LEARNING FUNCTIONAL SUBSPACES FOR NEURAL NETWORK COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern transformers couple impressive capabilities with substantial memory and compute demands. Low-rank weight factorization reduces both while keeping the matrices dense, and therefore efficient on standard hardware. However, several existing methods choose the subspace removed from each weight matrix by *local* closed-form criteria—activation energy, layer-wise reconstruction error, or a quadratic loss approximation—and thus ignore how errors propagate through the rest of the network. At aggressive compression ratios, these errors compound through depth and performance collapses. The number of directions removed from each layer is set before training according to that layer's measured effect on the output. We introduce *Learnable Subspace Projections* (LSP), which instead learns the removed subspaces end-to-end. Each linear layer, or *tied group* of layers that read the same activations, receives an orthogonal projector. All projectors are optimized jointly against a *global* objective, the KL divergence to the dense model's output distribution, while the pretrained weights stay frozen. Projectors are initialized from a whitened SVD truncation, and ranks are allocated by measured cost: the output KL divergence each projector induces, per parameter saved. After training, the projectors merge into standard low-rank factors, and each tied group shares one input factor, so attention can cache a single narrow latent instead of full keys and values. Across decoder-only LLMs (OPT-125M/1.3B, Qwen3-4B, Llama-2-7B) and a Vision Transformer (ViT-B/16), LSP outperforms activation-based, curvature-based, and rank-learning baselines by a margin that grows with the compression ratio. At 70% compression, LSP brings Llama-2-7B to 10.9 WikiText-2 perplexity versus 13.3 for the strongest baseline, and to 42.2% mean zero-shot accuracy versus 36.0%. The factorized model decodes up to $1.6\times$ faster than the dense model at small batch sizes, and caching the shared latent reduces the combined memory of weights and KV cache by $13.5\times$ at a 128k-token context, versus at most $6.5\times$ for untied factorizations.

1 INTRODUCTION

Transformers (Vaswani et al., 2017) have become the dominant architecture in deep learning, owing to their flexibility, scalability, and strong performance across tasks and modalities. Their uniform design enables rapid development and stable scaling; however, it also leaves trained models highly redundant. Large models exhibit low intrinsic dimensionality (Aghajanyan et al., 2021; Kuzborskij & Yadkori, 2025), with their capabilities concentrated in a low-dimensional subspace of their parameters. Compression exploits this redundancy to reduce memory and compute while preserving the model's behavior.

Low-rank compression is a natural way to realize these savings, replacing each weight matrix with the product of two smaller factors. Yet this redundancy is not directly visible in the weights' spectra. The singular values of pretrained Transformer weights decay slowly, so naive truncated SVD severely degrades performance (Hsu et al., 2022). Practical low-rank compression instead builds on an empirical observation: the *activations* flowing between layers lie in low-dimensional subspaces (Yu & Wu, 2023; Yuan et al., 2024; Garg et al., 2024; Skean et al., 2025). Existing methods read the useful subspace off activation statistics, either from the energy of the layer input, as in ASVD (Yuan et al., 2024), SliceGPT (Ashkboos et al., 2024), and MoDeGPT (Lin et al., 2025), or from the reconstruction error of the layer output, as in SVD-LLM (Wang et al., 2025d) and Swift-SVD (Qi

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.

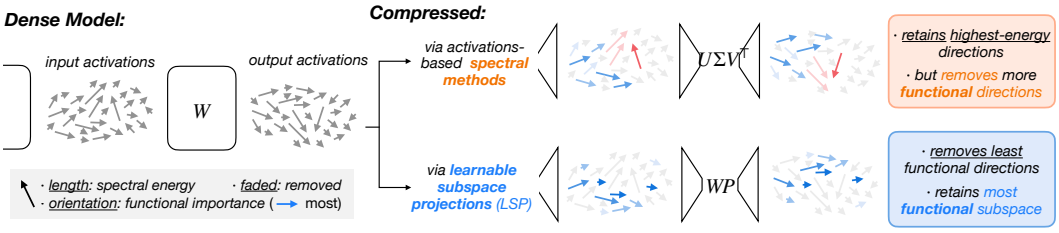


Figure 1: **Spectral energy is not function.** Each arrow is a direction in the input activation space of a linear layer $\boldsymbol{W}$: its length is the spectral energy it carries, its orientation how much the network output depends on it (horizontal right: most), and faded arrows are removed. Activation-based low-rank compression (top) keeps the highest-energy directions, even when the output barely depends on them (red), and can discard functional ones (blue). LSP (bottom) learns the projector $\boldsymbol{P}$ against the network output, removing the directions the output is least sensitive to, whatever their energy.

et al., 2026). Either criterion captures the layer, not the network: a low-energy direction is discarded whether or not the network's output depends on it, and a layer-wise optimum says nothing about how its residual error propagates downstream. Loss-aware methods bring the network in through a local surrogate—Fisher-weighted reconstruction in FW-SVD (Hsu et al., 2022) or a quadratic curvature model in LLM-Surgeon (van der Ouderaa et al., 2024)—but such surrogates hold only for small perturbations of the dense weights, whereas aggressive compression moves far from them. Dobi-SVD (Wang et al., 2025b) optimizes the loss directly, but learns only the per-matrix truncation rank. The loss thus shapes local approximations, sets ranks, or repairs the weights after truncation; in none of these methods does the network loss itself choose which directions are removed. What compression has to identify, then, is the network's *functional subspace*: the directions whose retention preserves its behavior. Our starting observation is that choosing a subspace amounts to choosing an orthogonal projector: applying a rank-r projector to a weight matrix leaves it with rank at most r, so it factors into two thin matrices. The search for the functional subspace can thus be cast as a search over projectors. This raises our central question: *can we obtain better compressed models by learning which subspace to remove, optimizing the projectors end-to-end against the frozen network's output?*

We propose Learnable Subspace Projections (LSP), which does exactly this: every projector is learned against a global objective, by default the KL divergence to the dense model's output distribution, or alternatively the model's original training objective, a variant we denote LSP^T. Our contributions are: (1) **Learned subspace projections.** Layers that read the same activations (Q/K/V, gate/up) form a *tied group* and share one projector; we optimize all projectors jointly across the network with the pretrained weights frozen. An activation-space parameterization with fused QR orthonormalization keeps this tractable at billion-parameter scale. (2) **Whitened initialization and measured-KL allocation.** Each projector is initialized from a whitened truncation, recast as an orthogonal projector and extended to tied groups. To decide how many directions each layer or tied group removes, we apply each projector in isolation, measure the KL divergence it induces on the model output, and allocate the budget by KL cost per saved parameter. The model at initialization, which we call NoLSP, serves as a control that isolates the effect of learning. (3) **Efficient low-rank inference.** After training, the projectors merge into plain low-rank factors, so the deployed model contains no LSP-specific operations. At the same compression ratio it requires the same FLOPs per token as other factorizations, yet the input factor shared by each tied group speeds up small-batch decoding and, in a latent-cache deployment, stores one narrow latent per group where untied factorizations store two (one each for keys and values), shrinking the KV cache. Because LSP applies to any linear layer, the same recipe transfers across Transformer models without architecture-specific surgery. LSP outperforms pruning and SVD baselines by a margin that grows with the compression ratio, and on ViT it degrades least among the compared methods when calibration and evaluation distributions differ.

2 Learnable Subspace Projections (LSP)

LSP compresses a pretrained network into its functional subspace: each targeted linear layer, or tied group of layers that read the same activations, is composed with a learned orthogonal projector, and all projectors are optimized end-to-end against the model output with the pretrained weights frozen. Section 2.1 gives the formulation, Section 2.2 the whitened initialization and measured-KL allocation, and Section 2.3 the training procedure and the merge into low-rank factors.

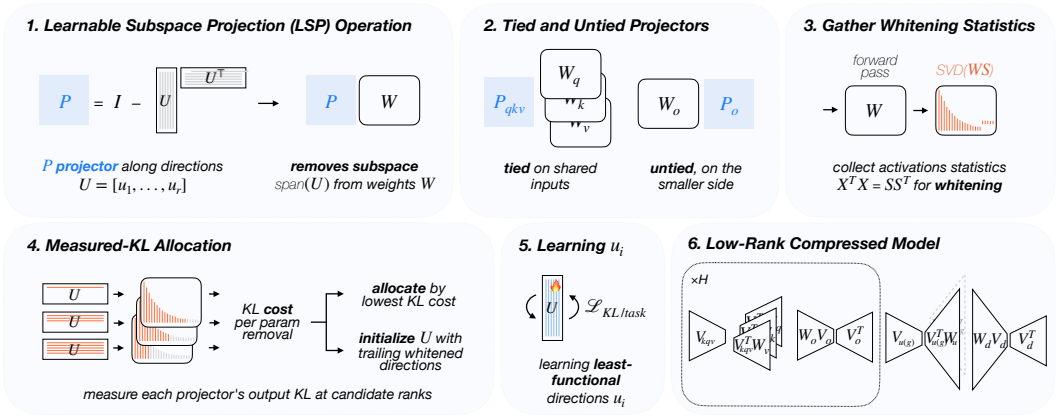


Figure 2: **Overview of LSP.** (1) An orthogonal projector $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{U}\boldsymbol{U}^\top$ removes the learned subspace $\mathrm{span}(\boldsymbol{U})$ from the input or output of $\boldsymbol{W}$ (Section 2.1). (2) Layers that read the same activations (Q/K/V, gate/up) share a tied projector; the remaining layers are projected on their smaller side. (3) Whitened SVD ranks candidate directions. (4) The output KL each projector induces, per saved parameter, allocates how many directions each layer or tied group removes, and the trailing whitened directions initialize $\boldsymbol{U}$ (Section 2.2). (5) All projectors are trained end-to-end against the model output, with $\boldsymbol{W}$ frozen (Section 2.3). (6) The trained projectors decompose into low-rank factors $\boldsymbol{V}\boldsymbol{V}^\top$, with one shared $\boldsymbol{V}$ per tied group (Equation (6)).

2.1 Learning Subspace Projections

Projector parameterization. Consider a linear layer $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ with $\boldsymbol{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$; the bias is left unchanged throughout. To remove k input directions, we learn $\boldsymbol{U} \in \mathbb{R}^{d_{\text{in}} \times k}$ with orthonormal columns and apply the orthogonal projector $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{U}\boldsymbol{U}^\top$ to the weight:

$$\boldsymbol{W}\boldsymbol{P} \;=\; \boldsymbol{W} - (\boldsymbol{W}\boldsymbol{U})\boldsymbol{U}^\top, \tag{1}$$

which, as a result, has rank at most $d_{\text{in}} - k$. Orthonormality holds by construction: we optimize an unconstrained $\boldsymbol{V} \in \mathbb{R}^{d_{\text{in}} \times k}$ and set $\boldsymbol{U} = \mathrm{qf}(\boldsymbol{V})$, the orthogonal factor of its thin QR decomposition, on every forward pass. Since $\boldsymbol{P}$ is invariant under $\boldsymbol{U} \mapsto \boldsymbol{U}\boldsymbol{O}$ for any orthogonal $\boldsymbol{O}$, the effective variable is the removed subspace $\mathrm{span}(\boldsymbol{U})$, which equals $\mathrm{span}(\boldsymbol{V})$ whenever $\boldsymbol{V}$ has full column rank. The output-side form is $\boldsymbol{P}\boldsymbol{W}$, with $\boldsymbol{U} \in \mathbb{R}^{d_{\text{out}} \times k}$ and rank at most $d_{\text{out}} - k$.

Tied groups and projection side. Q/K/V read the same normalized activation, as do gate/up; each such *tied group* shares a single *tied projector* on this common input. Every other layer is projected on its smaller side, where, for a full-rank weight, each removed direction lowers the rank by one and $\boldsymbol{U}$ is smallest. Sharing lets the group retain a higher rank at the same parameter budget and, in attention, cache a single latent instead of full keys and values (Section 2.3).

Practical design choices. Three choices make training scalable and stable; details are in Section A.2. *(i) Activation-space form.* We compute $\boldsymbol{W}(\boldsymbol{x} - \boldsymbol{U}(\boldsymbol{U}^\top\boldsymbol{x}))$ without forming $\boldsymbol{W}\boldsymbol{P}$, which would materialize a dense $d_{\text{out}} \times d_{\text{in}}$ matrix and its gradient per layer, and a fused QR backward stores $\mathcal{O}(d_{\text{in}}k)$ per layer, versus $\mathcal{O}(d_{\text{in}}k^2)$ for an explicit Gram–Schmidt loop. *(ii) Warm-up and direction dropout.* Training uses $\boldsymbol{P}_{\alpha,\boldsymbol{m}} = \boldsymbol{I} - \alpha\,\boldsymbol{U}\,\mathrm{diag}(\boldsymbol{m})\boldsymbol{U}^\top$: α ramps from 0 to 1 over the first epoch to avoid abrupt output changes, and a random mask $\boldsymbol{m} \in \{0,1\}^k$ temporarily keeps some removed directions, acting as dropout. *(iii) Orthogonality penalty.* A penalty $\mathcal{L}_{\text{ort}}$ on correlations between the columns of $\boldsymbol{V}$ (Equation (9)) stabilizes QR and its gradient.

2.2 Whitened Initialization and Measured-KL Allocation

Before training, each compression unit—a tied group or an individual layer—needs a subspace to start from and a number of directions to remove. We first construct an ordered basis from SVD-LLM's truncation-aware whitening (Wang et al., 2025d), adapted to orthogonal projectors and tied groups, so that removing k directions amounts to dropping the trailing k basis vectors. We then measure the output KL induced by candidate truncations to allocate ranks under the global parameter budget: the basis is spectral, while the allocation is loss-aware.

Whitened initialization. SVD-LLM scores directions by their effect on the layer output rather than by their input energy, but it truncates each matrix separately and, on the input side, through an oblique projector. Dropping the layer index, let $\boldsymbol{X} \in \mathbb{R}^{n \times d_{\text{in}}}$ hold n calibration inputs, with positive-definite Gram matrix $\boldsymbol{G} = \frac{1}{n}\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{S}\boldsymbol{S}^\top$ ($\boldsymbol{S}$ is its Cholesky factor), so that a candidate weight $\boldsymbol{Z}$ has reconstruction error $\mathcal{E}(\boldsymbol{Z}) = \frac{1}{n}\|(\boldsymbol{W} - \boldsymbol{Z})\boldsymbol{X}^\top\|_F^2 = \|(\boldsymbol{W} - \boldsymbol{Z})\boldsymbol{S}\|_F^2$. SVD-LLM minimizes it at rank r by truncating $\boldsymbol{W}\boldsymbol{S} = \boldsymbol{U}^w\boldsymbol{\Sigma}^w(\boldsymbol{V}^w)^\top$ to its leading r directions ($k = d - r$ removed from a side of dimension d; "rest" marks the trailing ones):

$$\widehat{\boldsymbol{W}} \;=\; \boldsymbol{U}_r^w(\boldsymbol{U}_r^w)^\top \boldsymbol{W} \;=\; \boldsymbol{W}\,\boldsymbol{S}\boldsymbol{V}_r^w(\boldsymbol{V}_r^w)^\top \boldsymbol{S}^{-1}, \qquad \mathcal{E}(\widehat{\boldsymbol{W}}) = \textstyle\sum_{i>r}(\sigma_i^w)^2. \tag{2}$$

The first form is already an orthogonal projector on the output side; on the input side, we replace the oblique second form, which keeps $\operatorname{span}(\boldsymbol{S}\boldsymbol{V}_r^w)$ and zeroes $\operatorname{span}(\boldsymbol{S}\boldsymbol{V}_{\text{rest}}^w)$, by the orthogonal projector that keeps the same subspace.

Proposition 1 (Orthogonal recast)**.** *Let $\boldsymbol{A} = \boldsymbol{S}\boldsymbol{V}_r^w$, $\boldsymbol{B} = \boldsymbol{S}\boldsymbol{V}_{rest}^w$, and $\boldsymbol{P}_A = \boldsymbol{A}\boldsymbol{A}^+$, the orthogonal projector that removes* $\operatorname{span}(\boldsymbol{S}^{-\top}\boldsymbol{V}_{rest}^w)$. (i) *On the output side, $\boldsymbol{I} - \boldsymbol{U}_{rest}^w(\boldsymbol{U}_{rest}^w)^\top$ attains the optimum $\sum_{i>r}(\sigma_i^w)^2$.* (ii) *On the input side, $\mathcal{E}(\boldsymbol{W}\boldsymbol{P}_A) = \sum_{i>r}(\sigma_i^w)^2 + \|\boldsymbol{\Sigma}_r^w\boldsymbol{A}^+\boldsymbol{B}\|_F^2$; if $\sigma_r^w > 0$, the excess term vanishes if and only if $\boldsymbol{A}^\top\boldsymbol{B} = 0$.* (iii) *For a tied group, the summed error under one shared kept subspace equals that of the row-stacked (input-tied) or column-stacked (output-tied) whitened weight; its leading right or left singular subspace, respectively, minimizes this error, and (ii) or (i) applies to the stacked weight.*

The proof is in Section A.3. The initialized layer thus agrees with SVD-LLM on the kept subspace, while on the directions $\boldsymbol{B}$, which SVD-LLM zeroes, it retains their least-squares component along $\boldsymbol{A}$; the excess is that component weighted by the kept singular values. It vanishes when SVD-LLM's projector is itself orthogonal, as for an isotropic input Gram matrix.

Measured-KL rank allocation. Units differ in how much the model output depends on them, so we measure each unit's sensitivity directly and remove parameters where the measured cost per saved parameter is lowest; sensitive units keep more rank or stay dense. For a unit u, removing the trailing k directions of its initialized basis, with the rest of the network dense, costs

$$\Delta_u(k) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{\text{probe}}} D_{\text{KL}}\big(p_{\theta_0}(\cdot \mid \boldsymbol{x}) \,\big\|\, p_{\theta_0\setminus(u,k)}(\cdot \mid \boldsymbol{x})\big)\,, \tag{3}$$

where $\mathcal{D}_{\text{probe}}$ is a calibration subset and $\theta_0 \setminus (u,k)$ denotes the dense model with only this truncation applied. The probe needs no gradients and no new SVDs: the dense model's outputs are cached once, and every candidate reuses the unit's ordered basis. We probe a grid of removal fractions and interpolate the running maximum of each curve, so that the resulting estimate $\widetilde{\Delta}_u$ is nondecreasing in k. Starting from the dense model, the allocator repeatedly takes the move $k \to k'$ with the smallest marginal cost $\big(\widetilde{\Delta}_u(k') - \widetilde{\Delta}_u(k)\big)/\big(s_u(k') - s_u(k)\big)$, where $s_u(k)$ is the number of parameters the unit saves after factorization, counting shared factors once, until the target savings are reached (Section A.4). Unlike a quadratic model around the dense weights, the probe evaluates finite truncations through the full network, but it treats units as independent: the accumulated KL of the jointly compressed model is addressed by joint training at fixed ranks.

2.3 Training Procedure

We replace each targeted linear layer with its LSP counterpart and initialize it as in Section 2.2: we construct the whitened bases, allocate ranks by measured KL, and set $\{\boldsymbol{V}_\ell\}$ accordingly. We then optimize all $\{\boldsymbol{V}_\ell\}$ jointly at fixed ranks, with every pretrained parameter frozen, against

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{obj}}\big(f\big(\cdot\,; \{\boldsymbol{P}_{\alpha,\boldsymbol{m}}^{(\ell)}\}\big)\big) + \lambda_{\text{ort}}\,\mathcal{L}_{\text{ort}}. \tag{4}$$

By default, $\mathcal{L}_{\text{obj}}$ is the output-distillation loss (Hinton et al., 2015)

$$\mathcal{L}_{\text{KL}} = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_{\text{cal}}}\,\text{KL}\big(p_{\text{dense}}(\cdot \mid \boldsymbol{x}) \,\big\|\, p_{\text{compressed}}(\cdot \mid \boldsymbol{x})\big), \tag{5}$$

where p is the distribution over the next token for LLMs, averaged over all positions, and over classes for classifiers. Alternatively, $\mathcal{L}_{\text{obj}}$ is the model's original training loss (next-token or classification cross-entropy), a variant we denote LSP ^{T}. The teacher is the same network with its projectors disabled ($\alpha = 0$), run without gradient tracking, so no second copy of the weights is stored. We use a linear-warmup cosine learning-rate schedule with early stopping on a held-out validation split, and keep the best validation epoch (Section B).

Choice of objective. Distillation is the default: it targets the dense model's full output distribution rather than a single target, and for classifiers it needs no labels, so any unlabeled pool can serve for calibration (Section 3.2). The task loss (LSP ^{T}) instead fits the calibration data directly, so it can specialize the model when the calibration data matches the deployment task; in our experiments, LSP ^{T} reaches its best validation score in fewer epochs on average (Section B).

Merging. After training, each projector is folded into its weight, a step we call *merging*. For an input-side projector, let the columns of $\boldsymbol{U}_\perp \in \mathbb{R}^{d_{\text{in}} \times r}$, with $r = d_{\text{in}} - k$, form an orthonormal basis of $\text{span}(\boldsymbol{U})^\perp$. Then $\boldsymbol{P} = \boldsymbol{U}_\perp \boldsymbol{U}_\perp^\top$, and the merged weight factorizes *exactly*:

$$\boldsymbol{W}\boldsymbol{P} \;=\; (\boldsymbol{W}\boldsymbol{U}_\perp)\,\boldsymbol{U}_\perp^\top \;=\; \boldsymbol{B}\boldsymbol{A}, \qquad \boldsymbol{A} \;=\; \boldsymbol{U}_\perp^\top \in \mathbb{R}^{r \times d_{\text{in}}}, \quad \boldsymbol{B} \;=\; \boldsymbol{W}\boldsymbol{U}_\perp \in \mathbb{R}^{d_{\text{out}} \times r}, \quad (6)$$

a rank-r factorization with $(d_{\text{in}} + d_{\text{out}})\, r$ parameters; on the output side, $\boldsymbol{P}\boldsymbol{W} = \boldsymbol{U}_\perp(\boldsymbol{U}_\perp^\top \boldsymbol{W})$. Every member of an input-tied group shares $\boldsymbol{A}$, which is stored and applied once; in attention, the common latent $\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x}$ suffices to reconstruct both keys and values, so the KV cache can store $\boldsymbol{z}$ alone.

3 Experiments

We evaluate LSP up to 70% compression on decoder language models (Section 3.1) and on vision transformers (Section 3.2), then measure inference efficiency (Section 3.3) and analyze the subspaces LSP learns (Section 3.4).

Setup. For language models we compress OPT-125M/1.3B (Zhang et al., 2022), Qwen3-4B (Yang et al., 2025) and Llama-2-7B (Touvron et al., 2023). We report WikiText-2 (Merity et al., 2016) test perplexity, calibrating as SliceGPT does on 1024 sequences of 2048 tokens from the training split, and zero-shot accuracy on six benchmarks with the calibration data given in Section 3.1. Early stopping and every tuned setting are selected on held-out validation splits. On Qwen3-4B, whose grouped-query K and V are a quarter of Q's width, K and V share an output-side projector and Q is projected alone; every other model ties Q/K/V on the input side (Section B). For vision we compress a ViT-B/16 (Dosovitskiy et al., 2021) fine-tuned on CIFAR-100 (Krizhevsky, 2009) and report source accuracy and linear-probe transfer. A compression ratio is the fraction of the parameters of the linear layers (no embeddings, no head) that is removed. All baselines run from their official code with same LSP's calibration data and precision; SVD baselines are evaluated in factorized form, with tunable settings swept around the official optima (Section B).

Table 1: Wikitext-2 perplexity (↓) of decoder LLMs at 30/50/70%. Principle that selects the removed directions or the final weights are on the left of each method.

Comp.		Method	OPT-125M	OPT-1.3B	Qwen3-4B	Llama2-7B
Dense (Base)			27.9	14.7	8.0	5.5
-30%	*activations/ reconstruc.*	NoLSP	48.5	23.0	16.8	9.4
		ASVD	106.8	396.1	154.5	98.2
		SliceGPT	51.7	23.9	15.0	10.3
		SVD-LLM (W)	48.7	20.2	19.1	10.4
		Swift-SVD	47.0	19.8	15.6	10.4
	loss-aware	LLM-Surgeon	31.7	16.2	11.2	8.9
		Dobi-SVD	63.2	23.8	18.5	10.0
		SVD-LLM	31.1	13.9	10.3	6.5
		LSP^T (Ours)	**24.5**	**12.9**	9.2	**6.4**
		LSP (Ours)	31.0	15.7	**9.1**	6.6
-50%	*activations/ reconstruc.*	NoLSP	206.9	81.2	43.5	29.8
		ASVD	1463.2	5213.1	3586.5	$>10^4$
		SliceGPT	97.2	57.8	32.0	25.2
		SVD-LLM (W)	126.5	44.2	82.1	30.2
		Swift-SVD	114.9	39.5	41.4	29.4
	loss-aware	LLM-Surgeon	47.3	22.5	25.8	17.8
		Dobi-SVD	183.1	54.0	39.3	28.1
		SVD-LLM	54.5	18.7	14.1	8.4
		LSP^T (Ours)	**30.6**	**16.4**	12.5	8.5
		LSP (Ours)	35.4	18.5	**11.4**	**8.0**
-70%	*activations/ reconstruc.*	NoLSP	1899	922.8	1274	222.8
		ASVD	3849.0	9080.1	8991.4	$>10^5$
		SliceGPT	290.1	262.7	89.0	86.3
		SVD-LLM (W)	633.4	590.0	2246.7	214.9
		Swift-SVD	561.0	412.8	1458.5	203.6
	loss-aware	LLM-Surgeon	113.7	53.9	371.0	105.0
		Dobi-SVD	759.6	338.3	92.5	61.2
		SVD-LLM	187.6	36.0	22.5	13.3
		LSP^T (Ours)	**42.4**	23.3	19.4	13.0
		LSP (Ours)	48.9	**22.4**	**16.2**	**10.9**

Baselines. *Activation- and reconstruction-based bases*, constructed from activation statistics or reconstruction criteria: SliceGPT (Ashkboos et al., 2024), ASVD (Yuan et al., 2024), SVD-LLM (W) (Wang et al., 2025d), whose whitening minimizes the layer-wise reconstruction error, and Swift-SVD (Qi et al., 2026), which reaches the same per-layer optimum from the output covariance and adds a per-matrix rank allocation. *Loss-aware*: LLM-Surgeon (van der Ouderaa et al., 2024), through a Kronecker-factored curvature model, and Dobi-SVD (Wang et al., 2025b), which learns only the per-matrix truncation rank within a closed-form basis, run without its quantization step so that parameter counts match. SVD-LLM is reported also with its LoRA recovery fine-tuning. On vision we compare with SliceGPT, SVD-LLM (W), FLAR-SVD (Thoma et al., 2025), and PELA (Guo et al., 2024). The last two are proposed on vision models, with PELA retraining all parameters by distilling every block's output features from the dense model. NoLSP is the non-trained version of LSP, controlling for the learning stage at fixed initialization, ranks and ties.

Table 2: Zero-shot accuracy (↑, %). Left: Llama-2-7B at 30/50/70% compression, every method calibrated on Alpaca. Right: Qwen3-4B at 20/40/60% compression, C4 calibration. Both LSP variants lead every baseline in mean accuracy at every ratio, by a margin that grows with the ratio.

	Llama-2-7B (Alpaca calibration)								**Qwen3-4B** (C4 calibration)							
Method	**Comp.**	**ARC-e**	**PIQA**	**Openb.**	**WinoG.**	**HellaS.**	**MathQA**	**Avg.**	**Comp.**	**ARC-e**	**PIQA**	**Openb.**	**WinoG.**	**HellaS.**	**MathQA**	**Avg.**
Dense (Base)	–	75.6	77.8	32.8	69.9	57.1	28.1	56.9	–	79.0	77.9	32.0	70.3	54.6	53.8	61.3
SVD-LLM (W)	-30%	61.4	67.2	24.4	61.3	38.7	23.0	46.0	-20%	59.0	70.5	22.4	59.4	40.1	26.5	46.3
Swift-SVD		61.6	68.1	24.0	61.5	38.9	23.0	46.2		63.6	72.2	26.8	64.2	45.7	28.6	50.2
Dobi-SVD		53.3	64.1	20.4	57.5	35.7	22.8	42.3		57.5	69.0	22.6	62.9	41.5	26.7	46.7
SVD-LLM		68.5	72.7	30.2	64.7	49.0	24.3	51.6		63.1	72.7	27.2	64.9	47.6	31.2	51.1
LSP^T (Ours)		71.3	74.6	30.2	63.3	50.7	27.3	**52.9**		77.7	77.1	29.0	65.1	51.0	38.6	56.4
LSP (Ours)		71.2	75.4	30.0	64.0	50.4	26.4	**52.9**		78.3	76.7	30.8	65.6	51.8	40.0	**57.2**
SVD-LLM (W)	-50%	41.0	58.4	17.4	52.2	29.6	21.9	36.8	-40%	36.3	59.6	13.0	50.8	29.3	22.3	35.2
Swift-SVD		41.8	58.7	19.0	52.5	29.8	22.1	37.3		50.8	66.1	18.4	58.8	34.1	23.2	41.9
Dobi-SVD		36.4	57.7	16.2	51.7	29.3	22.0	35.6		38.0	60.7	15.0	55.6	32.2	21.6	37.2
SVD-LLM		57.3	65.7	24.6	57.4	39.9	22.5	44.6		58.1	67.5	21.2	58.3	39.7	23.4	44.7
LSP^T (Ours)		64.3	69.6	25.0	57.5	43.6	23.6	47.3		69.7	72.7	27.2	60.5	45.7	27.7	50.6
LSP (Ours)		65.2	70.8	24.6	58.6	43.4	23.9	**47.8**		70.6	72.3	28.8	60.9	45.4	26.5	**50.7**
SVD-LLM (W)	-70%	28.7	53.4	14.8	49.3	26.5	20.3	32.2	-60%	26.9	54.5	12.4	50.0	26.5	21.6	32.0
Swift-SVD		28.8	52.7	13.8	49.4	26.4	20.0	31.9		30.2	56.1	12.6	49.3	27.3	22.2	33.0
Dobi-SVD		29.3	54.9	14.4	52.6	26.5	21.5	33.2		28.1	55.5	14.6	52.0	27.1	21.9	33.2
SVD-LLM		39.5	58.2	16.2	50.9	29.9	21.3	36.0		36.9	59.4	14.8	51.1	30.7	20.8	35.6
LSP^T (Ours)		50.3	63.2	20.0	52.9	33.8	22.0	40.4		55.5	68.7	21.6	53.7	37.6	22.9	43.3
LSP (Ours)		55.6	66.1	21.8	51.7	36.1	21.9	**42.2**		56.9	68.8	21.2	55.3	37.3	23.3	**43.8**

3.1 Large Language Models

Perplexity. Table 1 reports WikiText-2 test perplexity at -30/50/70% compression. One of the two LSP variants has the lowest perplexity in all 12 model–ratio settings, and the margin grows with the ratio. At -70% the training-free SVD methods deteriorate sharply, and so does NoLSP, the untrained initialization at the same ranks: learning the directions is what keeps the compressed models usable. No loss-aware baseline closes the gap at -70%, including LoRA-recovered SVD-LLM, the strongest baseline on the three larger models. The task loss (LSP^T) leads mostly on the smaller models and at lower ratios, where it can fall below dense test perplexity (both OPT models at -30%). Since the calibration text is the WikiText-2 training split, this reflects specialization to the evaluation domain (Section 2.3); LoRA-recovered SVD-LLM shows same effect on OPT-1.3B. Distillation (LSP) leads instead on the larger models and at higher ratios.

Zero-shot accuracy. Optimizing compression on a target objective raises a natural concern: does the compressed model retain *general* capabilities, or is the retained capacity overfit to the calibration objective? Table 2 follows two protocols from the literature on six commonsense, science and math benchmarks, OpenBookQA (Mihaylov et al., 2018), ARC-easy (Clark et al., 2018), WinoGrande (Sakaguchi et al., 2019), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020) and MathQA (Amini et al., 2019): Llama-2-7B at -30/50/70% with Alpaca calibration (Taori et al., 2023), following Wang et al. (2025d), and Qwen3-4B at -20/40/60% with C4 calibration (Raffel et al., 2020), following Qi et al. (2026). We compare with the low-rank baselines, the closest to LSP (Section B). Both LSP variants lead the strongest training-free baseline in mean accuracy at every ratio, by 6.7–10.5 points on Llama-2-7B and 6.2–10.8 on Qwen3-4B. The two objectives are within two points of each other, with distillation ahead or tied in every setting.

3.2 Vision Transformers

Vision models let us control the calibration data cleanly: we can change the composition of the calibration set at a fixed size, and evaluate the compressed network both on its source task and, through linear probes, on unseen datasets. We use this to ask how much each method depends on seeing the evaluation distribution at compression time. We build two calibration pools of the same size: a *single* pool of 47k CIFAR-100 training images, in-domain with the model and its evaluation task, and a *diverse* pool that keeps 10k of them and adds Food-101 (Bossard et al., 2014), CIFAR-10 (Krizhevsky, 2009), EuroSAT (Helber et al., 2018), STL-10 (Coates et al., 2011) and DTD (Cimpoi et al., 2014), capped at 10k images each. The added datasets share no label space with CIFAR-100, so Table 3 compares only label-free methods: the training-free baselines; PELA, whose feature distillation needs no labels; and LSP, whose distillation target is the dense model's CIFAR-100 output distribution for every image, whatever its source.

Accuracy under calibration shift. Calibrated in-domain, the training-free baselines are close to LSP at 30%, but the gap opens with the ratio (Table 3): at 70% LSP still scores 85.9 against the dense model's 89.9, where the strongest training-free baseline falls to 77.1. PELA, which retrains every weight, is the only baseline on par with LSP. Moving the calibration data away from the evaluation distribution costs every method source accuracy, and LSP the least; since the two pools are matched in size, this isolates the composition of the calibration data from its amount.

Downstream transfer. *Transfer* is the mean frozen-feature linear-probe accuracy over Pets (Parkhi et al., 2012), Aircraft (Maji et al., 2013) and Places365 (Zhou et al., 2017), none of them in any calibration pool, and *Gain* compares the two pools at matched size. Calibration diversity helps the baselines little, while LSP has the largest gain at every ratio and the best transfer on the diverse pool at all three ratios, including 50%, where it trails on the single pool (per-target breakdown in Section G). Label-free training alone does not explain this: PELA trains on the same pools with the same budget, reaches the best single-pool transfer at 50 and 70%, but gains less from diversity than LSP at every ratio. A KL label-free objective, therefore, not only provides the majority of best performance in either language and vision benchmarks, but also opens to a new compression axis towards better zero-shot performance, exploiting unlabelled data to mimic the original abilities of the dense model.

Table 3: ViT-B/16 compressed on a *single* pool (∼47k CIFAR-100 images) or a *diverse* pool (∼47k images from CIFAR-100, Food-101, CIFAR-10, EuroSAT, STL-10 and DTD). *Source*: CIFAR-100 accuracy; *Transfer*: mean linear-probe accuracy on Pets, Aircraft and Places365, in neither pool; *Gain*: diverse minus single transfer, computed before rounding.

		Single pool		Diverse pool		
Comp.	**Method**	*Source*	Transfer	*Source*	*Transfer*	*Gain*
Dense (Base)		89.9	47.4	89.9	47.4	—
-30%	SliceGPT	86.3	38.5	85.1	40.0	+1.5
	SVD-LLM (W)	88.2	41.2	87.7	42.3	+1.1
	FLAR-SVD	88.8	**43.6**	88.6	44.7	+1.1
	PELA	**89.2**	42.3	88.9	43.9	+1.6
	LSP (Ours)	**89.3**	**43.9**	**89.2**	**47.5**	**+3.7**
-50%	SliceGPT	83.3	33.8	78.8	36.9	+3.2
	SVD-LLM (W)	86.2	38.1	84.1	38.6	+0.5
	FLAR-SVD	86.4	38.1	83.1	36.2	−2.0
	PELA	**88.5**	**40.1**	**87.8**	41.8	+1.7
	LSP (Ours)	88.3	37.6	87.7	**44.7**	**+7.1**
-70%	SliceGPT	66.3	26.6	44.5	29.7	+3.1
	SVD-LLM (W)	77.1	31.7	66.2	33.4	+1.7
	FLAR-SVD	71.5	26.6	64.6	27.9	+1.3
	PELA	85.3	**34.1**	82.8	38.3	+4.2
	LSP (Ours)	**85.9**	33.0	**84.6**	**39.6**	**+6.6**

3.3 Inference Efficiency

Figure 3 compares decode throughput and the longest context that fits on one GPU for Llama-2-7B at all three ratios (LSP ^{T} has the same factor shapes and is omitted). Table 4 details the 70% checkpoints in two deployments of the same factorized model: with a standard full-width KV cache, in which throughput is measured (SDPA, CUDA-graph decoding), and with a latent cache, for which cache size and context capacity are computed (Section E).

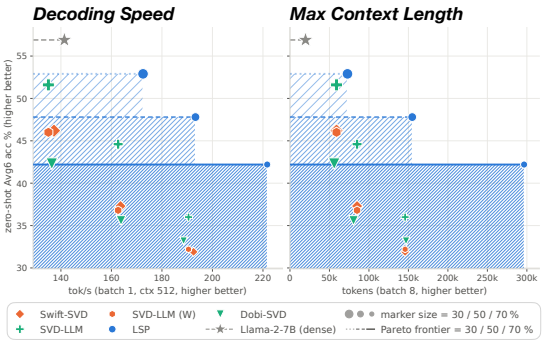


Figure 3: **Inference efficiency on Llama-2-7B.** Zero-shot accuracy (Avg over the six benchmarks of Table 2) at $-30/50/70\%$ compression (marker size) against CUDA-graph-compiled decode throughput at batch 1 and 512 tokens of context (left), and against the number of tokens per sequence whose weights and latent KV cache fit on one GH200 at batch size 8 (right). Lines trace the Pareto frontier at each ratio (dotted, dashed, solid), hatched in the color of the method that attains it.

Table 4: **Inference efficiency of -70% compressed Llama-2-7B.** The factorized methods match on weights and FLOPs; LSP's tied low-rank projector caches one narrow latent per layer instead of separate K and V, so its KV cache is 2.2–2.6× smaller than the untied factorizations' and 14.7× smaller than dense, it holds the longest context on one GPU; LSP also decodes fastest at full-width KV cache.

	Dense	*SVD-LLM*	*Swift-SVD*	*Dobi-SVD*	**LSP (Ours)**
Memory (batch 8)					
Weights (GiB)	12.55	4.11	4.11	4.12	4.11
KV floats/tok/layer	8192	1228	1228	1434	**556**
vs. dense	–	6.7×	6.7×	5.7×	**14.7×**
W+KV @128k (GiB)	524.6	80.9	80.9	93.7	**38.9**
vs. dense	–	6.5×	6.5×	5.6×	**13.5×**
Max ctx (k tokens)	19.7	145.8	145.8	124.9	**322.1**
Compute and speed (batch 1)					
Prefill TFLOPs @2k	29.26	10.69	10.69	10.71	10.69
Decode GFLOP/step @2k	14.29	5.22	5.22	5.23	5.22
Decode tok/s @512	141	191	193	189	**221**
@4k	85	100	101	99	**108**
@16k	36	38	38	38	**40**

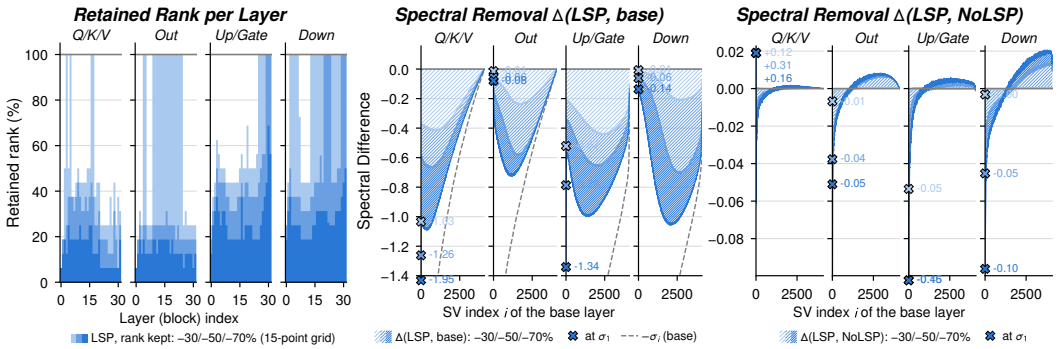


Figure 4: **Rank allocation and spectral removal on Llama-2-7B.** *Left:* fraction of full rank retained by each unit after compression, by block and projection type; measured-KL allocation is far from uniform. *Middle:* amplitude Δ_i lost along each singular direction of the original weights; average over layers; *Right:* difference to NoLSP, $\sigma_i(a_i^{\text{LSP}} - a_i^{\text{NoLSP}})$; the leading direction changes most, and outside Q/K/V LSP removes more of it and retains more of the trailing directions, most visibly in the down projection. All panels show runs at $-30/50/70\%$ compression (light to dark). Crosses give the delta for σ_1. Plots for more models and layers in Section D

Decode throughput. At small batch sizes, decoding is memory-bound: each step reads every weight once. At the same compression ratio, all factorizations read about the same number of weight bytes, so what separates them is how these bytes are split into matrix multiplies, since small multiplies achieve lower memory bandwidth than large ones. LSP issues fewer of them: a tied group runs one shared input factor for Q/K/V where an untied factorization runs three, and a unit the allocator leaves dense (Section 3.4) runs as one matrix multiply rather than two. It is the fastest factorized model in all model–ratio settings we measure (Section E); on Llama-2-7B, it decodes faster than the dense model at every ratio ($1.21\times$, $1.36\times$, and $1.56\times$ at 30, 50, and 70%), whereas every untied factorization is slower than dense at 30%.

Latent-cache memory. Factorizing $\boldsymbol{W} = \boldsymbol{BA}$ (Equation (6)) does not by itself shrink the KV cache, since $\boldsymbol{B}$ restores the full width, but caching the latent $\boldsymbol{z} = \boldsymbol{Ax}$ does, given a dedicated attention implementation (Section A.6). As in MLA and Palu (DeepSeek-AI, 2024; Chang et al., 2025), the value factor absorbs into the output projection, while under rotary embeddings keys are rebuilt from the latent at each step, at a cost linear in context length and latent width that any latent factorization pays; the latent cache thus serves capacity rather than speed (Section A.7). A tied group caches one latent where an untied factorization caches two, and the allocator compresses Q/K/V strongly (Section 3.4), so LSP has the narrowest cache (Table 4). At 70%, with 128k cached tokens and batch size 8, weights plus cache take $13.5\times$ less memory than dense, against at most $6.5\times$ for untied factorizations. A 95.5 GiB GPU thus holds about 320k cached tokens per sequence, against at most 146k: a memory capacity well beyond Llama-2-7B's 4k training context.

3.4 What Does LSP Compress?

Allocation. Measured-KL allocation is far from uniform (Figure 4, left). The Q/K/V groups are compressed far more than the MLP projections, even though each direction removed from gate/up saves more parameters (26,112) than one removed from Q/K/V (16,384): per saved parameter, attention inputs are markedly cheaper to compress. This is what keeps the shared K/V latent narrow at high ratios (about a tenth of full rank at 70%), and hence drives the KV-cache savings of Section 3.3. Later blocks retain more rank than earlier ones at every ratio, and at low compression many units (untied layers or tied groups) stay dense, 53 of 128 at 30%: a unit is factorized only when its allocated rank falls below the break-even rank at which the two factors become smaller than the dense weight. The same contrast between attention and MLP projections holds across all LLMs (Figure 5).

Retained subspace. Let $\boldsymbol{w}_i$ be the i-th singular vector of an original weight on its projected side, and $a_i = \|\boldsymbol{P}\boldsymbol{w}_i\|_2 \in [0, 1]$. Projection shrinks the i-th rank-one term of the weight from σ_i to $\sigma_i a_i$, so $\Delta_i = \sigma_i(a_i - 1)$ is the amplitude lost along that direction (Figure 4, middle). Unlike weight-SVD truncation, removal has no sharp cutoff: it spans the whole spectrum, deepest on the leading directions in Q/K/V and inside the spectrum in the other projections. Learning shifts it further (Figure 4,

right): relative to NoLSP, which shares the allocated ranks, LSP generally removes more of its high singular directions (left part) in exchange for small singular direction ones (right part). The leading direction changes most: direction 1 has the largest absolute difference to NoLSP. Notice that learning, the only difference between the two, lowers perplexity at 70% from 222.8 to 10.9 (Table 1). The weight spectrum thus ranks directions poorly by function: whitened truncation discards trailing directions the output depends on, and learning corrects it, most strongly along the leading direction.

4 Related Work

Pruning. Pruning removes weights scored by magnitude (Han et al., 2015; 2016), by their movement during fine-tuning (Sanh et al., 2020), or by a local quadratic model of the loss (LeCun et al., 1990; Hassibi et al., 1993). At LLM scale, the scores come from layer-wise reconstruction (Frantar & Alistarh, 2023), activation-scaled magnitudes (Sun et al., 2024), gradient-based group importance (Ma et al., 2023) or layer redundancy (Men et al., 2025). Recent methods learn what to remove against a global objective with the weights frozen, as semi-structured masks (Fang et al., 2024; Liu et al., 2025) or per-block widths (Gao et al., 2024b), part of a broader shift from local to global criteria in structured pruning (Wang et al., 2026). LSP brings this idea to low-rank compression: the removed set is a continuous subspace rather than a set of coordinates, so no discrete relaxation is needed, and the savings are realized as dense low-rank factors.

Low-rank compression. Trained weights are not low-rank themselves, so most low-rank methods rely on the activations lying in lower-dimensional subspaces (Yuan et al., 2024; Garg et al., 2024). They choose the removed subspace per layer in closed form, either from activation statistics, as in ASVD (Yuan et al., 2024), SliceGPT (Ashkboos et al., 2024), MoDeGPT (Lin et al., 2025), FLAR-SVD (Thoma et al., 2025) and further variants (Huang et al., 2025; Li et al., 2025; Chiang et al., 2026), or from the layer's output reconstruction error, as in SVD-LLM (Wang et al., 2025d) and Swift-SVD (Qi et al., 2026). Acting on one layer in isolation, these criteria do not see the network output, and their errors accumulate through depth (Odema et al., 2026). Where the loss is used, it enters through a local surrogate, as in FW-SVD (Hsu et al., 2022) and LLM-Surgeon (van der Ouderaa et al., 2024), or by fine-tuning the weights after truncation, as in SVD-LLM's LoRA stage and PELA (Guo et al., 2024); LSP instead optimizes the removed subspace itself against the network output, with the weights frozen.

Rank allocation and shared bases. Beyond choosing directions, compression also decides how the remaining capacity is laid out across the network. Ranks are allocated non-uniformly across matrices from measured or estimated sensitivity (Yuan et al., 2024; Wang et al., 2025c; Qi et al., 2026) or learned through a relaxation of the truncation position (Wang et al., 2025b), and factors are shared among matrices, across layers (Wang et al., 2025a) or among the Q/K/V projections that read the same activation (Wang et al., 2025e). The latter two adapt the structure of the compressed model to the network, but the basis at each rank remains closed-form. LSP uses the same two levers: projectors are tied within each group of layers that share an input, and ranks are allocated per compression unit by applying whitened projections and measuring the change in output KL (Section 2.2).

5 Conclusion

We introduced Learnable Subspace Projections (LSP), which compresses a pretrained network by learning, for each linear layer, the subspace the network can do without. Its premise is that statistical redundancy, whether measured by activation energy or by layer-wise reconstruction error, is not functional redundancy. LSP therefore selects the removed subspaces against the network output: it allocates ranks by measured output KL and optimizes all projectors jointly, with the pretrained weights frozen. Across four decoder LLMs, one of its two objectives attains the lowest perplexity in all twelve model–ratio settings, and both achieve the best zero-shot accuracy among low-rank methods. The margin grows with the compression ratio: at 70% on Llama-2-7B, LSP reaches 10.9 perplexity, against 13.3 for the strongest baseline and 222.8 without learning. On a vision transformer, it is the most robust to calibration shift and transfers best when calibrated on diverse unlabeled data. The merged model is also the fastest factorization we measure, decoding up to $1.56\times$ faster than the dense model. Its shared latent reduces the memory of weights and KV cache by $13.5\times$ at a 128k-token context, twice the savings of untied factorizations.

6 AI DISCLOSURE STATEMENT

In this work, we used generative AI tools for: design or provide feedback on research methodology or experiments (feedback to double-check for correctness); implement methods (implementation portions; tested and manually checked); assist in formulating mathematical claims, providing critical ingredients, and helping writing proofs (with each step checked by two authors), refine hypotheses, support qualitative and thematic data analysis (data visualization and organization, tested and manually checked), interpret results (summarization). We have not used generative AI tools for: help develop theoretical models or conceptual frameworks, assist with translation, clean and reformat dataset, generating synthetic data.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

7 ETHICS STATEMENT

By substantially lowering the parameter and compute footprint of large models, LSP contributes to the democratization of pretrained transformers, making strong language and vision models deployable on edge devices and in domain-specialized applications where retraining from scratch is infeasible. As with any compression method, the compressed model is a lossy approximation of the original; while our experiments show small aggregate degradation, this loss may be unevenly distributed across subpopulations, domains, or rare-event behaviors that the calibration distribution under-represents. We recommend that downstream practitioners evaluate compressed models on the same fairness, robustness, and safety metrics they would apply to the dense baseline before deploying them.

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 7319–7328, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.568. URL https://aclanthology.org/2021.acl-long.568/.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms, 2019.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress Large Language Models by Deleting Rows and Columns. In *International Conference on Learning Representations (ICLR)*, 2024.

Massimo Bini, Karsten Roth, Zeynep Akata, and Anna Khoreva. Ether: Efficient finetuning of large-scale models with hyperplane reflections. In *International Conference on Machine Learning (ICML)*, 2024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, and Kai-Chiang Wu. Palu: Compressing kv-cache with low-rank projection. In *International Conference on Learning Representations (ICLR)*, 2025.

Hung-Yueh Chiang, Chi-Chih Chang, Yu-Chen Lu, Chien-Yu Lin, Kai-Chiang Wu, Mohamed S. Abdelfattah, and Diana Marculescu. UniQL: Unified quantization and low-rank compression for adaptive edge LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=iOGu4wtDTF.

M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Adam Coates, Andrew Ng, and Honglak Lee. An Analysis of Single Layer Networks in Unsupervised Feature Learning. In *AISTATS*, 2011. https://cs.stanford.edu/~acoates/papers/coatesleeng_aistats_2011.pdf.

DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL https://arxiv.org/abs/2405.04434.

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. URL https://openreview.net/forum?id=YicbFdNTTy.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Gongfan Fang, Hongxu Yin, Saurav Muralidharan, Greg Heinrich, Jeff Pool, Jan Kautz, Pavlo Molchanov, and Xinchao Wang. MaskLLM: Learnable semi-structured sparsity for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. Spotlight.

Elias Frantar and Dan Alistarh. Sparsegpt: massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024a. URL https://zenodo.org/records/12608602.

Shangqian Gao, Chi-Heng Lin, Ting Hua, Tang Zheng, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. DISP-LLM: Dimension-independent structural pruning for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

Isha Garg, Christian Koguchi, Eshan Verma, and Daniel Ulbricht. Revealing the utilized rank of subspaces of learning in neural networks. In *ICML Workshop*, 2024. URL https://arxiv.org/abs/2407.04797.

Yangyang Guo, Guangzhi Wang, and Mohan Kankanhalli. PELA: Learning parameter-efficient models with low-rank approximation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15699–15709, June 2024.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 1135–1143, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/ae0eb3eed39d2bcef4622b2499a05fe6-Abstract.html.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1510.00149.

Babak Hassibi, David G. Stork, Gregory Wolff, and Takahiro Watanabe. Optimal brain surgeon: extensions and performance comparisons. In *Proceedings of the 7th International Conference on Neural Information Processing Systems*, NIPS'93, pp. 263–270, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pp. 204–207. IEEE, 2018.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uPv9Y3gmAI5.

Xinhao Huang, You-Liang Huang, and Zeyi Wen. Sola: Leveraging soft activation sparsity and low-rank decomposition for large language model compression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Ilja Kuzborskij and Yasin Abbasi Yadkori. Low-rank bias, weight decay, and model merging in neural networks, 2025. URL https://arxiv.org/abs/2502.17340.

Yann LeCun, John S. Denker, and Sara A. Solla. *Optimal brain damage*, pp. 598–605. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. ISBN 1558601007.

Wei Li, Lujun Li, Hao Gu, You-Liang Huang, Mark G. Lee, Shengjie Sun, Wei Xue, and Yike Guo. Moe-SVD: Structured mixture-of-experts LLMs compression via singular value decomposition. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=acJ3vdFljk.

Chi-Heng Lin, Shangqian Gao, James Seale Smith, Abhishek Patel, Shikhar Tuli, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. Modegpt: Modular decomposition for large language model compression. In *International Conference on Learning Representations (ICLR)*, 2025.

Hongyi Liu, Rajarshi Saha, Zhen Jia, Youngsuk Park, Jiaji Huang, Shoham Sabach, Yu-Xiang Wang, and George Karypis. ProxSparse: Regularized learning of semi-structured sparsity masks for pretrained LLMs. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. URL https://arxiv.org/abs/2502.00258.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*, 2023.

S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

Mohanad Odema, Gabrielle De Micheli, Dayin Gou, Nilesh Malpeddi, Prathamesh Vaste, and Jacob Song. Understanding calibration and truncation error propagation in training-free low-rank compression for LLMs. In *Conference on Language Modeling (COLM)*, 2026. URL https://arxiv.org/abs/2608.08506.

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

Ruoling Qi, Yirui Liu, Xuaner Wu, Xiangyu Wang, Ming Li, Chen Chen, Jian Chen, Yin Chen, and Qizhen Weng. Swift-SVD: Theoretical optimality meets practical efficiency in low-rank LLM compression. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=nAQ4h8FpdM.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), January 2020. ISSN 1532-4435.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019.

Victor Sanh, Thomas Wolf, and Alexander M. Rush. Movement pruning: adaptive sparsity by fine-tuning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Nikul Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=WGXb7UdvTX.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=PxoFut3dWW.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Moritz Thoma, Jorge Villasante, Emad Aghajanzadeh, Shambhavi Balamuthu Sampath, Pierpaolo Mori, Maximilian Groetzinger, Daniil Dylkin, Manoj-Rohit Vemparala, Nael Fasfous, Alexander Frickenstein, Daniel Mueller-Gritschneder, and Ulf Schlichtmann. Flar-svd: Fast and latency-aware singular value decomposition for model compression. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR) Workshops*, pp. 1898–1907, June 2025.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Tycho FA van der Ouderaa, Markus Nagel, Mart van Baalen, Yuki M Asano, and Tijmen Blankevoort. The llm surgeon. In *International Conference of Learning Representations (ICLR)*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Jingcun Wang, Yu-Guang Chen, Ing-Chao Lin, Bing Li, and Grace Li Zhang. Basis sharing: Cross-layer parameter sharing for large language model compression. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Qinsi Wang, Jinghan Ke, Masayoshi Tomizuka, Kurt Keutzer, and Chenfeng Xu. Dobi-SVD: Differentiable SVD for LLM compression and some new perspectives. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=kws76i5XB8.

Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. SVD-LLM V2: Optimizing singular value truncation for large language model compression. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 4287–4296, 2025c. URL https://aclanthology.org/2025.naacl-long.217/.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations (ICLR)*, 2025d. URL https://openreview.net/forum?id=LNYIUouhdt.

Yutong Wang, Haiyu Wang, and Sai Qian Zhang. QSVD: Efficient low-rank approximation for unified query-key-value weight compression in low-precision vision-language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025e. URL https://arxiv.org/abs/2510.16292. Spotlight.

Ziyan Wang, Enmao Diao, Qi Le, Pu Wang, Minwoo Lee, Shu-ping Yeh, Evgeny Stupachenko, Hao Feng, and Li Yang. From local to global: Revisiting structured pruning paradigms for large language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Hao Yu and Jianxin Wu. Compressing transformers: features are low-rank, but weights are not! In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i9.26304. URL https://doi.org/10.1609/aaai.v37i9.26304.

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. Asvd: Activation-aware singular value decomposition for compressing large language models, 2024. URL https://arxiv.org/abs/2312.05821.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke

Zettlemoyer. Opt: Open pre-trained transformer language models. *CoRR*, abs/2205.01068, 2022. URL http://dblp.uni-trier.de/db/journals/corr/corr2205.html#abs-2205-01068.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

A METHOD DETAILS

A.1 OPTIMALITY OF THE ORTHOGONAL PROJECTOR

Section 2.1 applies the projector $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{U}\boldsymbol{U}^\top$. The following proposition is why that particular operator: once the removed subspace is fixed, the orthogonal projector is the unique least-disruptive map that annihilates it.

Proposition. Let $S \subset \mathbb{R}^d$ have orthonormal basis $\boldsymbol{U} \in \mathbb{R}^{d\times k}$, and let $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{U}\boldsymbol{U}^\top$ denote the orthogonal projector onto $S^\perp$. Among all $\boldsymbol{Q} \in \mathbb{R}^{d\times d}$ with $\boldsymbol{Q}\boldsymbol{u} = 0$ for every $\boldsymbol{u} \in S$, $\boldsymbol{P}$ is the unique minimizer of $\|\boldsymbol{I} - \boldsymbol{Q}\|_F$, attaining the optimal value $\sqrt{k}$.

Proof. Write $\boldsymbol{Q} = \boldsymbol{I} - \boldsymbol{E}$ so the annihilation constraint $\boldsymbol{Q}\boldsymbol{U} = 0$ becomes $\boldsymbol{E}\boldsymbol{U} = \boldsymbol{U}$. Let $\boldsymbol{U}_\perp \in \mathbb{R}^{d\times(d-k)}$ complete $\boldsymbol{U}$ to an orthonormal basis of $\mathbb{R}^d$. In this basis, every admissible $\boldsymbol{E}$ has the block form

$$\boldsymbol{E} \;=\; [\boldsymbol{U} \quad \boldsymbol{U}_\perp] \begin{bmatrix} \boldsymbol{I}_k & \boldsymbol{C} \\ 0 & \boldsymbol{D} \end{bmatrix} [\boldsymbol{U} \quad \boldsymbol{U}_\perp]^\top, \qquad \boldsymbol{C} \in \mathbb{R}^{k\times(d-k)},\ \boldsymbol{D} \in \mathbb{R}^{(d-k)\times(d-k)}, \tag{7}$$

where the constraint $\boldsymbol{E}\boldsymbol{U} = \boldsymbol{U}$ has fixed the $(1,1)$ block to $\boldsymbol{I}_k$ and the $(2,1)$ block to 0, while $\boldsymbol{C}$ and $\boldsymbol{D}$ are free. Orthogonal invariance of the Frobenius norm and the block decomposition give

$$\|\boldsymbol{E}\|_F^2 \;=\; \|\boldsymbol{I}_k\|_F^2 + \|\boldsymbol{C}\|_F^2 + \|\boldsymbol{D}\|_F^2 \;=\; k + \|\boldsymbol{C}\|_F^2 + \|\boldsymbol{D}\|_F^2 \;\geq\; k, \tag{8}$$

with equality iff $\boldsymbol{C} = 0$ and $\boldsymbol{D} = 0$. In that case $\boldsymbol{E} = \boldsymbol{U}\boldsymbol{U}^\top$ and $\boldsymbol{Q} = \boldsymbol{I} - \boldsymbol{U}\boldsymbol{U}^\top = \boldsymbol{P}$, establishing optimality and uniqueness. □

The proposition settles the operator, not the subspace: which directions are removed is chosen by the whitened criterion of Section 2.2 at initialization and by the loss thereafter, whereas among all maps that annihilate a given S only $\boldsymbol{P}$ leaves $S^\perp$ untouched. Because $\boldsymbol{P}$ is a projector ($\boldsymbol{P}^2 = \boldsymbol{P}$), the modulated operator of Section 2.1 at $\boldsymbol{d} = \mathbf{1}$ is the convex combination $(1-\alpha)\boldsymbol{I} + \alpha\boldsymbol{P}$, a true interpolation between identity and projection rather than an arbitrary contraction.

A.2 PRACTICAL DESIGN CHOICES

This section details the tied groups and the three design choices of Section 2.1. Throughout, ℓ indexes projectors (one per untied layer or tied group), d_ℓ is the dimension of the projected side, and k_ℓ is the number of removed directions.

Activation-space form. Applying the projector through the activation keeps no dense $d_{\text{out}} \times d_{\text{in}}$ projected weight in the autograd graph: the only additional state is $\boldsymbol{U}_\ell$ and the narrow activation $\boldsymbol{U}_\ell^\top \boldsymbol{x} \in \mathbb{R}^{k_\ell}$ per token, on top of the activations the frozen network already stores. The fused QR backward retains $\mathcal{O}(d_\ell k_\ell + k_\ell^2)$ state, compared with $\mathcal{O}(d_\ell k_\ell^2)$ for an explicit Gram–Schmidt loop; both return an orthonormal basis of $\operatorname{span}(\boldsymbol{V}_\ell)$ when $\boldsymbol{V}_\ell$ has full column rank, and hence the same projector in exact arithmetic. Since $k_\ell \leq d_\ell$, the projector and QR state totals $\mathcal{O}(\sum_\ell d_\ell k_\ell)$ over the network, plus $\mathcal{O}(\sum_\ell k_\ell)$ per token for the narrow activations.

Tied groups. For an input-tied group $\mathcal{G}$, we apply Equation (1) to the row-stacked weight $[\boldsymbol{W}^1; \ldots; \boldsymbol{W}^{|\mathcal{G}|}] \in \mathbb{R}^{(\sum_m d_{\text{out}}^m)\times d_{\text{in}}}$. At a common rank r, the merged group costs $r\,(d_{\text{in}} + \sum_{m\in\mathcal{G}} d_{\text{out}}^m)$ parameters, against $r\sum_{m\in\mathcal{G}}(d_{\text{in}} + d_{\text{out}}^m)$ for $|\mathcal{G}|$ separate factorizations (Equation (6)). Sharing thus saves $(|\mathcal{G}| - 1)\, d_{\text{in}} r$ parameters, which the group can spend on a higher rank, at the cost of constraining all members to the same input subspace. The same construction applies to an output-tied group, whose members write the same output: the weights are column-stacked, the output factor is shared, each member keeps its own input factor, and the merged group costs $r\,(d_{\text{out}} + \sum_{m\in\mathcal{G}} d_{\text{in}}^m)$ parameters (Equation (18)).

Warm-up, direction dropout, and orthogonality penalty. A projector that removes k directions is never the identity: $\|\boldsymbol{I} - \boldsymbol{P}\|_F = \sqrt{k}$ for every $\boldsymbol{U}$, an analogue of the fixed distance of reflections from the identity noted by Bini et al. (2024). The ramp of α in $\boldsymbol{P}_{\alpha,\boldsymbol{m}}$ (Section 2.1) therefore starts training from the identity. The mask entries $m_i \sim \text{Bernoulli}(1-p)$ are redrawn at every step, independently for each member of a tied group over the shared $\boldsymbol{U}$. We apply no $1/(1-p)$ rescaling: it would give each removed direction the coefficient $1-\alpha/(1-p)$, which becomes negative as $\alpha \to 1$ and would reflect the direction rather than remove it. Without rescaling, at $\alpha = 1$ the modulated $\boldsymbol{P}$

is an orthogonal projector onto the complement of the directions removed at that step. Both $\alpha < 1$ and the mask relax the target rank during training; it is reached only at $\alpha = 1$ and $\boldsymbol{m} = \mathbf{1}$.

The orthogonality penalty acts on the unconstrained columns of $\boldsymbol{V}_\ell$:

$$\mathcal{L}_{\text{ort}} = \sum_{\ell:\, k_\ell > 1} \frac{1}{k_\ell(k_\ell - 1)} \sum_{i \neq j} \left| \langle \boldsymbol{v}_i^{(\ell)}, \boldsymbol{v}_j^{(\ell)} \rangle \right|. \tag{9}$$

It discourages correlated columns, which in practice stabilizes QR and its gradient; orthonormality of $\boldsymbol{U}_\ell$ itself is already enforced by QR. We use the raw inner product rather than a cosine: the columns are unit-norm at initialization, and since the optimizer applies no weight decay, the raw form also keeps their norms from growing. Unlike $\boldsymbol{P}$ itself, direction dropout and the penalty depend on the chosen basis of $\mathrm{span}(\boldsymbol{V}_\ell)$, so during training the optimization is not fully invariant to rotations within the subspace; the exported projector is.

A.3 Whitened Initialization and Its Excess Error

We prove Proposition 1. Drop the superscript w from the SVD of $\boldsymbol{W}\boldsymbol{S}$, and partition complete singular bases as $[\boldsymbol{U}_r\ \boldsymbol{U}_{\text{rest}}]$ and $[\boldsymbol{V}_r\ \boldsymbol{V}_{\text{rest}}]$, padding $\boldsymbol{\Sigma}_{\text{rest}}$ with zero singular values where needed. As in the proposition, $\boldsymbol{A} = \boldsymbol{S}\boldsymbol{V}_r$, $\boldsymbol{B} = \boldsymbol{S}\boldsymbol{V}_{\text{rest}}$, and $\boldsymbol{P}_A = \boldsymbol{A}\boldsymbol{A}^+$. Since $[\boldsymbol{V}_r\ \boldsymbol{V}_{\text{rest}}]$ is orthogonal, rotating by it preserves the Frobenius norm, so for any $\boldsymbol{Z}$

$$\mathcal{E}(\boldsymbol{Z}) = \|(\boldsymbol{W} - \boldsymbol{Z})\boldsymbol{A}\|_F^2 + \|(\boldsymbol{W} - \boldsymbol{Z})\boldsymbol{B}\|_F^2, \qquad \boldsymbol{W}\boldsymbol{A} = \boldsymbol{U}_r\boldsymbol{\Sigma}_r, \quad \boldsymbol{W}\boldsymbol{B} = \boldsymbol{U}_{\text{rest}}\boldsymbol{\Sigma}_{\text{rest}}. \tag{10}$$

(i) Output side. $(\boldsymbol{I} - \boldsymbol{U}_{\text{rest}}\boldsymbol{U}_{\text{rest}}^\top)\boldsymbol{W} = \boldsymbol{U}_r\boldsymbol{U}_r^\top\boldsymbol{W}$, the first form of Equation (2), so its error is the optimum $\|\boldsymbol{\Sigma}_{\text{rest}}\|_F^2$.

(ii) Input side. Since $\boldsymbol{A}^\top\boldsymbol{S}^{-\top}\boldsymbol{V}_{\text{rest}} = \boldsymbol{V}_r^\top\boldsymbol{V}_{\text{rest}} = 0$ and the dimensions add to d_{in}, removing $\mathrm{span}(\boldsymbol{S}^{-\top}\boldsymbol{V}_{\text{rest}})$ leaves exactly $\mathrm{span}(\boldsymbol{A})$, so the initialized weight is $\boldsymbol{W}\boldsymbol{P}_A$. In Equation (10), the error on $\boldsymbol{A}$ vanishes because $\boldsymbol{P}_A\boldsymbol{A} = \boldsymbol{A}$, and the error on $\boldsymbol{B}$ is

$$\boldsymbol{W}(\boldsymbol{I} - \boldsymbol{P}_A)\boldsymbol{B} = \boldsymbol{U}_{\text{rest}}\boldsymbol{\Sigma}_{\text{rest}} - \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{A}^+\boldsymbol{B}. \tag{11}$$

The two terms have orthogonal column spaces, giving

$$\mathcal{E}(\boldsymbol{W}\boldsymbol{P}_A) = \|\boldsymbol{\Sigma}_{\text{rest}}\|_F^2 + \|\boldsymbol{\Sigma}_r\boldsymbol{A}^+\boldsymbol{B}\|_F^2, \tag{12}$$

where the first term is the optimal rank-r error and the second is the excess. Because $\boldsymbol{S}$ is invertible, $\boldsymbol{A}$ has full column rank and $\boldsymbol{A}^+ = (\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top$; if $\sigma_r > 0$, $\boldsymbol{\Sigma}_r$ is invertible as well, so the excess vanishes if and only if $\boldsymbol{A}^\top\boldsymbol{B} = 0$. Here $\boldsymbol{A}^+\boldsymbol{B}$ holds the least-squares coefficients of the removed input directions on the kept ones, so the excess is their squared norm weighted by the kept singular values.

(iii) Tied groups. For M layers sharing their input, and hence $\boldsymbol{S}$, such as Q/K/V, one shared kept subspace constrains the row-stacked approximation of $\bar{\boldsymbol{W}} = [\boldsymbol{W}_1; \ldots; \boldsymbol{W}_M]$ to rank r, and the summed error is $\sum_m \|(\boldsymbol{W}_m - \widehat{\boldsymbol{W}}_m)\boldsymbol{S}\|_F^2 = \|(\bar{\boldsymbol{W}} - \widehat{\bar{\boldsymbol{W}}})\boldsymbol{S}\|_F^2$. By Eckart–Young (Eckart & Young, 1936), the leading right singular subspace of $\bar{\boldsymbol{W}}\boldsymbol{S}$ gives the optimal shared subspace in whitened coordinates, and (ii) applies to $\bar{\boldsymbol{W}}$. For layers sharing their output side, such as K and V under grouped-query attention, one shared output projector $\boldsymbol{P}$ gives the summed error $\sum_m \|(\boldsymbol{I} - \boldsymbol{P})\boldsymbol{W}_m\boldsymbol{S}_m\|_F^2 = \|(\boldsymbol{I} - \boldsymbol{P})[\boldsymbol{W}_1\boldsymbol{S}_1\ \cdots\ \boldsymbol{W}_M\boldsymbol{S}_M]\|_F^2$, which the leading left singular subspace of the column-stacked whitened weight minimizes, and (i) applies. Output tying shares output coordinates, not a projected activation, so it yields no single latent to cache. □

Alternative recast. One could instead remove SVD-LLM's null space $\mathrm{span}(\boldsymbol{B})$ itself. With $\boldsymbol{P}_B = \boldsymbol{B}\boldsymbol{B}^+$, the same split gives

$$\mathcal{E}\big(\boldsymbol{W}(\boldsymbol{I} - \boldsymbol{P}_B)\big) = \|\boldsymbol{\Sigma}_{\text{rest}}\|_F^2 + \|\boldsymbol{\Sigma}_{\text{rest}}\boldsymbol{B}^+\boldsymbol{A}\|_F^2. \tag{13}$$

Neither excess dominates in general: the alternative's excess is weighted by the smaller singular values $\boldsymbol{\Sigma}_{\text{rest}}$, but involves $\boldsymbol{B}^+$, which can amplify error when the removed directions are ill-conditioned. We use $\boldsymbol{P}_A$, which reproduces SVD-LLM exactly on the kept subspace.

Numerical stabilization and output-side bases. The implementation accumulates the unnormalized input Gram matrix $\boldsymbol{H} = n\boldsymbol{G}$. Cholesky is first attempted without a ridge; if it fails, the

implementation replaces $\boldsymbol{H}$ by $\boldsymbol{H} + \eta\boldsymbol{I}$, with $\eta = -\lambda_{\min}(\boldsymbol{H}) + 10^{-6}$, and retries. Scaling the Cholesky factor by $1/\sqrt{n}$ gives a factor of $\boldsymbol{G} + (\eta/n)\boldsymbol{I}$, so the whitened optimum and the excess identities then concern the regularized objective

$$\mathcal{E}_\eta(\boldsymbol{Z}) = \mathcal{E}(\boldsymbol{Z}) + (\eta/n)\|\boldsymbol{W} - \boldsymbol{Z}\|_F^2, \tag{14}$$

with $\eta = 0$ when the first factorization succeeds. For an output-tied group, the implementation instead decomposes the summed bias-free output Gram matrix, proportional to $\sum_m \boldsymbol{W}_m \boldsymbol{G}_m \boldsymbol{W}_m^\top$; its eigenvectors are the left singular vectors of $[\boldsymbol{W}_1\boldsymbol{S}_1 \ \cdots \ \boldsymbol{W}_M\boldsymbol{S}_M]$, so no input Cholesky factor is required. The added ridge, 0.01 times the mean diagonal, shifts eigenvalues without changing eigenspaces in exact arithmetic.

A.4 Measured-KL Allocation

This section states the allocator of Section 2.2 in full. For a unit u with projected dimension d_u, dense parameter count N_u, and factor count $c_u(r)$ at retained rank r (input- or output-tied costs of Section A.2, counting each shared factor once), removing the trailing k basis directions saves

$$s_u(k) = \begin{cases} 0, & k = 0 \text{ (dense)}, \\ N_u - c_u(d_u - k), & k > 0, \end{cases} \tag{15}$$

which is negative for small k, since a shallow truncation stores more parameters than the dense unit. Each probe applies the actual initialized orthogonal projector, including the input-side recast of Proposition 1, and $\Delta_u(0) = 0$. We probe a grid of removal fractions, replace each curve Δ_u of Equation (3) by its running maximum so that it is nondecreasing, and interpolate linearly between probed counts, yielding $\widetilde{\Delta}_u$. For a target compression ratio τ, allocation uses the separable surrogate

$$\min_{\{k_u\}} \sum_u \widetilde{\Delta}_u(k_u) \quad \text{subject to} \quad \sum_u s_u(k_u) \geq \tau \sum_u N_u, \qquad k_u \in \mathcal{K}_u, \tag{16}$$

where $\mathcal{K}_u$ contains zero and the probed removal counts. Starting with all units dense, we solve it greedily: among all units and all probed $k' > k$ with $s_u(k') > s_u(k)$, we take the move with the smallest marginal cost

$$\frac{\widetilde{\Delta}_u(k') - \widetilde{\Delta}_u(k)}{s_u(k') - s_u(k)}, \tag{17}$$

until the target is reached. Because a move may span several grid points, a unit's first move can jump past the break-even rank; sensitive units may never move and stay dense. An integer search finally trims the last move to the closest attainable target.

A.5 LSP ALGORITHM

Algorithm 1 states the pipeline of Sections 2.1 to 2.3 end to end.

Algorithm 1 LSP.

Require: model M; calibration set $\mathcal{D}_{\text{cal}}$; held-out validation set $\mathcal{D}_{\text{val}}$; target compression τ; epochs E; modulation epochs E_α; dropout p; orthogonality weight λ_{ort}; learning rate η; patience P; merge tolerance ϵ_{SVD}

Replace each targeted linear layer (all but the head and the embeddings) with its LSP counterpart; tie compatible input groups (Q/K/V, gate/up), or K/V on the output side under grouped-query routing
Gather on $\mathcal{D}_{\text{cal}}$ without gradients: collect the input Gram for each input-side unit and each member's bias-free output Gram for an output-side unit
Build bases: use the row-stacked whitened weight for input-tied groups, or the summed output Gram for output-tied groups (Section A.3)
Probe: cache dense outputs on $\mathcal{D}_{\text{probe}} \subseteq \mathcal{D}_{\text{cal}}$; truncate one unit at a time over the removal grid and record $\Delta_u(k)$ (Equation (3))
Allocate: take running maxima and interpolate; compare all higher grid endpoints by marginal KL per parameter saved (Equation (17)); integer-trim only the last selected interval to the nearest target
Probe the allocation jointly and compare its KL with $\sum_u \widetilde{\Delta}_u(k_u)$; fix the selected ranks for training
Initialize: select each unit's trailing k_u directions; set $\boldsymbol{V} \leftarrow \mathrm{qf}(\boldsymbol{S}^{-\top}\boldsymbol{V}^{w}_{\text{rest}})$ for an input-side unit, or use the trailing output-Gram eigenvectors for an output-side unit ▷ *NoLSP at the selected ranks and ties*
Set up Adam over $\{\boldsymbol{V}_\ell\}$ with a linear-warmup cosine schedule over E epochs; every pretrained weight and bias stays frozen
for epoch $e = 1, \ldots, E$ **or** until early stop **do**
    for micro-batch in $\mathcal{D}_{\text{cal}}$ **do**
        $\alpha \leftarrow \min\big(1,\ t/(E_\alpha N_\mu)\big)$ ▷ t = *cumulative micro-batch index, starting at 1;* N_μ = *micro-batches per epoch per replica*
        for compressed unit **do**
            $\boldsymbol{U} \leftarrow \mathrm{qf}(\boldsymbol{V})$; draw $\boldsymbol{d} \sim \mathrm{Bernoulli}(1-p)^k$, independently per member of a tied group
            Forward with $\boldsymbol{P}_{\alpha,\boldsymbol{d}} = \boldsymbol{I} - \alpha\,\boldsymbol{U}\,\mathrm{diag}(\boldsymbol{d})\boldsymbol{U}^\top$, applied in activation space
        if distillation **then**
            $p_{\text{dense}} \leftarrow$ forward of the same M with the projections disabled, without gradients ▷ *no second copy of the weights*
        $\mathcal{L} \leftarrow \mathcal{L}_{\text{obj}} + \lambda_{\text{ort}}\mathcal{L}_{\text{ort}}, \quad \mathcal{L}_{\text{obj}} \in \{\mathcal{L}_{\text{KL}}\ (LSP),\ \mathcal{L}_{\text{task}}\ (LSP^T)\}$
        Backprop; update $\{\boldsymbol{V}_\ell\}$ and step the schedule after each gradient-accumulation window
    Evaluate on $\mathcal{D}_{\text{val}}$ (validation perplexity for language models, accuracy for the ViT); record $\{\boldsymbol{V}_\ell\}$ if best so far; **break** if no improvement for P epochs
Restore the best-validation $\{\boldsymbol{V}_\ell\}$
Merge at $\alpha = 1$, $\boldsymbol{d} = \mathbf{1}$ (Section A.6): factor the row-stacked input-projected or column-stacked output-projected weights by thin SVD, sharing $\boldsymbol{A}$ or $\boldsymbol{B}$ respectively; drop singular values below ϵ_{SVD}
return permanently compressed model M'

A.6 MERGE DETAILS

An input-tied group factors as $\boldsymbol{W}_m\boldsymbol{P} = \boldsymbol{B}_m\boldsymbol{A}$ with $\boldsymbol{B}_m = \boldsymbol{W}_m\boldsymbol{Q}$ and shared $\boldsymbol{A} = \boldsymbol{Q}^\top$ (Equation (6)). An output-tied group instead factors as

$$\boldsymbol{P}\boldsymbol{W}_m \;=\; \boldsymbol{B}\boldsymbol{A}_m, \qquad \boldsymbol{B} \;=\; \boldsymbol{Q} \in \mathbb{R}^{d_{\text{out}} \times r}, \quad \boldsymbol{A}_m \;=\; \boldsymbol{Q}^\top\boldsymbol{W}_m \in \mathbb{R}^{r \times d^m_{\text{in}}}, \tag{18}$$

with the output factor $\boldsymbol{B}$ shared and $\boldsymbol{Q}$ an orthonormal basis of $\mathrm{span}(\boldsymbol{U})^\perp$ in the output space. Both factorizations are exact before numerical truncation and leave the bias unchanged. In practice, factors are read from a thin SVD of the merged weight: row-stacked for an input tie, column-stacked

for an output tie. If all r latent directions are nonzero, this differs from the displayed factors only by an invertible change of latent basis. Rank deficiency or the numerical merge tolerance can reduce the exported rank below r; dropping nonzero singular values at that tolerance introduces a further approximation.

For input-tied K/V, a latent cache stores $\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x}$ and reconstructs keys and values as $\boldsymbol{B}_k\boldsymbol{z}$ and $\boldsymbol{B}_v\boldsymbol{z}$, adding biases and applying rotary embeddings after reconstruction as appropriate. Output-tied K/V generally needs separate latents $\boldsymbol{A}_k\boldsymbol{x}$ and $\boldsymbol{A}_v\boldsymbol{x}$, even though it shares $\boldsymbol{B}$. Either latent-cache implementation requires changes beyond replacing linear layers with two factors. No QR, modulation or direction dropout remains after export; merging uses the deterministic evaluation projector.

A.7 Limitations

(i) The measured-KL allocation is a local surrogate: isolated unit costs omit interactions, and current strategy has no global-optimality guarantee. This makes the allocation uncertain on quality in specific settings: in a few cases uniform ranks can still be preferred; however empirical evidence shows that measured KL is still better, since able to adapt to different architectures: performance improvement over Qwen3-4B, OPT-1.3B, and two out of three compression ranges of Llama2-7B. (ii) Compression is also not free: LSP requires both KL probing and projection training, although the probe is reusable across ratios and objectives. On the other hand, compression is a one-off cost that is repaid by the inference savings of Section 3.3 and LSP justifies this with the strong performance. (iii) Tied groups only lead to savings with latent cache if KV are tied on the input side. This means our best configuration for Qwen3-4B does not gain in efficiency as much as the input-tied version does. This represents a performance-efficiency trade-off, as shown in Table 11: tying KV on the output side leads to the best performance, but one can choose to tie KV on the input side for improved inference efficiency instead.

B Implementation Details

Shared configuration. The main accuracy tables use the measured-KL allocation of Equations (3) and (17) against the compressible-linear parameter count, the whitened initialization of Section 2.2 with a Cholesky whitener, shared-input tying (with the Qwen exception below), a linear-warmup cosine schedule, Adam with no weight decay, modulation ramped to 1 over the first epoch, $\lambda_{\text{ort}} = 0.05$, and a factorization tolerance of 0.05 below which singular directions are dropped at merge. Weights are loaded in bf16; compression arithmetic (covariance accumulation, Cholesky, SVD, QR) runs in fp32 on Qwen3-4B and Llama-2-7B and in fp64 on OPT-125M, OPT-1.3B and ViT-B/16.

Probe settings and tying. The LLM probes use 128K calibration tokens and the ViT probes $4{,}096$ images, comparing the dense and the truncated model's next-token or class distributions. The removal grids are $i/8$ ($i = 1, \ldots, 7$) on OPT and ViT, and $i/16$ ($i = 1, \ldots, 15$) on Qwen3-4B and Llama-2-7B. Each probe truncates one unit with every other unit dense, and the allocation is not refined jointly afterwards. Probing units in isolation is an additive surrogate: on the selected allocations the joint KL of the compressed model exceeds the sum of the isolated probe costs by 2.1–$4.3\times$ on Llama-2-7B, 1.7–$3.3\times$ on Qwen3-4B, 2.2–$4.6\times$ on the OPT models and 1.5–$14\times$ on the ViT, generally rising with the ratio. On Qwen3-4B, K and V share an output-side factor, following the grouped-query routing, rather than the forced Q/K/V input tie; the probe leaves them mostly dense (Table 11).

Replicates. Table 5 provides evidence of LSP robustness, reporting the standard deviations over three seeds for LSP results in Section 3.

Per-model settings. Calibration is 1024 sequences of 2048 tokens for the decoder models and $10{,}000$ images for the ViT covariance pass, with a per-epoch draw of 1024 sequences (4092 images) for the training stage. Batch sizes are 32 on the OPT models, 8 globally on Qwen3-4B and Llama-2-7B (four-way data parallel at micro-batch 1), and 512 on ViT. Per-direction dropout is $p = 0.05$ (0.1 on Llama-2-7B). Validation is a held-out split throughout, so that no test data is seen during selection.

Table 5: Seed sensitivity of LSP and LSP ^{T}: mean $\pm$ standard deviation over three full-pipeline seeds, each run read at its own validation-selected epoch. The LLMs are wikitext-2 perplexity, ViT-B/16 is CIFAR-100 accuracy on the single (CIFAR-100 only) pool.

Model	Metric	Method	-30%	-50%	-70%
OPT-125M	ppl ↓	LSP ^{T}	24.48 ± 0.11	30.63 ± 0.28	42.41 ± 0.06
		LSP	30.95 ± 0.42	35.35 ± 0.67	48.88 ± 0.24
OPT-1.3B	ppl ↓	LSP ^{T}	12.93 ± 0.02	16.42 ± 0.11	23.33 ± 0.26
		LSP	15.70 ± 0.15	18.51 ± 0.11	22.39 ± 0.30
Qwen3-4B	ppl ↓	LSP ^{T}	9.20 ± 0.03	12.48 ± 0.02	19.40 ± 0.14
		LSP	9.13 ± 0.07	11.42 ± 0.10	16.18 ± 0.08
Llama-2-7B	ppl ↓	LSP ^{T}	6.39 ± 0.01	8.43 ± 0.06	13.03 ± 0.10
		LSP	6.62 ± 0.05	8.05 ± 0.08	10.94 ± 0.12
ViT-B/16	acc ↑	LSP ^{T}	89.26 ± 0.13	87.80 ± 0.17	84.21 ± 0.06
		LSP	89.38 ± 0.19	88.32 ± 0.14	85.92 ± 0.48

Selection. The *epoch* is selected on validation data, and the selected epoch's projections are the ones merged and reported. The *learning rate* is swept, keeping the best validation setting per model and ratio, which is the same rule applied to every baseline that exposes a tunable knob.

Compression ratio. A ratio is the fraction of the parameters of the *compressible linear layers* that is removed — the layers LSP substitutes, excluding embeddings, the language-model head and the normalizations — counted on the merged model with shared factors counted once. This is the denominator SVD-LLM, Swift-SVD and Dobi-SVD also report against, so nominal ratios are directly comparable. The measured-KL allocator greedily fills the budget and integer-trims its last move, and realized savings land within 0.25 points of nominal in every cell (-30.0–30.2, -50.0–50.2 and -70.0–70.2%).

Baselines. All baselines are using the same calibration budget of 1024 sequences of 2048 tokens, run with bf16 weights and compression arithmetic in fp32 and, for the SVD methods, are evaluated in factorized form. Where a baseline exposes a tuning knob we sweep it around the published value and keep the best per model and ratio: SVD-LLM's recovery learning rate against its single published one, Dobi-SVD's γ learning rate around the released default, Swift-SVD's allocation α over its own eleven-point grid, and ASVD's α. Dobi-SVD is run without its quantization step so that parameter counts match; Swift-SVD's OPT rows are bias-corrected; SVD-LLM's recovery stage is LoRA of rank 8 on the calibration corpus of the table it appears in. Notice that LSP keeps original weights frozen, and learns to compress their subspace, which means it does not move from the original subspace but only project. SVD-LLM-v2 (Wang et al., 2025c) and ModeGPT (Lin et al., 2025), though relevant, could not be reported as no official implementation has been released. The ViT ports of SVD-LLM (W) and SliceGPT are adapted from the official released implementation, while FLAR-SVD and PELA come from their official implementation. PELA, specifically, re-trains all the network's weights with the objective to copy the dense model's intermediate features, which acts as a recovery fine-tuning stage.

Evaluation. Perplexity follows the SparseGPT and SliceGPT convention: the test split is joined, tokenized once and cut into non-overlapping 2048-token windows. Zero-shot accuracy is measured with the EleutherAI harness (Gao et al., 2024a) at zero shots on the merged model, and we report plain accuracy.

Benchmarks. Choices of benchmarks is done following other compression methods in the literature. In 1 we report perplexity for a diverse mix of pruning and low-rank compression methods, as reported by Ashkboos et al. (2024); van der Ouderaa et al. (2024); Wang et al. (2025d;b); Qi et al. (2026). For zero-shot performance (2) we restrict the analysis to low-rank compression methods (Wang et al., 2025d;b; Qi et al., 2026), since they are the closest to LSP. Specifically, we report the same six benchmark evaluations as baselines do, and calibrate on same datasets. Specifically, while Alpaca-calibrated compression follows our compression ratios reported for other tables, for C4-calibrated compression we follow same procedure as Swift-SVD, with compression ratios of -20%/-40%/-60%, to introduce further richness in compression ratios. We re-evaluate all methods and benchmarks for evaluation fairness, using the same setup for calibration and evaluation for all methods.

C Initialization, Tying and Allocation

We ablate three choices: the spectral initialization, sharing a projector across a tied group, and allocating ranks from output KL. The initialization and tying ablations use a shape-based uniform allocation, so that within each table only the stated factor changes.

Uniform allocation. The uniform control assigns every compressed matrix the same retained fraction ρ of its dense parameter count, before accounting for ties:

$$r_\ell = \left\lfloor \rho \, \frac{d_{\text{in}} d_{\text{out}}}{d_{\text{in}} + d_{\text{out}}} \right\rfloor, \qquad k_\ell = d_\ell - r_\ell, \tag{19}$$

where d_ℓ is the projected dimension: $\min(d_{\text{in}}, d_{\text{out}})$ for an individual layer, d_{in} for an input-tied group and d_{out} for an output-tied group. A tied group uses $k_{\mathcal{G}} = \min_{\ell \in \mathcal{G}} k_\ell$, and ρ is set by binary search on the merged parameter count, shared factors counted once.

Initialization. Before training, neither start dominates (Table 6): the whitened truncation is better on OPT-1.3B and Llama-2-7B, the plain Gram on Qwen3-4B and on OPT-125M at -30 and -50%, and all cells are far from dense. The untrained ordering does not carry over to training: on OPT-125M, where the plain start leads by up to $2\times$ untrained, the whitened start ends about 2% lower in perplexity at every ratio (Table 7), and on Llama-2-7B it also stays ahead after training (Table 8). We therefore keep the whitened truncation.

Tying. Table 8 crosses initialization and tying on Llama-2-7B, with the tied and separate arms searched to the same parameter target (realized savings within 0.15 points). The whitened start leads under both tyings, by 1.9–3.0 Avg6 points and 20–33% in perplexity. Tying raises Avg6 at every ratio for both starts (0.3–3.1 points, most at -30%), while perplexity is mixed beyond -30%. A tied group stores its input factor once, so it keeps a higher rank at a fixed budget, and its shared latent is what the cache advantage of Section 3.3 rests on.

Allocation. Table 9 compares the measured-KL allocation on OPT-1.3B with three alternatives searched to the same realized savings: the uniform retained parameters of Equation (19), a uniform removed rank (the same fraction of directions removed from every unit), and a budget read off a diagonal Fisher of the calibration loss. Measured KL is best on both metrics at every ratio, by 0.2–0.9 perplexity and 0.7–1.0 Avg6 points over the best alternative.

Measured-KL against uniform allocation. Table 10 compares measured-KL and uniform allocation on zero-shot accuracy, for the models and calibration sets of Table 2. Measured allocation leads in mean accuracy in five of six settings, by up to 3.8 points (Qwen3-4B, -20%), and trails by 0.4 points on Llama-2-7B at -50%. Per benchmark it leads in 26 of 36 cells, including ARC-e and PIQA in every setting, while WinoGrande favours the uniform rule in four of six. On Qwen3-4B the

Table 6: **Initialization before training.** WikiText-2 test perplexity of the training-free truncation at uniform allocation, from the whitened truncation of Section 2.2 (*Whitened*, NoLSP at uniform allocation) or from the trailing directions of the plain activation Gram (*Plain*), removing the same number of directions per unit.

		-30%		-50%		-70%	
Model	**Dense**	**Whitened**	**Plain**	**Whitened**	**Plain**	**Whitened**	**Plain**
OPT-125M	27.6	582	**282**	664	**534**	**1,538**	2,211
OPT-1.3B	14.6	**26.1**	52.5	**74.4**	571	**814**	18,786
Qwen3-4B	7.9	40.1	**29.4**	255	**125**	7,186	**4,449**
Llama-2-7B	5.5	**9.2**	13.8	**22.1**	36.9	**172**	215

Table 7: **Initialization before and after training, OPT-125M.** WikiText-2 test perplexity (dense 27.6) of the training-free truncation and of LSP distilled from it, only the initialization changed.

	-30%		-50%		-70%	
OPT-125M	**Whitened**	**Plain**	**Whitened**	**Plain**	**Whitened**	**Plain**
Untrained	582	**282**	664	**534**	**1,538**	2,211
Trained (LSP)	**31.7**	32.3	**35.9**	36.6	**44.8**	45.8

Table 8: **Initialization and tying after training, Llama-2-7B.** LSP distilled on Alpaca, uniform removed rank per layer, the better of two learning rates per arm; WikiText-2 test perplexity and mean zero-shot accuracy (Avg6) of the validation-selected epoch.

		-30%		-50%		-70%	
Initialization	**Q/K/V, gate/up**	**PPL**	**Avg6**	**PPL**	**Avg6**	**PPL**	**Avg6**
Whitened	tied	**12.2**	**52.5**	21.8	**47.3**	**37.9**	**42.0**
Whitened	untied	14.8	50.4	**21.3**	47.0	38.2	41.5
Plain	tied	16.4	50.6	30.7	45.3	49.8	39.3
Plain	untied	22.2	47.5	27.9	44.0	47.6	38.8

Table 9: **Rank allocation, OPT-1.3B.** Four budgets at the same realized savings (within 0.05 points), whitened tied initialization, distilled on WikiText-2, the better of two learning rates per arm; WikiText-2 test perplexity and Avg6 of the validation-selected epoch.

	-30%		-50%		-70%	
Allocation	**PPL**	**Avg6**	**PPL**	**Avg6**	**PPL**	**Avg6**
Uniform retained parameters (Equation (19))	16.5	41.1	18.7	37.8	24.2	35.0
Uniform removed rank	16.8	40.6	18.9	37.6	23.4	35.4
Diagonal Fisher	16.7	41.4	18.6	38.1	25.6	34.5
Measured KL (**ours**)	**15.6**	**42.4**	**17.9**	**38.8**	**23.2**	**36.3**

Table 10: **Uniform against measured-KL allocation, zero-shot accuracy** (↑, %). *Uniform* is the retained fraction of Equation (19); *Measured KL* is the per-unit KL probe of Equation (3). Same whitened initialization, distillation objective (LSP) and realized budget; the Qwen3-4B pair also differs in K/V routing (Table 11). Llama-2-7B is calibrated on Alpaca and compressed by $-30/50/70\%$, Qwen3-4B on C4 by $-20/40/60\%$, as in Table 2.

Allocation	**Comp.**	**ARC-e**	**PIQA**	**Openb.**	**WinoG.**	**HellaS.**	**MathQA**	**Avg.**
Llama-2-7B (Alpaca calibration)								
Dense (Base)	–	75.6	77.8	32.8	69.9	57.1	28.1	56.9
Uniform	-30%	70.3	74.5	**30.6**	63.4	50.0	25.5	52.4
Measured KL		**71.2**	**75.4**	30.0	**64.0**	**50.4**	**26.4**	**52.9**
Uniform	-50%	64.3	70.6	**25.8**	**59.8**	**43.8**	**24.8**	**48.2**
Measured KL		**65.2**	**70.8**	24.6	58.6	43.4	23.9	47.8
Uniform	-70%	53.2	64.4	19.0	**53.4**	35.3	**22.1**	41.3
Measured KL		**55.6**	**66.1**	**21.8**	51.7	**36.1**	21.9	**42.2**
Qwen3-4B (C4 calibration)								
Dense (Base)	–	79.0	77.9	32.0	70.3	54.6	53.8	61.3
Uniform	-20%	74.1	74.4	26.0	64.2	49.6	32.1	53.4
Measured KL		**78.3**	**76.7**	**30.8**	**65.6**	**51.8**	**40.0**	**57.2**
Uniform	-40%	67.2	71.4	25.2	**61.8**	44.7	26.0	49.4
Measured KL		**70.6**	**72.3**	**28.8**	60.9	**45.4**	**26.5**	**50.7**
Uniform	-60%	55.2	67.8	20.6	**55.5**	**38.3**	22.6	43.3
Measured KL		**56.9**	**68.8**	**21.2**	55.3	37.3	**23.3**	**43.8**

measured runs also change the K/V routing, which Table 11 separates from the allocation. The gain comes at the cost of the probe (Section F), although one probe serves every ratio and both objectives; the effect on the deployed shape is given in Table 13.

K/V routing on Qwen3-4B. Table 11 crosses the two allocations with forced Q/K/V input tying and with the grouped-query routing (K and V share an output-side factor, Q is projected alone), at fixed objective, learning rate, epoch budget and realized savings. Under the uniform allocation, routing changes Avg6 by at most 0.6 points and narrows the cache by about 8%. Under the measured

allocation, the output-side routing is ahead at every ratio (0.5–1.6 Avg6 points, 2–5% in perplexity), but the probe leaves K and V almost uncompressed, so the cache is as wide as the dense model's at -20 and -40%. At matched input tying, measured allocation leads by 3.0 Avg6 points at -20% and trails by 0.1 and 0.4 at -40 and -60%. The wide measured cache follows from the probe: per parameter saved, truncating the K/V pair costs about an order of magnitude more KL than any other unit type, so greedy allocation buys it last, whereas the uniform rule compresses every unit.

Table 11: **Allocation and K/V routing, Qwen3-4B.** LSP distilled on C4, same learning rate, epoch budget and realized savings; gate/up are tied in every arm. *Q/K/V, input*: one shared input projection for Q, K and V. *K/V, output*: the grouped-query routing, with K and V sharing an output-side factor and Q projected alone. The measured budget uses the $i/8$ probe grid here instead of the $i/16$ grid of the main tables, so its K/V-output row is not the Measured cell of Table 10; the uniform Q/K/V row is that table's Uniform cell. Avg6 and C4 test perplexity of the validation-selected epoch; KV is the floats cached per token per layer (dense 2048). Best per column in bold.

		-20%			-40%			-60%		
Allocation	**Tied group**	**Avg6**	**PPL**	**KV**	**Avg6**	**PPL**	**KV**	**Avg6**	**PPL**	**KV**
Uniform	Q/K/V, input	53.4	15.9	1357	49.4	18.4	1018	43.3	23.0	679
Uniform	K/V, output	54.0	16.0	**1242**	49.3	18.6	**932**	42.9	23.6	**622**
Measured KL	Q/K/V, input	56.4	15.0	1822	49.3	17.9	1701	42.9	23.2	1282
Measured KL	K/V, output	**56.9**	**14.7**	2048	**50.9**	**17.1**	2048	**44.2**	**22.4**	1920

D WHAT LSP COMPRESSES, ACROSS MODELS

Figure 5 repeats the allocation measurement of Section 3.4 on the measured-KL checkpoints of OPT-125M, OPT-1.3B, Qwen3-4B and the CIFAR-100 ViT-B/16, and Figure 6 repeats its spectral measurement on single blocks of OPT-1.3B and Llama-2-7B.

Where the budget goes. On OPT-125M, Q/K/V blocks are the least compressed at all compression budgets, especially latest layers, which stay uncompressed at all compression ratios. For both OPT models, latest layers tend to be less compressed than earlier layers. For Qwen3-4B, an interesting behavior appears: because of non-tying Q and KV together, Q becomes the most compressed layer at all compression ratios, while KV layers almost never compressed (only for few early layer, at -70% compression). In addition MLP layer are also compressed less, especially the down-projection one. The ViT instead shows a pattern where middle layers gets compressed less, at all compression ratios.

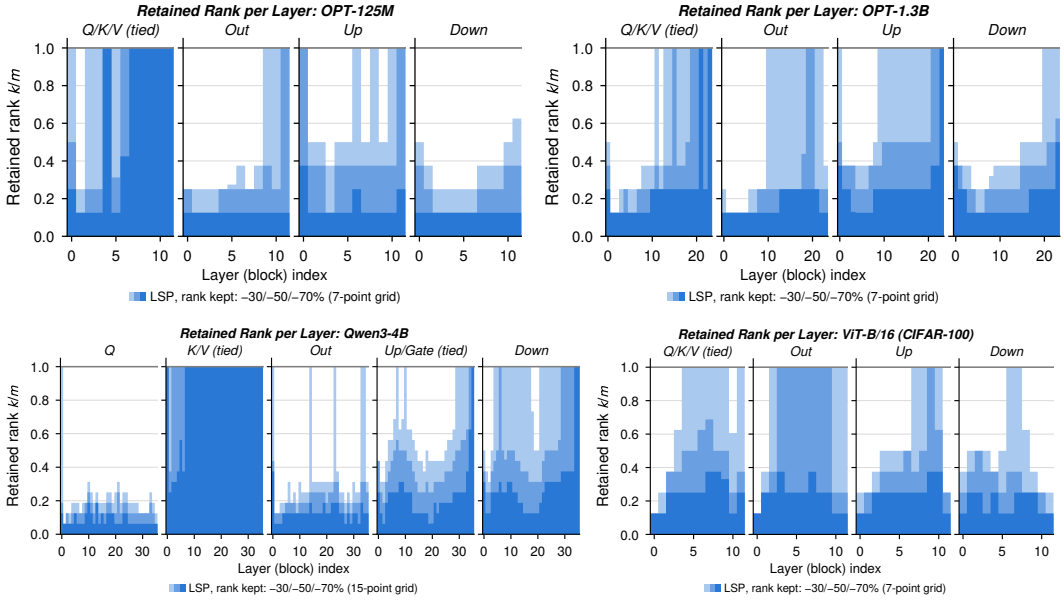


Figure 5: **Rank allocation across models.** Retained rank k as a fraction of the full matrix rank m, by block and projection type, for the measured-KL LSP checkpoints of OPT-125M, OPT-1.3B, Qwen3-4B and the CIFAR-100 ViT-B/16; the same panel for Llama-2-7B is Figure 4 (left). Shades stack the three ratios, -30% lightest behind and -70% darkest in front, so each shade's top edge is that ratio's profile wherever the allocations nest; $k/m = 1$ is a unit the allocator left dense. The probe grid is 7 points on OPT-125M, OPT-1.3B and the ViT and 15 on Qwen3-4B. Qwen3-4B has grouped-query attention, so Q is projected alone and K and V share an output-side factor, and each gets its own panel.

What the removal looks like, block by block. Figure 6 repeats the two spectral panels of Figure 4 for an early, a middle and a late block of OPT-1.3B (blocks 3, 12 and 21 of 24) and Llama-2-7B (blocks 4, 16 and 28 of 32). Removal looks as in Llama's average: it spreads over the whole spectrum rather than the tail a weight SVD would cut, deepens with the ratio, and reaches the first singular direction, most strongly in the up projection on OPT-1.3B (direction 1 loses 2.5–4.4 of its amplitude in block 3) and in the tied Q/K/V on Llama-2-7B (0.7–2.3). Deeper blocks are spared more: OPT-1.3B's block 21 keeps Q/K/V dense at every ratio and its other projections almost intact at -30%, and on Llama-2-7B block 16 keeps its Q/K/V, output and down projections dense at -30% and block 28 its up and down projections. The difference to NoLSP is small in every block, so the thin difference of the averaged measurement is not an averaging artifact: at most 2.5% of the removal in the Llama-2-7B blocks shown (median over all compressed units 1.0%), and on OPT-1.3B growing with depth, from 1–2% in block 3 to 5.8% in the down projection of block 21 (median 1.6%). Its sign pattern holds across depth: LSP removes more of the leading quarter of directions and keeps more of the trailing half than NoLSP in 171 of the 180 compressed OPT-1.3B units and 244 of the 248 Llama-2-7B units at -50% and -70%.

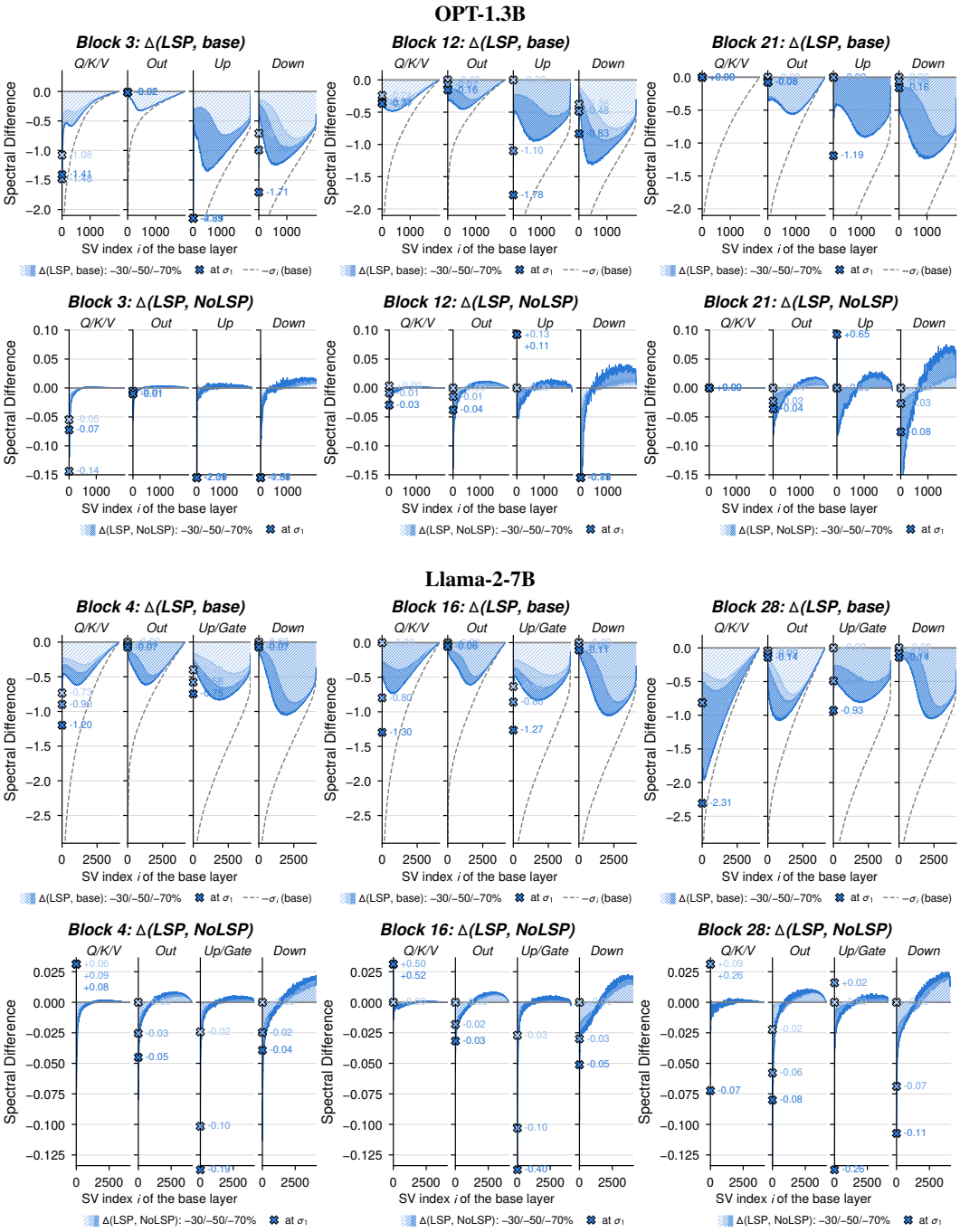


Figure 6: **Spectral removal in single blocks.** The two spectral panels of Figure 4 for an early, a middle and a late block (columns) of OPT-1.3B (blocks 3, 12, 21 of 24, top) and Llama-2-7B (blocks 4, 16, 28 of 32, bottom), on the measured-KL checkpoints of Table 1. For each model, the first row is the amplitude lost along each original singular direction, $\Delta_i = \sigma_i(\|\boldsymbol{Q}^\top \boldsymbol{w}_i\|_2 - 1)$, with $-\sigma_i$ of the uncompressed layer dashed, and the second the difference to NoLSP, $\sigma_i(a_i^{LSP} - a_i^{\text{init}})$, at the same KL-probe ranks. Each row shares one vertical axis. Shades are the $-30/50/70\%$ runs (light to dark), the bands are a running mean over 9 directions, a missing band is a unit left dense at that ratio, and tied Q/K/V and up/gate panels average their member matrices; crosses give the raw value on direction 1, parked on the axis edge when outside it.

E INFERENCE EFFICIENCY

Table 12 extends Table 4 to all three ratios on the WikiText-2-calibrated Llama-2-7B checkpoints. Throughout this section, throughput is measured on one GH200 in bf16 with SDPA and CUDA-graph decoding, at batch 1 and with a full-width KV cache, while latent-cache widths and context capacities are analytical estimates computed from the factor widths, assuming a latent cache (Section A.6) and counting persistent tensors only; context capacities assume a 95.5 GiB device with 6 GiB headroom. At weights and FLOPs matched to within 2%, LSP decodes fastest at every ratio, 15–26% ahead of the best per-matrix factorization at 512 tokens, and its latent cache is 1.2×, 1.9× and 2.2× narrower. Every per-matrix factorization decodes slower than the dense model at −30%, while LSP is faster than dense at every ratio (1.21×, 1.36× and 1.56× at −30%, −50% and −70%). This margin is a short-context one: it falls to 1.13× at 16k tokens, where reading the cache dominates.

Allocation at deployment. Table 13 compares the uniform and measured-KL allocations on the same checkpoints; at a given ratio both hold the same weights (8.93, 6.52 and 4.11 GiB) and decode FLOPs. The measured budget decodes faster at every ratio (10, 17 and 6%), partly because a unit it leaves dense runs as one matrix multiply instead of two. Its cache is 38% wider at −30%, where a few Q/K/V groups stay uncompressed and cache full-width K and V, but 8 and 22% narrower at −50 and −70%, where it compresses attention harder. Both allocations hold longer contexts and decode faster than every per-matrix factorization of Table 12, so the advantage over them does not depend on the allocation.

K/V routing at deployment. Table 14 gives the deployment cost of the four Qwen3-4B arms of Table 11. Within a ratio they hold the same weights (6.14, 4.78 and 3.43 GiB) and issue the same decode FLOPs (7.80, 6.35 and 4.89 GFLOP per step). The allocation sets the speed: the measured budget is faster than the uniform one in all six pairs (22, 10 and 8% at −20%, −40% and −60%) because it leaves units dense, whereas the uniform rule factorizes every layer and decodes slower than the dense model at −20 and −40%. Input tying is 2–4% faster than the output-side routing in all six pairs, since one shared input factor serves Q, K and V in a single matrix multiply. The routing mainly decides the cache, in a direction set by the allocation: it narrows the uniform cache by about 8% and widens the measured one to or near the dense width. No configuration wins on

Table 12: Inference efficiency of compressed Llama-2-7B at every ratio, the checkpoints of Table 1, on one GH200 (bf16, SDPA). *Memory*, analytic from the checkpoints' ranks (factors vs. dense): weights, floats cached per token per layer with the latent KV cache ($\boldsymbol{z} = \boldsymbol{A}\boldsymbol{x}$; a unit the measured-KL allocation leaves uncompressed stays dense and caches full-width K/V), weights plus KV cache at 128k tokens and batch 8, and the longest context that fits 95.5 GiB at batch 8. *Speed*: FLOPs to prefill 2k tokens and per decode step at 2k context, and measured CUDA-graph-compiled decode tok/s at batch 1 for 512, 4k and 16k tokens of context. Best factorized method per ratio in bold.

	Memory				Compute and speed (batch 1)				
Method	**Weights (GiB)**	**KV floats /token/layer**	**Weights+KV @128k, b8 (GiB)**	**Max ctx (k tok, b8)**	**Prefill TFLOPs @2k**	**Decode GFLOP/step**	**Decode tok/s @512**	**@4k**	**@16k**
Llama-2-7B (dense)	12.55	8192	524.6	19.7	29.26	14.29	141	85	36
−30%									
SVD-LLM (W)	8.93	2866 (2.9×)	188.1 (2.8×)	59.0	21.30	10.40	135	83	35
Swift-SVD	8.93	2866 (2.9×)	188.1 (2.8×)	59.0	21.30	10.40	137	83	36
Dobi-SVD	9.06	3279 (2.5×)	214.0 (2.5×)	51.4	21.59	10.54	136	83	35
LSP (Ours)	8.93	**2316** (3.5×)	**153.7** (3.4×)	**73.0**	21.30	10.40	**172**	**95**	**37**
−50%									
SVD-LLM (W)	6.52	2048 (4.0×)	134.5 (3.9×)	85.0	16.00	7.81	163	92	37
Swift-SVD	6.52	2048 (4.0×)	134.5 (3.9×)	85.0	16.00	7.81	164	92	37
Dobi-SVD	6.53	2060 (4.0×)	135.2 (3.9×)	84.5	16.01	7.82	164	93	37
LSP (Ours)	6.52	**1096** (7.5×)	**75.0** (7.0×)	**158.8**	16.00	7.81	**193**	**101**	**38**
−70%									
SVD-LLM (W)	4.11	1228 (6.7×)	80.9 (6.5×)	145.8	10.69	5.22	191	100	38
Swift-SVD	4.11	1228 (6.7×)	80.9 (6.5×)	145.8	10.69	5.22	193	101	38
Dobi-SVD	4.12	1434 (5.7×)	93.7 (5.6×)	124.9	10.71	5.23	189	99	38
LSP (Ours)	4.11	**556** (14.7×)	**38.9** (13.5×)	**322.1**	10.69	5.22	**221**	**108**	**40**

Table 13: **Deployment cost of the allocation, Llama-2-7B.** The uniform and measured-KL allocations compared in Table 10, here on the WikiText-2-calibrated checkpoints, on one GH200, Q/K/V tied on the input side. *KV*: floats cached per token per layer, and *Max ctx*: the longest context (k tokens) that fits 95.5 GiB at batch 8, both analytic from the checkpoints' ranks, counting a unit the allocator left uncompressed as caching full-width K,V. *tok/s*: measured CUDA-graph-compiled decode throughput at batch 1 and 512 tokens of context, both allocations in the same session, best of two passes. Within a ratio both hold the same weights and issue the same decode FLOPs.

	-30%			-50%			-70%		
Allocation	**KV**	**Max ctx**	**tok/s**	**KV**	**Max ctx**	**tok/s**	**KV**	**Max ctx**	**tok/s**
Llama-2-7B (dense)	8192	19.7	142	8192	19.7	142	8192	19.7	141
Uniform	**1673**	**101.0**	157	1195	145.6	165	717	249.8	209
Measured KL	2316	73.0	**172**	**1096**	**158.8**	**193**	**556**	**322.1**	**221**

all three axes: uniform with output-side K/V holds the longest context but is the slowest, measured with input tying is the fastest, and measured with output-side K/V is the most accurate but caches nearly as much as the dense model.

Measured-KL allocation with K/V tied on the output side is the most accurate configuration at every ratio (Table 11), so we use it as the default. When deployment efficiency matters more, LSP supports the other trade-offs: uniform allocation with output-side K/V holds the longest context, and measured allocation with input-side tying decodes fastest.

Table 14: **Deployment cost of allocation and K/V routing, Qwen3-4B.** The four arms of Table 11 on one GH200. *KV*: floats cached per token per layer, and *Max ctx*: the longest context that fits 95.5 GiB at batch 8, both analytic from the merged checkpoints' ranks, counting a unit the allocator left uncompressed as caching full-width K,V. *tok/s*: measured CUDA-graph-compiled decode throughput at batch 1 and 512 tokens of context, best of two passes. Within a ratio all four arms hold the same weights and issue the same decode FLOPs, given in the text.

		-20%				-40%				-60%			
Allocation	**Tied group**	**KV**	**Max ctx**	**tok/s**	**Avg6**	**KV**	**Max ctx**	**tok/s**	**Avg6**	**KV**	**Max ctx**	**tok/s**	**Avg6**
Qwen3-4B (dense)		2048	74.6	175	61.3	2048	74.6	175	61.3	2048	74.6	175	61.3
Uniform	Q/K/V, input	1357	114.5	147	53.4	1018	155.1	170	49.4	679	236.3	186	43.3
Uniform	K/V, output	**1242**	**125.1**	144	54.0	**932**	**169.4**	165	49.3	**622**	**257.9**	179	42.9
Measured KL	Q/K/V, input	1822	85.3	**180**	56.4	1701	92.8	**187**	49.3	1282	125.2	**201**	42.9
Measured KL	K/V, output	2048	75.9	176	**56.9**	2048	77.1	182	**50.9**	1920	83.6	193	**44.2**

F Compression Cost

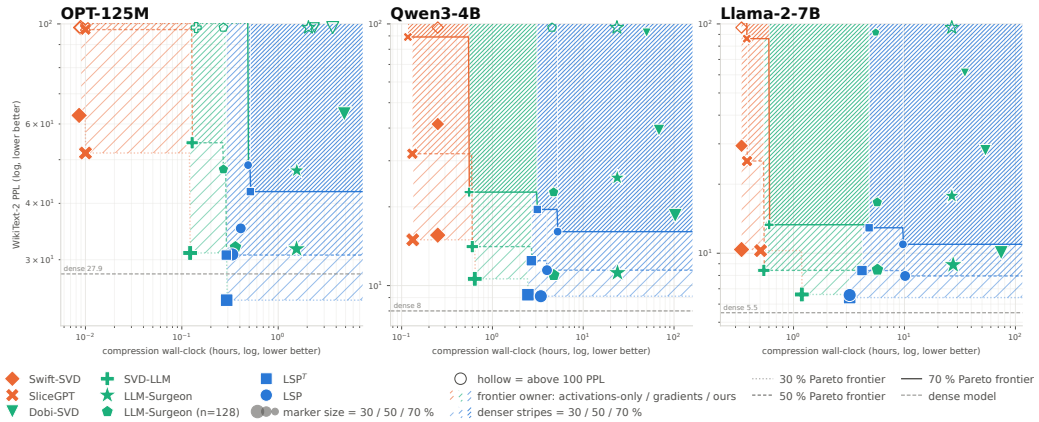


Figure 7: **Compression cost with measured-KL allocation on three LLMs.** WikiText-2 perplexity at $-30/50/70\%$ compression (marker size) against wall-clock hours. LSP costs include initialization, training through the selected epoch, and a full KL probe on the 7-point grid on OPT-125M and the 15-point grid on Qwen3-4B and Llama-2-7B.

Figure 7 compares LSP's compression cost with that of activation-based SVD methods (SVD-LLM, Swift-SVD), loss-aware ones (Dobi-SVD, LLM-Surgeon) and the structured pruning methods SliceGPT and LLM-Surgeon, on OPT-125M, Qwen3-4B and Llama-2-7B. Giving the loss-aware baselines more budget does not close the gap. LLM-Surgeon calibrated on LSP's 1024 sequences costs 4–$6\times$ more than at its default 128, and its perplexity does not improve. Dobi-SVD, trained for 20 epochs on the same 1024 samples, costs 2–103 wall-clock hours and stays far behind: 61.2 against 10.9 perplexity on Llama-2-7B at -70%.

Compression cost is paid once per model and ratio, and it is small next to the cost of serving the compressed model. We argue it should therefore not drive the choice between methods, whereas accuracy and inference efficiency (Section 3.3) are paid on every query. Even so, LSP is Pareto-optimal at higher budgets: at every model–ratio setting, one of its two objectives gives the lowest perplexity, within 0.5–10 wall-clock hours. Only methods that are cheaper and less accurate share the frontier: SVD-LLM with LoRA recovery, SliceGPT and Swift-SVD. LSP ^{T} converges faster than LSP, reaching its validation optimum at up to $2.4\times$ lower cost (4.1 against 10.1 hours on Llama-2-7B at -50%), though at a higher perplexity at -50 and -70%. The probe is a substantial fixed cost: 0.17, 1.74 and 1.11 wall-clock hours on OPT-125M, Qwen3-4B and Llama-2-7B (0.17, 6.9 and 4.4 GPU-hours). The LSP points use individual-run perplexities, which can differ from the three-seed means of Table 1.

G Transfer on the Growing Calibration Pool

Table 15 breaks the ViT transfer results down of Table 3 by calibration pool and target. The pools grow from CIFAR-100 alone (10k images) to the six-dataset pool of Table 3 (47k), with the same images for every method. LSP's mean transfer rises with every added dataset at every ratio, and on the full pool it is best on all three targets at every ratio; the baselines gain less and not always monotonically (FLAR-SVD at 50% loses 4.4 points when CIFAR-10 and EuroSAT are added). The last row of each method is CIFAR-100 alone at the size of the full pool. For LSP, this extra CIFAR-100 data changes mean transfer by $+1.2$, -0.6 and $+0.8$ points at $-30/50/70\%$, whereas the diverse pool exceeds this size-matched control by 3.7, 7.1 and 6.6 points (the Gain of Table 3): the gain comes from pool composition, not size.

Table 15: Complete downstream-transfer results for compressed ViT-B/16, for every calibration pool, method and target. *Compressed on*: the datasets in the pool (✓) and its size, the same pools for every method; ✓† is CIFAR-100 alone at the size of the six-dataset pool. *Transfer to*: linear-probe accuracy on Pets, Aircraft and Places365, absent from every pool, and their mean; *Source*: CIFAR-100 accuracy through the original head. LSP ^{T} needs CIFAR-100 labels. Best per column, pool and ratio in **bold**.

	Compressed on							Transfer to				Source
Method	CIFAR-100	Food-101	CIFAR-10	EuroSAT	STL-10	DTD	Images	Pets	Aircraft	Places365	Mean	CIFAR-100
Dense (Base)				—				71.9	36.9	33.5	47.4	89.9
-30% parameters												
FLAR-SVD	✓						10k	**67.1**	32.3	**31.2**	**43.6**	88.7
	✓	✓					20k	68.6	34.1	**32.2**	45.0	88.8
	✓	✓	✓	✓			40k	68.6	33.2	**32.7**	44.9	88.7
	✓	✓	✓	✓	✓	✓	47k	68.4	33.3	32.4	44.7	88.6
	✓†						47k	66.9	32.7	**31.1**	43.6	88.8
SVD-LLM (W)	✓						10k	63.2	31.1	29.5	41.2	88.2
	✓	✓					20k	63.3	32.3	28.8	41.5	87.9
	✓	✓	✓	✓			40k	64.4	32.2	29.2	41.9	87.8
	✓	✓	✓	✓	✓	✓	47k	64.7	32.6	29.5	42.3	87.7
	✓†						47k	63.0	31.2	29.2	41.2	88.2
PELA	✓						10k	66.2	**32.9**	30.1	43.1	89.2
	✓	✓					20k	65.0	32.2	30.3	42.5	89.1
	✓	✓	✓	✓			40k	67.3	34.1	30.1	43.8	88.9
	✓	✓	✓	✓	✓	✓	47k	67.1	34.2	30.5	43.9	88.9
	✓†						47k	66.0	31.1	29.8	42.3	89.2
LSP^{T} (Ours)	✓†						47k	**67.8**	**33.9**	30.8	**44.2**	**89.4**
LSP (Ours)	✓						10k	66.0	32.4	29.8	42.7	**89.3**
	✓	✓					20k	**70.6**	**36.8**	32.1	**46.5**	**89.3**
	✓	✓	✓	✓			40k	**71.3**	**37.2**	32.0	**46.8**	**89.2**
	✓	✓	✓	✓	✓	✓	47k	**71.7**	**37.7**	**33.2**	**47.5**	**89.2**
	✓†						47k	67.3	33.8	30.5	43.9	89.3
-50% parameters												
FLAR-SVD	✓						10k	58.1	27.2	**28.7**	38.0	86.2
	✓	✓					20k	61.1	30.8	**29.7**	40.5	86.1
	✓	✓	✓	✓			40k	54.3	27.0	27.1	36.1	83.5
	✓	✓	✓	✓	✓	✓	47k	54.9	26.9	26.7	36.2	83.1
	✓†						47k	58.6	27.5	**28.3**	38.1	86.4
SVD-LLM (W)	✓						10k	58.6	28.3	26.6	37.8	86.1
	✓	✓					20k	59.6	29.1	27.0	38.6	84.9
	✓	✓	✓	✓			40k	59.6	28.9	27.2	38.6	84.4
	✓	✓	✓	✓	✓	✓	47k	60.1	28.4	27.1	38.6	84.1
	✓†						47k	58.4	28.7	27.2	38.1	86.2
PELA	✓						10k	**62.6**	**29.7**	28.5	**40.3**	87.9
	✓	✓					20k	61.6	30.9	28.1	40.2	**87.7**
	✓	✓	✓	✓			40k	64.6	31.3	29.3	41.7	87.0
	✓	✓	✓	✓	✓	✓	47k	63.6	32.3	29.3	41.8	**87.8**
	✓†						47k	**62.0**	**30.0**	28.2	**40.1**	**88.5**
LSP^{T} (Ours)	✓†						47k	56.0	23.7	25.7	35.1	88.0
LSP (Ours)	✓						10k	58.9	28.1	27.5	38.2	**88.2**
	✓	✓					20k	**63.2**	**31.7**	28.5	**41.1**	87.5
	✓	✓	✓	✓			40k	**67.5**	**34.4**	**30.4**	**44.1**	**88.0**
	✓	✓	✓	✓	✓	✓	47k	**68.2**	**35.2**	**30.7**	**44.7**	87.7
	✓†						47k	58.7	26.9	27.3	37.6	88.3
-70% parameters												
FLAR-SVD	✓						10k	37.7	18.5	22.7	26.3	70.7
	✓	✓					20k	40.5	19.3	23.7	27.8	68.0
	✓	✓	✓	✓			40k	41.2	20.0	23.3	28.1	66.9
	✓	✓	✓	✓	✓	✓	47k	40.2	19.8	23.6	27.9	64.6
	✓†						47k	38.5	18.9	22.5	26.6	71.5
SVD-LLM (W)	✓						10k	47.9	**23.2**	23.9	31.7	77.2
	✓	✓					20k	50.9	24.6	24.0	33.2	70.5
	✓	✓	✓	✓			40k	50.5	23.7	24.1	32.8	68.2
	✓	✓	✓	✓	✓	✓	47k	51.9	24.0	24.4	33.4	66.2
	✓†						47k	48.2	**23.0**	23.9	31.7	77.1
PELA	✓						10k	**53.1**	22.4	**27.1**	**34.2**	84.4
	✓	✓					20k	54.3	24.3	**27.5**	35.4	**84.0**
	✓	✓	✓	✓			40k	56.2	26.3	28.1	36.9	83.0
	✓	✓	✓	✓	✓	✓	47k	58.9	27.8	28.2	38.3	82.8
	✓†						47k	**53.0**	22.3	**26.9**	**34.1**	85.3
LSP^{T} (Ours)	✓†						47k	46.4	20.3	22.7	29.8	84.2
LSP (Ours)	✓						10k	49.7	22.2	24.8	32.2	**85.8**
	✓	✓					20k	**57.1**	**27.6**	27.2	**37.3**	83.6
	✓	✓	✓	✓			40k	**58.6**	**26.9**	**29.6**	**38.3**	**83.9**
	✓	✓	✓	✓	✓	✓	47k	**60.4**	**29.1**	**29.4**	**39.6**	**84.6**
	✓†						47k	49.4	22.9	26.6	33.0	**85.9**

H REPRODUCIBILITY STATEMENT

Code will be published after acceptance.