\|\mathbf{H}(\mathbf{w}) - \mathbf{H}(\mathbf{v})\|_2 \le L_H\|\mathbf{w} - \mathbf{v}\|_2 \quad \text{for all } \mathbf{w}, \mathbf{v} \in B_r(\mathbf{w}_0). \tag{6}$$

The assumption refers to a fixed, deterministic loss function. If gradients are estimated from different minibatches, their sampling variability is an additional source of error not covered by this bound.

Since $\mathbf{H}_0 := \mathbf{H}(\mathbf{w}_0)$ is symmetric, write its eigendecomposition as $\mathbf{H}_0 = Q\,\mathrm{diag}(\lambda_1, \dots, \lambda_N)Q^\mathsf{T}$, with $|\lambda_1| \ge \cdots \ge |\lambda_N|$. For an absolute eigenvalue threshold $\tau > 0$, define

$$k = \#\{j : |\lambda_j| > \tau\}, \qquad \mathbf{H}_{0,k} = \sum_{j=1}^{k} \lambda_j \mathbf{q}_j\mathbf{q}_j^\mathsf{T}, \qquad R_0 = \mathbf{H}_0 - \mathbf{H}_{0,k}. \tag{7}$$

We assume $1 \le k < N$, so $\mathrm{rank}\,\mathbf{H}_{0,k} = k$ and $\|R_0\|_2 = |\lambda_{k+1}| \le \tau$. If $k = 0$, the zero-rank case may be treated separately as a locally constant driving force.

Select k row indices I for which $(\mathbf{H}_{0,k})_I$ has full row rank, and let D contain the other $N - k$ indices. Such an I exists because $\mathbf{H}_{0,k}$ has rank k. Define

$$A_0 = (\mathbf{H}_{0,k})_D(\mathbf{H}_{0,k})_I^+, \qquad \mathbf{c}_0 = \mathbf{F}_D(\mathbf{w}_0) - A_0\mathbf{F}_I(\mathbf{w}_0), \tag{8}$$

where $+$ denotes the Moore–Penrose pseudoinverse. Every row of $(\mathbf{H}_{0,k})_D$ lies in the row span of $(\mathbf{H}_{0,k})_I$, and hence $(\mathbf{H}_{0,k})_D = A_0(\mathbf{H}_{0,k})_I$. In this full-row-rank setting A_0 is unique.

Proposition 1 (Local affine-force error)**.** *For every* $\mathbf{w} \in B_r(\mathbf{w}_0)$*, set* $d = \|\mathbf{w} - \mathbf{w}_0\|_2$*. Then*

$$\left\|\mathbf{F}_D(\mathbf{w}) - A_0\mathbf{F}_I(\mathbf{w}) - \mathbf{c}_0\right\|_2 \le \sqrt{1 + \|A_0\|_2^2}\left(|\lambda_{k+1}|d + \frac{L_H}{2}d^2\right) \le \sqrt{1 + \|A_0\|_2^2}\left(\tau d + \frac{L_H}{2}d^2\right). \tag{9}$$

For a gradient-descent step with learning rate $\eta_t > 0$*, define* $\mathbf{u}^{(t)} = \eta_t\mathbf{F}(\mathbf{w}^{(t)})$*. If the inferred dependent update is* $\widehat{\mathbf{u}}_D^{(t)} = \eta_t[A_0\mathbf{F}_I(\mathbf{w}^{(t)}) + \mathbf{c}_0]$*, its error is at most* η_t *times the right-hand side of Eq. equation 9 at* $\mathbf{w} = \mathbf{w}^{(t)}$*.*

Proof. Reorder coordinates as (I, D) and let $B = [-A_0 \;\; I_{N-k}]$. The force relation is anchored at $\mathbf{w}_0$, so, with $\boldsymbol{\xi} = \mathbf{w} - \mathbf{w}_0$,

$$\mathbf{F}_D(\mathbf{w}) - A_0\mathbf{F}_I(\mathbf{w}) - \mathbf{c}_0 = B[\mathbf{F}(\mathbf{w}) - \mathbf{F}(\mathbf{w}_0)]. \tag{10}$$

By the integral form of Taylor's theorem and Eq. equation 6,

$$\mathbf{F}(\mathbf{w}) - \mathbf{F}(\mathbf{w}_0) = -\mathbf{H}_0\boldsymbol{\xi} + \boldsymbol{\rho}, \qquad \|\boldsymbol{\rho}\|_2 \le \frac{L_H}{2}d^2. \tag{11}$$

Since $B\mathbf{H}_{0,k} = 0$ and $\mathbf{H}_0 = \mathbf{H}_{0,k} + R_0$, the error is $-BR_0\boldsymbol{\xi} + B\boldsymbol{\rho}$. Finally, $BB^\mathsf{T} = I_{N-k} + A_0A_0^\mathsf{T}$ implies $\|B\|_2 = \sqrt{1 + \|A_0\|_2^2}$. Applying the operator norm inequality and $\|R_0\|_2 = |\lambda_{k+1}| \le \tau$ proves the result. □

A fitted affine map. The preceding bound uses the exact truncated-Hessian map and an exact full-gradient anchor. For fitted coefficients $(\widehat{A}, \widehat{\mathbf{c}})$, define the measurable calibration residual and the map error by

$$\delta_0 = \|\mathbf{F}_D(\mathbf{w}_0) - \widehat{A}\mathbf{F}_I(\mathbf{w}_0) - \widehat{\mathbf{c}}\|_2, \qquad \Delta_A = \|\widehat{A} - A_0\|_2. \tag{12}$$

The same expansion gives, for $d = \|\mathbf{w} - \mathbf{w}_0\|_2 \le r$,

$$\begin{aligned}&\|\mathbf{F}_D(\mathbf{w}) - \widehat{A}\mathbf{F}_I(\mathbf{w}) - \widehat{\mathbf{c}}\|_2 \\ &\quad\le \delta_0 + \left[\sqrt{1 + \|A_0\|_2^2}\,|\lambda_{k+1}| + \Delta_A\|\mathbf{H}_I(\mathbf{w}_0)\|_2\right]d + \frac{L_H}{2}\sqrt{1 + \|\widehat{A}\|_2^2}\,d^2.\end{aligned} \tag{13}$$

To obtain this expression, write $[-\widehat{A} \;\; I]\mathbf{H}_0 = [-A_0 \;\; I]R_0 - (\widehat{A} - A_0)\mathbf{H}_I(\mathbf{w}_0)$ and bound the Taylor remainder using $\|[-\widehat{A} \;\; I]\|_2 = \sqrt{1 + \|\widehat{A}\|_2^2}$. If the fitted intercept is reset using one full-gradient measurement at $\mathbf{w}_0$, then $\delta_0 = 0$. Without an estimate of Δ_A or a direct bound on the Hessian-row residual, the threshold τ alone does not certify the fitted map.

Comment on the bound. Even though the ratio k/N does not occur in Eqs. equation 9–equation 13, it is useful to discuss the implications of such a proposition. A small k is useful here since the discarded eigenvalues are small in the chosen units. Therefore, the selected rows yield a controlled $\|A_0\|_2$, and the training weights stay in a region where the local estimate is informative. Small eigenvalues describe negligible second-order curvature at $\mathbf{w}_0$ and therefore imply that the loss is locally flat. The bounds strictly control the error in one update evaluated at a specified weight vector. However, our empirical evidence (from comparing our method to full stochastic gradient descent) has pointed to the proposition that they seem to bound the separation of entire training trajectories until convergence.

B Algorithmic Summary

B.1 Flatness, spectral concentration, and block structure

We measure how much squared curvature is captured by the k eigenvalues with largest magnitude:

$$E_H(k) = \frac{\sum_{i=1}^k \lambda_i^2}{\mathrm{tr}(H^2)}. \tag{14}$$

We estimate the denominator separately using Hutchinson probes, $\mathrm{tr}(H^2) = \mathbb{E}_{\boldsymbol{z}}\|H\boldsymbol{z}\|_2^2$, where each entry of $\boldsymbol{z}$ is independently $+1$ or -1 with equal probability. This avoids treating the eigenvalues returned by Lanczos as the full spectrum.

For positive semidefinite H, we report the participation ratio

$$r_{\mathrm{eff}}(H) = \frac{\mathrm{tr}(H)^2}{\mathrm{tr}(H^2)}. \tag{15}$$

For indefinite H, we use $(\mathrm{tr}\,|H|)^2/\,\mathrm{tr}(H^2)$, estimating $\mathrm{tr}\,|H|$ by stochastic Lanczos quadrature. We also report the fraction of eigenvalues below a relative curvature threshold:

$$F_\epsilon(H) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}\{|\lambda_i| \le \epsilon\|H\|_2\}. \tag{16}$$

This measures local quadratic flatness, not the usefulness of those directions for adaptation.

Partition the parameters into B architectural blocks, with Hessian blocks H_{bc}. We measure the share of squared curvature within blocks:

$$C_{\mathrm{block}} = \frac{\sum_{b=1}^{B}\|H_{bb}\|_F^2}{\|H\|_F^2}, \qquad C_{\mathrm{cross}} = 1 - C_{\mathrm{block}}. \tag{17}$$

Full and block-restricted Hessian–vector products estimate these shares. A large C_{block} supports a block-diagonal approximation.

Finally, because entrywise sparsity depends on the parameter coordinates, we report it as a function of the threshold β:

$$S_H(\beta) = 1 - \frac{1}{N^2}\sum_{i,j} \mathbf{1}\{|H_{ij}| > \beta\|H\|_F/N\}. \tag{18}$$

We compute this curve only for blocks small enough to form explicitly. Reordered heat maps illustrate the block partition but do not establish sparsity on their own.

Alternate affine map construction approach. Alternatively, we can also generate the affine relation from M early force snapshots obtained using Gradient Descent. Let $U_S \in \mathbb{R}^{k\times M}$ and $U_{S^c} \in \mathbb{R}^{(N-k)\times M}$ contain their selected and remaining components. Solve

$$(\widehat{A}, \widehat{\boldsymbol{c}}) = \arg\min_{A,\boldsymbol{c}} \|U_{S^c} - AU_S - \boldsymbol{c}\mathbf{1}_M^{\mathsf{T}}\|_F^2. \tag{19}$$

For column-centered matrices $\widetilde{U}_S$ and $\widetilde{U}_{S^c}$, one solution is

$$\widehat{A} = \widetilde{U}_{S^c}\widetilde{U}_S^{+}, \qquad \widehat{\boldsymbol{c}} = \overline{\boldsymbol{u}}_{S^c} - \widehat{A}\overline{\boldsymbol{u}}_S. \tag{20}$$

This provides a calibration baseline. A unique fit requires the centered selected snapshots to span $\mathbb{R}^k$, which consecutive training steps need not achieve.

Task-Aligned Hessian–DEIM. Given a reference $\boldsymbol{\theta}_0$, calibration data, candidate count p, dimension k, and damping γ:

1. Compute $\boldsymbol{g}$ and expose the HVP operator at $\boldsymbol{\theta}_0$.
2. Extract p candidate eigenpairs with Lanczos.
3. Compute $s_i = (\boldsymbol{v}_i^\top \boldsymbol{g})^2/(\lambda_i + \gamma)$ and retain the k highest-scoring eigenvectors in V_k.
4. Apply pivoted QR to $V_k^\top$ to obtain physical indices S.
5. Form $B = V_k(V_k[S,:])^{-1}$ and initialize $\boldsymbol{z} = 0$.
6. Optimize $\mathcal{L}(\boldsymbol{\theta}_0 + B\boldsymbol{z})$ using reduced gradients $B^\top\nabla_{\boldsymbol{\theta}}\mathcal{L}$.
7. Optionally recalibrate around the current model when an alignment or interval criterion is met.

B.2 Curvature and block diagnostics

The controlled classification-head experiments use the complete head Hessian. In experiments that treat several parameter blocks independently, we explicitly approximate H by

$$\widetilde{H} = \mathrm{diag}(H_{11}, \ldots, H_{BB}), \tag{21}$$

thereby omitting cross-block curvature. We label such experiments blockwise. A linear classification head on fixed features with a convex loss has a positive-semidefinite head Hessian; internal neural-network blocks need not be positive semidefinite. We report positive and negative eigenmodes separately where applicable.

The spectral-energy, participation-ratio, flatness, block-concentration, and entrywise-sparsity diagnostics are defined in Section B.1. The independent probe estimate of $\mathrm{tr}(H^2)$ prevents the extracted Lanczos spectrum from being treated as the complete spectrum.

B.3 Local affine force relation and error bound

Write $\boldsymbol{F}(\boldsymbol{\theta}) = -\nabla\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta})$ and let $H = \nabla^2\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}_0)$. Order eigenpairs by decreasing $|\lambda_i|$ and write

$$H = H_k + R, \qquad H_k = V_k\Lambda_k V_k^{\mathsf{T}}, \qquad \|R\|_2 = |\lambda_{k+1}|. \tag{22}$$

In this subsection V_k denotes the leading modes used to approximate the full Hessian. The task-conditioned basis in the main method may select different modes from the candidate eigenpool.

Select k linearly independent rows S of H_k and let S^c be their complement. Define

$$A_0 = H_k[S^c, :]H_k[S, :]^+, \qquad \boldsymbol{c}_0 = \boldsymbol{F}_{S^c}(\boldsymbol{\theta}_0) - A_0\boldsymbol{F}_S(\boldsymbol{\theta}_0). \tag{23}$$

Because the selected rows span the row space of H_k, $H_k[S^c, :] = A_0H_k[S, :]$. Assume the Hessian is Lipschitz in spectral norm with constant L_H on a ball around $\boldsymbol{\theta}_0$. For $d = \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_2$ inside that ball,

$$\|\boldsymbol{F}_{S^c}(\boldsymbol{\theta}) - A_0\boldsymbol{F}_S(\boldsymbol{\theta}) - \boldsymbol{c}_0\|_2 \le \sqrt{1 + \|A_0\|_2^2}\left(|\lambda_{k+1}|d + \frac{L_H}{2}d^2\right). \tag{24}$$

To prove this, reorder components as (S, S^c) and set $C = [-A_0 \;\; I_{N-k}]$. The intercept makes the error zero at the reference point, while $CH_k = 0$. Taylor's theorem gives

$$\boldsymbol{F}(\boldsymbol{\theta}) - \boldsymbol{F}(\boldsymbol{\theta}_0) = -H(\boldsymbol{\theta} - \boldsymbol{\theta}_0) + \boldsymbol{\rho}, \qquad \|\boldsymbol{\rho}\|_2 \le \frac{L_H}{2}d^2. \tag{25}$$

The force error is therefore $-CR(\boldsymbol{\theta}-\boldsymbol{\theta}_0)+C\boldsymbol{\rho}$. Since $\|C\|_2 = \sqrt{1 + \|A_0\|_2^2}$, the triangle inequality proves Eq. equation 24. Multiplication by a learning rate gives a one-step update error. The bound applies locally to a force evaluated at a given parameter vector; it does not bound an entire training trajectory.

B.4 Interpolation and span preservation

Let $V_k \in \mathbb{R}^{N\times k}$ have orthonormal columns and let $P_S \in \mathbb{R}^{N\times k}$ select coordinates for which $P_S^{\mathsf{T}}V_k$ is invertible. Set $B = V_k(P_S^{\mathsf{T}}V_k)^{-1}$.

Proposition 2 (Interpolation and span preservation)**.** *The basis B satisfies $P_S^{\mathsf{T}}B = I_k$ and* $\mathrm{span}(B) = \mathrm{span}(V_k)$.

Proof. Multiplication gives $P_S^{\mathsf{T}}B = (P_S^{\mathsf{T}}V_k)(P_S^{\mathsf{T}}V_k)^{-1} = I_k$. Multiplication of V_k on the right by an invertible matrix preserves its column span. □

Thus $\boldsymbol{z}$ in $\boldsymbol{\theta} = \boldsymbol{\theta}_0 + B\boldsymbol{z}$ is the update at the selected physical coordinates. With coordinates ordered as (S, S^c), B has blocks $[I_k^{\mathsf{T}} \; M^{\mathsf{T}}]^{\mathsf{T}}$, where $M = V_k[S^c, :](V_k[S, :])^{-1}$. Its reduced gradient is

$$\nabla_{\boldsymbol{z}}\mathcal{L}_{\mathcal{T}} = \nabla_{\boldsymbol{\theta}_S}\mathcal{L}_{\mathcal{T}} + M^{\mathsf{T}}\nabla_{\boldsymbol{\theta}_{S^c}}\mathcal{L}_{\mathcal{T}}. \tag{26}$$

Although direct and physical-coordinate representations span the same space, ordinary gradient descent induces parameter-space steps $-\eta V_kV_k^{\mathsf{T}}\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{T}}$ and $-\eta BB^{\mathsf{T}}\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{T}}$, respectively. We report $\kappa(V_k[S, :])$ and compare both parameterizations empirically.

B.5 TASK-CONDITIONED SCORE AND CONTROLS

At the reference checkpoint, consider the damped quadratic model

$$q(\boldsymbol{\delta}) = \mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}_0) + \boldsymbol{g}^{\mathsf{T}}\boldsymbol{\delta} + \tfrac{1}{2}\boldsymbol{\delta}^{\mathsf{T}}(H + \gamma I)\boldsymbol{\delta}. \tag{27}$$

We choose γ so that $\lambda_i + \gamma > 0$ for every retained candidate mode. Along $\boldsymbol{\delta} = \alpha_i \boldsymbol{v}_i$, minimization gives

$$\alpha_i^* = -\frac{\boldsymbol{v}_i^{\mathsf{T}}\boldsymbol{g}}{\lambda_i + \gamma}, \qquad q(\mathbf{0}) - q(\alpha_i^*\boldsymbol{v}_i) = \frac{(\boldsymbol{v}_i^{\mathsf{T}}\boldsymbol{g})^2}{2(\lambda_i + \gamma)}. \tag{28}$$

The main-text score omits the common factor $1/2$. Curvature-only selection orders positive modes by λ_i, while gradient-only selection orders by $(\boldsymbol{v}_i^{\mathsf{T}}\boldsymbol{g})^2$. Random orthonormal spaces control for subspace dimension. All policies search only within their specified candidate pool.

B.6 STABILIZATION AND RECALIBRATION

When $V_k[S, :]$ is poorly conditioned, exact interpolation can amplify numerical error. A stabilized variant replaces its inverse by a truncated pseudoinverse and optionally clips the resulting map:

$$M_{\mathrm{stab}} = \mathrm{clip}\Big(V_k[S^c, :]\big(V_k[S, :]\big)_{\tau}^{+}, -c, c\Big). \tag{29}$$

The stabilized basis has selected rows I_k and unselected rows M_{stab}. This clipping or truncation generally changes its span. We therefore treat exact and stabilized interpolation as separate methods and report their principal-angle distance.

The static variant keeps its basis, selected coordinates, and map fixed throughout fine-tuning. At predetermined intervals, the dynamic variant recalculates the gradient, candidate eigenpairs, selected subspace, coordinates, and map. Each recalibration takes the current model as its new reference so the represented model parameters remain continuous. Unless a transfer rule is tested, optimizer state is reset at that point. We compare variants separately because recalibration changes both computational cost and the sequence of available update spaces.

B.7 ALIGNMENT, BASELINES, AND OPERATIONAL DIMENSION

For a nonzero vector $\boldsymbol{u}$ and an orthonormal basis U, the fraction captured by that space is

$$A(\boldsymbol{u}, U) = \frac{\|U^{\mathsf{T}}\boldsymbol{u}\|_2^2}{\|\boldsymbol{u}\|_2^2}. \tag{30}$$

We apply this measure to the initial gradient and the final dense update. The former measures local descent coverage; the latter checks alignment with longer-run adaptation.

To choose a budget without using the test set, let $L^{\mathrm{dev}}_{\mathrm{frozen}}$, $L^{\mathrm{dev}}_{\mathrm{dense}}$, and L^{dev}_k denote development-set losses under the corresponding adaptations. When $L^{\mathrm{dev}}_{\mathrm{frozen}} > L^{\mathrm{dev}}_{\mathrm{dense}}$, define development-set recovery by

$$R_{\mathrm{dev}}(k) = \frac{L^{\mathrm{dev}}_{\mathrm{frozen}} - L^{\mathrm{dev}}_k}{L^{\mathrm{dev}}_{\mathrm{frozen}} - L^{\mathrm{dev}}_{\mathrm{dense}}}. \tag{31}$$

The operational dimension k_q is the smallest tested budget whose development-set recovery reaches at least q. We report unsmoothed recovery curves because performance need not increase monotonically with k. If no tested budget reaches q, we mark k_q as right-censored. After selecting k on the development set, we report its held-out performance using $R(k)$ from the main Methods section.

Controls include dense tuning, LoRA, random orthonormal spaces, uniformly selected physical coordinates, leading-curvature modes, gradient-only modes, task-conditioned modes, and an SVD of the dense update obtained after training. The last is a retrospective oracle for the existence of a compact update, not a prospective training algorithm. We compare direct and physical-coordinate training within the same selected span. For LoRA on an $m \times n$ weight matrix, rank r corresponds to approximately $r(m + n)$ trainable scalars; matching r directly to k would compare different parameter budgets.

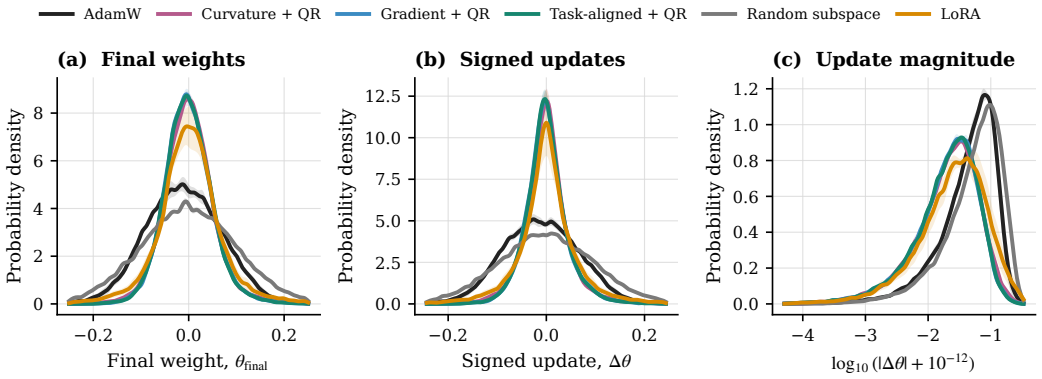


Figure 5: **ResNet-18/CIFAR-10 weight and update distributions.** The panels show final classification-head weights, signed changes from initialization, and logarithmic update magnitudes. Subspace methods use $k = 150$ anchors, while LoRA uses rank 1. Curves show the mean across five seeds, and shaded regions denote the standard error of the mean.

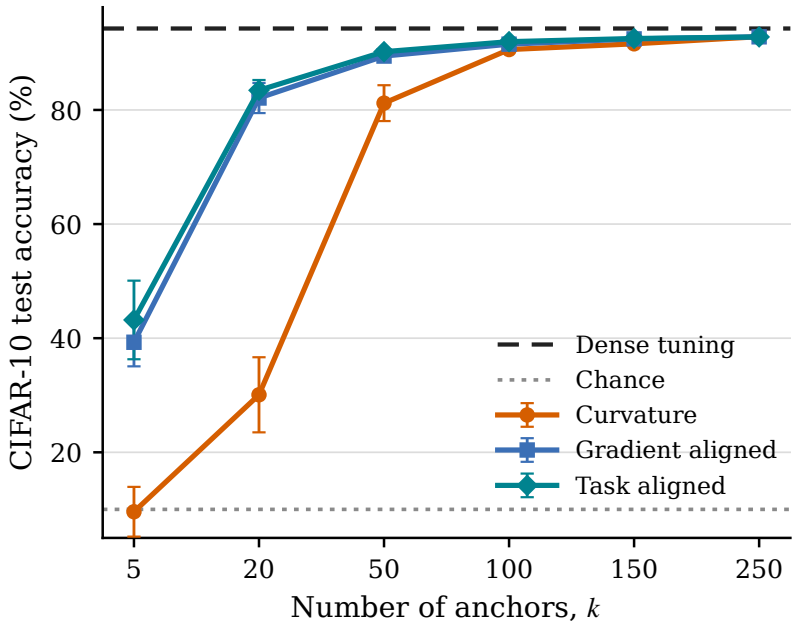


Figure 6: **Effect of mode selection on ViT-B/16/CIFAR-10.** Test accuracy as a function of adaptation dimension k for leading-curvature, gradient-aligned, and task-aligned Hessian modes. Task alignment provides its largest advantage at small k, where the limited coordinate budget makes direction selection most important. The methods converge as k increases and progressively larger portions of the candidate curvature space are retained. The dashed and dotted references denote dense head tuning and chance accuracy, respectively. Points and error bars report the mean and sample standard deviation over five seeds.

C Additional Results

C.1 Validation of the matrix-free Lanczos approximation

We validate the matrix-free Lanczos approximation in a setting where the complete network Hessian can be formed and diagonalized explicitly. The test uses a 196–8–8–10 tanh MLP with $N = 1{,}738$ parameters and 80 calibration examples. The full $1{,}738 \times 1{,}738$ Hessian is constructed in float64, giving a numerical rank of 792 at a relative threshold of 10^{-9}. We then recover the same eigensystem using only Hessian–vector products and Lanczos iteration. This comparison evaluates the

Table 2: **ResNet-18/CIFAR-10 anchor-budget results.** Test accuracy from seed 0. Physical-anchor methods optimize exactly k scalar coordinates. For the rank-parameterized PEFT methods, each entry reports test accuracy followed by the number of optimized scalars in parentheses. Classification-head AdamW obtains 86.87% accuracy while optimizing $5{,}120$ scalars.

Physical-anchor methods

Number of anchors k	Task-aligned anchors	Random anchors
5	33.07	10.97
20	54.92	18.69
50	68.99	28.98
100	75.34	34.85
150	78.10	50.99
250	79.56	54.52

Rank-parameterized PEFT methods

Rank r	LoRA	QLoRA	DoRA	AdaLoRA
1	34.49 (522)	34.57 (522)	36.36 (532)	62.16 (1,046)
2	62.29 (1,044)	62.61 (1,044)	63.52 (1,054)	80.76 (2,092)
4	82.16 (2,088)	82.14 (2,088)	82.25 (2,098)	85.81 (4,184)
6	85.52 (3,132)	85.38 (3,132)	85.53 (3,142)	86.85 (5,230)
8	86.24 (4,176)	86.54 (4,176)	86.48 (4,186)	87.18 (5,230)
10	87.13 (5,220)	87.02 (5,220)	86.86 (5,230)	87.29 (5,230)

Method	Test accuracy (%)	Optimized scalars
IA^3	11.64	10
AdamW	86.87	5,120

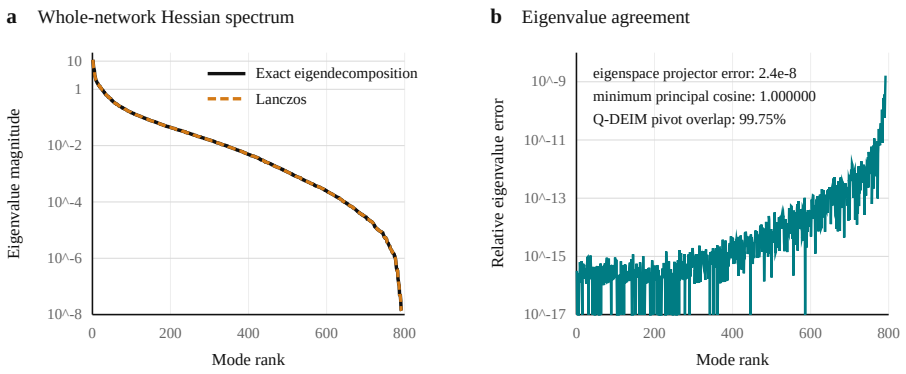


Figure 7: **Lanczos accurately recovers the whole-network Hessian eigensystem. (a)** Magnitudes of all 792 numerically nonzero eigenvalues, ordered by decreasing magnitude. The spectrum obtained by matrix-free Lanczos is visually indistinguishable from the spectrum obtained by explicit float64 eigendecomposition. **(b)** Per-mode relative eigenvalue error. The aggregate relative eigenvalue error is 2.6×10^{-16}, and the relative error between the exact and Lanczos eigenspace projectors is 2.4×10^{-8}. The minimum principal cosine is numerically one. Q-DEIM applied to the complete exact and Lanczos eigenspaces selects 99.75% of the same physical coordinates. These results validate the matrix-free approximation against the explicitly formed Hessian over the entire network.

approximation used by the scalable method against dense eigendecomposition of the entire network Hessian.

The numerical agreement also preserves the quantities used by the proposed method. We select $k = 128$ modes using either leading curvature or the task-aligned score, apply Q-DEIM, and fine-tune the resulting anchor coordinates for ten epochs. Exact and Lanczos constructions use the same checkpoint, calibration data, anchor budget, minibatch orders, and five paired seeds. For both selec-

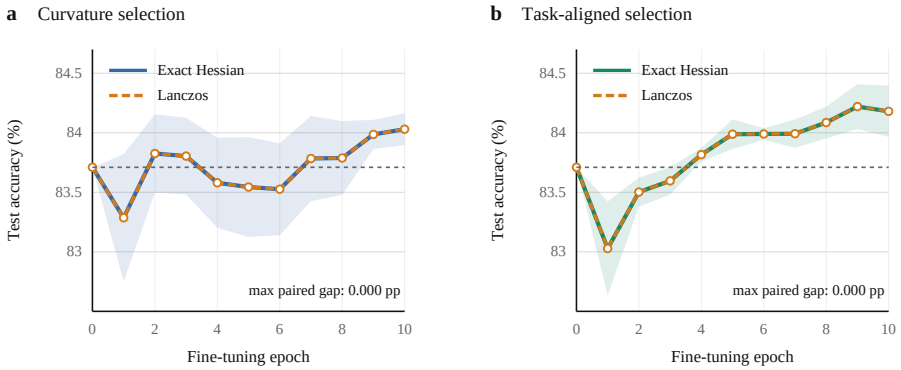


Figure 8: **Lanczos preserves downstream behavior for both mode selection rules.** Mean test accuracy is shown over five paired minibatch-order seeds; shaded regions denote the standard error of the mean. Solid curves use modes obtained from explicit eigendecomposition of the full Hessian, while dashed curves use modes obtained from matrix-free Lanczos iteration. **(a)** Leading-curvature selection ranks candidate modes by eigenvalue magnitude. **(b)** Task-aligned selection ranks modes by predicted local loss reduction. Exact and Lanczos trajectories coincide for every epoch and paired seed, giving a maximum paired accuracy difference of 0.000 percentage points at the reported precision. The final mean accuracies are 84.03% for curvature selection and 84.18% for task-aligned selection under both eigensolvers. Thus, Lanczos reproduces both the selected coordinates and the optimization behavior of the exact full-Hessian construction.

tion rules, the selected subspaces have minimum principal cosine greater than 0.99999999999995, projector errors below 1.5×10^{-7}, and identical Q-DEIM anchor sets.

C.2 Learning-rate and checkpoint selection

Each method, adaptation budget, and seed selects its learning rate independently from

$$\{3 \times 10^{-4},\ 10^{-3},\ 3 \times 10^{-3},\ 10^{-2}\}$$

using development accuracy. The anchor-budget experiments also select the checkpoint with the highest development accuracy within the 15-epoch training window. Test labels are never used for hyperparameter or checkpoint selection; the selected configuration is evaluated on the test set only after selection. The chosen learning rate, best epoch, and resulting development and test accuracies are recorded in the accompanying per-run CSV files.

D Reproducibility and Hyperparameters

This section reports the complete configurations used for the controlled classification-head experiments, the parameter-efficient adaptation comparison, the Hessian-rank scaling experiments, and the exact–Lanczos validation. Hyperparameters were held fixed across methods within each experiment family. Learning rates were selected using development accuracy, and test labels were not used for model or hyperparameter selection.

D.1 Controlled classification-head adaptation

We used ImageNet-pretrained ResNet-18 and ViT-B/16 feature extractors from `torchvision`. The pretrained backbones were frozen throughout the experiment. CIFAR-10 images were resized to 224×224 and normalized using the ImageNet channel statistics

$$\mu = (0.485, 0.456, 0.406), \qquad \sigma = (0.229, 0.224, 0.225).$$

Backbone features were computed once and cached. The resulting feature dimensions were 512 for ResNet-18 and 768 for ViT-B/16. We adapted a bias-free, ten-class linear head containing 5,120 or 7,680 parameters, respectively.

Table 3: **Configuration of the controlled ResNet-18 and ViT-B/16 classification-head experiments.**

Hyperparameter	Value
Dataset	CIFAR-10
Pretrained weights	`torchvision` default ImageNet weights
Input resolution	224×224
Frozen feature dimensions	512 (ResNet), 768 (ViT)
Adapted layer	Bias-free linear classification head
Head dimensions	10×512, 10×768
Training subset	10,000 images
Development fraction	0.15
Effective train/dev sizes	8,500/1,500 images
Test subset	5,000 images
Training-subset seed	1234
Test-subset seed	5678
Experimental seeds	$\{0, 1, 2, 3, 4\}$
Batch size	256
Epoch budget	30
Optimizer	AdamW
Weight decay	0
Learning-rate grid	$\{3 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}, 10^{-2}\}$
Loss	Multiclass cross-entropy
Adaptation dimensions	$k \in \{5, 20, 50, 100, 150, 250\}$
LoRA ranks	$r \in \{1, 2, 4, 8\}$
Number of workers	8
ResNet feature batch size	512
ViT feature batch size	256

For every seed, the initial head weights were sampled independently from

$$\theta_{ij} \sim \mathcal{U}\left(-d^{-1/2}, d^{-1/2}\right),$$

where d is the backbone feature dimension. All methods within a seed used the same initialization, data split, minibatch ordering, epoch budget, and learning-rate candidates.

Each learning-rate candidate was trained for the complete 30-epoch budget. The checkpoint with the highest development accuracy was retained for that learning rate. We then selected the learning rate with the highest development accuracy and evaluated its retained checkpoint once on the test set. Ties were resolved by the order of the learning-rate grid. Minibatch orders were regenerated from the same seed for every method and learning-rate candidate.

D.2 Curvature and subspace construction

The curvature operator was constructed from the cross-entropy loss of the linear classification head. For each experimental seed, the calibration set contained the first 4,096 examples of that seed's training split. Hessian products were evaluated by automatic differentiation without explicitly materializing the head Hessian.

The largest algebraic eigenpairs were computed using the symmetric implicitly restarted eigensolver in `scipy.sparse.linalg.eigsh`. Only eigenvalues satisfying

$$\lambda_i > \max\left(10^{-10}, 10^{-8}\lambda_{\max}\right)$$

were retained. The candidate pool contained at most 256 positive-curvature modes.

Given the extracted eigenvectors $V = [v_1, \ldots, v_p]$, the methods selected their k-dimensional spaces as follows:

$$\begin{aligned} \text{Curvature:} &\quad \text{largest } \lambda_i, \\ \text{Gradient:} &\quad \text{largest } (v_i^\top g)^2, \\ \text{Task-aligned:} &\quad \text{largest } \frac{(v_i^\top g)^2}{\lambda_i + \gamma}. \end{aligned}$$

Table 4: **Curvature extraction and coordinate-selection hyperparameters for the controlled head experiments.**

Hyperparameter	Value
Calibration examples	4,096
Candidate eigenpairs	256
Eigensolver	Symmetric implicitly restarted Lanczos (`eigsh`)
Eigensolver target	Largest algebraic eigenvalues (`which=LA`)
Eigensolver tolerance	10^{-4}
Positive-mode threshold	$\max(10^{-10}, 10^{-8}\lambda_{\max})$
Hutchinson probes	64 Rademacher vectors
Task-alignment damping γ	10^{-3}
Q-DEIM factorization	Column-pivoted QR of $V^\top$
Stabilized pseudoinverse cutoff	10^{-2}
Stabilized map clipping	$[-0.5, 0.5]$
Random-space distribution	Gaussian followed by thin QR

The dimension-matched random control was a Gaussian random basis orthonormalized by QR. Random physical coordinates were sampled uniformly without replacement.

For a selected basis $V_k \in \mathbb{R}^{N\times k}$, Q-DEIM applied column-pivoted QR to $V_k^\top$. If P denotes the selected coordinate operator, the exact reconstruction basis was

$$B = V_k(P^\top V_k)^{-1}, \qquad P^\top B = I_k.$$

Optimization began from zero reduced coordinates, so every constrained method started from the same initial head.

D.3 Parameter-efficient budget comparison

The accuracy-versus-anchor-budget experiment used the complete CIFAR-10 training and test sets. It therefore has a different data and training configuration from Table 3. The comparison included task-aligned Q-DEIM, random physical coordinates, LoRA, QLoRA, DoRA, IA^3, an AdaLoRA-style scheduled-rank control, and dense head tuning.

Development accuracy was measured after every epoch. Training stopped after four consecutive epochs without an improvement. For each learning rate, the checkpoint with the highest development accuracy was retained. The learning-rate winner was selected by development accuracy and evaluated once on the test set.

The adapter implementations used the following parameterizations:

$$\Delta W_{\mathrm{LoRA}} = \frac{1}{r}BA,$$

where $A \in \mathbb{R}^{r\times d}$ was initialized with Kaiming-uniform entries and $B \in \mathbb{R}^{C\times r}$ was initialized to zero. QLoRA used the same trainable factors and a symmetric four-bit quantized frozen base head,

$$s = \frac{\max_{ij}|W_{ij}|}{7}, \qquad W_q = s\ \mathrm{clip}\left(\mathrm{round}(W/s), -8, 7\right).$$

DoRA additionally trained one magnitude value per output class and normalized each updated weight row before applying that magnitude. IA^3 trained one multiplicative gate per output class. The AdaLoRA-style head control began with

$$r_{\max} = \min\{10, \max(r, 2r)\}$$

and linearly reduced the active rank toward the target rank r. Active components were selected using the learned magnitude variables. Because canonical QLoRA and AdaLoRA normally act across multiple transformer layers, their single-head versions are treated as matched controls in this experiment.

Table 5: **Configuration of the anchor-budget and parameter-efficient adaptation comparison.** Factorized methods are plotted using the exact number of variables maintained by the optimizer, rather than nominal matrix rank.

Hyperparameter	Value
Dataset	CIFAR-10
Training images	50,000
Development fraction	0.10
Effective train/dev sizes	45,000/5,000
Test images	10,000
Experimental seeds	$\{0, 1, 2, 3, 4\}$
Epoch budget	15
Early-stopping patience	4 epochs
Training batch size	128
ResNet feature batch size	128
ViT feature batch size	32
Optimizer	AdamW
Weight decay	0
Learning-rate grid	$\{10^{-3}, 3 \times 10^{-3}, 10^{-2}\}$
Anchor counts	$\{5, 20, 50, 100, 150, 250\}$
Adapter ranks	$\{1, 2, 4, 6, 8, 10\}$
Calibration examples	1,024
Candidate Hessian modes	300
Task damping γ	10^{-3}
Q-DEIM pseudoinverse cutoff	10^{-2}
Reconstruction-map clipping	$[-0.5, 0.5]$
Number of workers	8

D.4 Weight and update distributions

The weight-distribution figures reuse checkpoints produced by the controlled head-adaptation protocol. No additional models were trained specifically for the density plots. For every method, we recorded the matched initial head θ_0, final head θ_{final}, and signed update

$$\Delta\theta = \theta_{\text{final}} - \theta_0.$$

The three plotted quantities are θ_{final}, $\Delta\theta$, and

$$\log_{10}\left(|\Delta\theta| + 10^{-12}\right).$$

For multi-seed figures, density curves were first computed separately for each seed and then summarized by their pointwise mean and standard error.

D.5 Exact Hessian-rank saturation

The rank-saturation experiment was designed to separate ambient model size from task size. MNIST images were resized to 8×8, normalized using mean 0.1307 and standard deviation 0.3081, and flattened to 64 dimensions. A fixed random projection followed by a hyperbolic tangent defined the features:

$$\phi_w(x) = \tanh(P_w x), \qquad (P_w)_{ij} \sim \mathcal{N}(0, 1/64).$$

All widths used nested rows of the same maximum-width projection matrix. Only the final linear classification head was trained.

The Hessian was symmetrized as $(H + H^\top)/2$ before applying `torch.linalg.eigvalsh`. Rank was computed from the entire eigenspectrum, rather than from a fixed number of leading modes. For a softmax classifier with m calibration examples and C classes, the structural ceiling is $m(C-1)$; this quantity was reported alongside the measured rank.

D.6 GPT MLP scaling experiment

The GPT scaling experiment used randomly initialized GPT-2-style language models and the WikiText-2 training split. Text was tokenized with the GPT-2 tokenizer, truncated or padded to

Table 6: **Configuration of the exact Hessian-rank saturation experiment.** The complete Hessian spectrum was computed in float64 for every task and feature width.

Hyperparameter	Value
Dataset	MNIST training split
Task sizes	2, 5, and 10 classes
Included labels	First C labels, $\{0, \dots, C-1\}$
Input size	8×8
Feature map	Fixed Gaussian projection followed by $\tanh$
Feature widths	$\{8, 16, 32, 64, 96, 128, 192, 256\}$
Head parameter count	$N = Cw$
Training examples	4,800, balanced by class
Calibration examples	120, balanced by class
Training/calibration overlap	None
Head initialization	Zero
Optimizer	AdamW
Learning rate	3×10^{-3}
Weight decay	10^{-2}
Epochs	25
Batch size	256
Hessian precision	Float64
Hessian row batch size	16
Rank cutoff	$\lvert\lambda_i\rvert > 10^{-8} \max_j \lvert\lambda_j\rvert$
Random seed	0
Projection seed	913

Table 7: **Configuration of the GPT MLP spectral-dimension experiment.** Reported dimensions are defined relative to the extracted positive Ritz spectrum.

Hyperparameter	Nano-GPT	Micro-GPT	Small-GPT
Embedding dimension	128	512	768
Transformer layers	2	6	12
Attention heads	2	8	12
Context length	64	64	64
MLP matrix parameters	66,048	1,050,624	2,362,368
Warm-up steps	50	50	50
Warm-up batch size	16	16	16
Calibration batch size	8	8	8
Optimizer	AdamW	AdamW	AdamW
Learning rate	5×10^{-4}	5×10^{-4}	5×10^{-4}
Weight decay	10^{-2}	10^{-2}	10^{-2}
Maximum Ritz values	1,000	1,000	1,000
Energy thresholds		$90\%, 95\%, 99\%$	

64 tokens, and empty sequences were removed. The Hessian operator was formed for the weights and bias of the first transformer block's MLP input projection.

Hessian–vector products were evaluated in float32. The largest algebraic Ritz values were extracted with `eigsh`, and negative Ritz values were removed. For threshold ρ, the reported dimension was

$$k_\rho = \min\left\{ k : \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{p} \lambda_i} \geq \rho \right\},$$

where $p \leq 1000$ is the number of extracted positive Ritz values. Thus the denominator in this experiment is the extracted positive spectrum and not a complete full-model Hessian trace. The same definition was applied to all three model sizes.

Table 8: **Configuration of the exact–Lanczos whole-network Hessian validation.**

Hyperparameter	Value
Architecture	196–8–8–10 tanh MLP
Total parameters	1,738
Training examples	10,000
Calibration examples	80; 8 per class
Training/calibration overlap	None
Reference-training optimizer	AdamW
Reference-training learning rate	2×10^{-3}
Reference-training weight decay	10^{-4}
Reference-training epochs	10
Reference-training batch size	256
Exact Hessian precision	Float64
Exact eigensolver	`torch.linalg.eigh`
Relative rank tolerance	10^{-9}
Absolute rank tolerance	10^{-11}
Resolved numerical rank	792
Lanczos oversampling	48 iterations
Completed Lanczos iterations	794
Reorthogonalization	Two complete passes
Lanczos stopping tolerance	$\beta < 10^{-13}$
Selection budget	$k = 128$
Relative damping	10^{-2}
Random seed	2027
TF32 matrix multiplication	Disabled

D.7 Exact–Lanczos validation

We validated matrix-free Lanczos against the complete eigendecomposition of an explicitly materialized whole-network Hessian. The network was a 196–8–8–10 MLP with hyperbolic-tangent activations and 1,738 parameters, including all weights and biases. MNIST images were resized to 14×14 and normalized using mean 0.1307 and standard deviation 0.3081.

The numerical rank threshold was

$$\tau = \max\left(10^{-11}, 10^{-9} \max_i |\lambda_i|\right).$$

Lanczos used a random normalized starting vector and two-pass complete reorthogonalization. Because the Hessian was indefinite, damping was selected as

$$\gamma = \max\left(10^{-2} \max_i |\lambda_i|, -\lambda_{\min} + 10^{-2} \max_i |\lambda_i|\right).$$

Exact and Lanczos modes were passed independently through the same task-mode selection and Q-DEIM procedures.

For the downstream four-way validation, exact-curvature, Lanczos-curvature, exact-task, and Lanczos-task mappings all started from the same saved reference network. Fine-tuning used 5,000 MNIST examples disjoint from the reference training and curvature-calibration samples.

D.8 Randomness and reporting

Python, NumPy, PyTorch CPU, and PyTorch CUDA random-number generators were seeded for the controlled experiments. Deterministic cuDNN execution was enabled and cuDNN benchmarking was disabled. Each controlled result is reported as the arithmetic mean and sample standard deviation over five seeds:

$$\bar{x} = \frac{1}{5}\sum_{s=1}^{5} x_s, \qquad \mathrm{SD}(x) = \sqrt{\frac{1}{4}\sum_{s=1}^{5}(x_s - \bar{x})^2}.$$

All hyperparameter decisions were based on development accuracy. The test set was evaluated only after selecting the checkpoint and learning rate.

Table 9: **Downstream configuration for the four-way exact–Lanczos validation.** Minibatch orders were paired across all four methods within each seed.

Hyperparameter	Value
Adaptation examples	5,000
Evaluation set	Complete MNIST test set
Selected dimension	128
Optimized variables	128 Q-DEIM anchors
Optimizer	Adam
Learning rate	3×10^{-3}
Epochs	10
Batch size	256
Arithmetic precision	Float64
Seeds	$\{0, 1, 2, 3, 4\}$

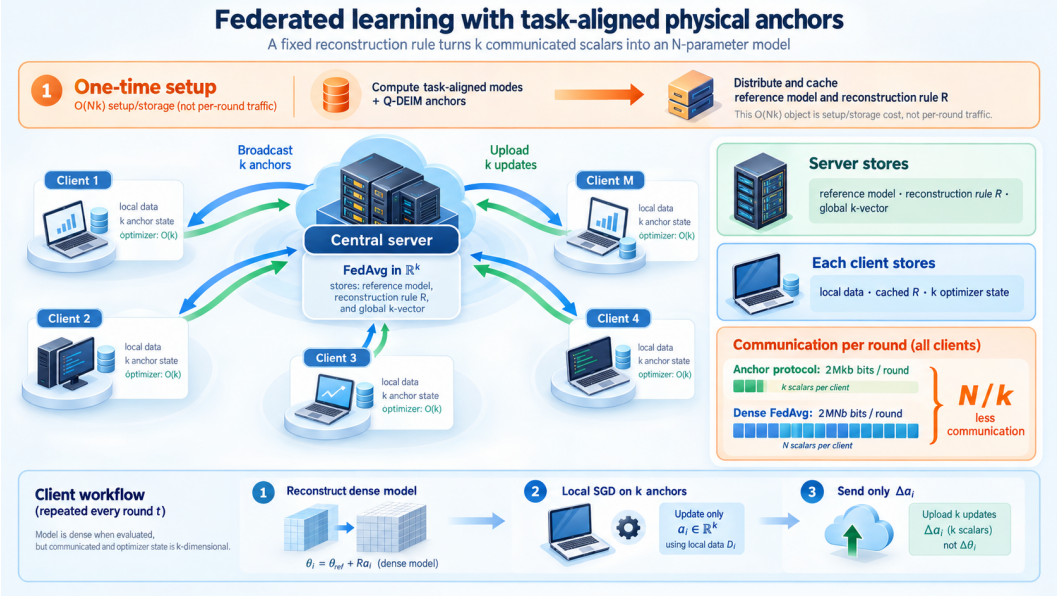


Figure 9: **Federated fine-tuning in anchor space.** Clients reconstruct the dense model, optimize k shared physical anchors, and communicate only $\Delta a_i \in \mathbb{R}^k$. The server aggregates updates directly in anchor space, reducing per-round communication by approximately N/k.

E Federated Fine-Tuning

Physical anchors provide a communication-efficient parameterization for federated fine-tuning. After receiving a shared reference model and reconstruction map R, each client optimizes and transmits only k anchors while reconstructing the dense model locally (Figure 9). For M clients and b bits per scalar, this reduces per-round bidirectional communication from approximately $2MNb$ to $2Mkb$ bits. The saving comes with additional local reconstruction cost, as summarized in Table 10.

Future work can evaluate anchor-space aggregation against FedAvg and federated PEFT methods under IID and non-IID client partitions, measuring accuracy against total communication, local computation, and memory.

Table 10: **Communication–computation tradeoff.**

Setting (N, k)	Payload reduction	FLOPs/dense	Map storage
MNIST MLP $(203{,}530, 300)$	99.85%	$\sim 50\times$	232.6 MB
CIFAR CNN $(1.07\text{M}, 463)$	99.96%	$\sim 77\times$	1.89 GB