RECOMP: LOW-RANK RESIDUAL COMPENSATION FOR CONTEXTUAL SPARSITY IN LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Contextual sparsity, or dynamically computing a subset of neurons during inference, reduces multilayer perceptron (MLP) cost in Large Language Models (LLMs), but discards the output of the rest of the neurons, which is individually small but collectively substantial. We show that this output is predictable rather than lost, and evaluate whether part of a fixed compute budget is better spent on reconstructing it than computing more neurons. For a given token t at layer ℓ, ReComp uses a low-rank neuron-importance predictor that selects the top-k neurons to compute by importance. Then, a low-rank affine map estimates the combined output of the skipped neurons from the MLP input, fit via reduced-rank ridge regression (ridge-RRR) and distilled against the dense model with the objective having a KL-divergence term and an optional Jacobian-lens term. Across five models from 0.5B to 14B parameters, reconstruction beats spending the same amount of computation on more neurons at every budget we test with our selector. For our predictor, the compensator helps even before distillation. After distillation, it also improves other selectors; for example, when added to TEAL, it lowers relative perplexity on Qwen2.5-7B from 1.42 to 1.16 at the same arithmetic. At 25% of compressed-layer MLP arithmetic, ReComp lowers relative WikiText-2 perplexity on Qwen2.5-7B from 1.44 for the best of TEAL, WINA, and R-Sparse to 1.16, and beats all three on every 7-8B model tested.

1 INTRODUCTION

Large Language Models (LLMs) built on the Transformer Architecture (Vaswani et al., 2017) have shown remarkable performance in natural language processing, particularly in reasoning, text generation, translation, code generation, and complex problem-solving skills. However, the computational cost of transformers has garnered significant attention, often requiring large, GPU-dependent data centers for training and inference, primarily due to the large amount of memory and computation needed to run the multilayer perceptron (MLP) block, and due to the vast size of the Key-Value (KV) cache for large contexts. Static compression, such as weight pruning (Frantar & Alistarh, 2023; Sun et al., 2024b), quantization (Xiao et al., 2023; Dettmers et al., 2022) and low-rank factorization (Yuan et al., 2025; Wang et al., 2025), applies the same reduced computation to every token, even though which neurons matter depends strongly on the input.

Contextual sparsity, or dynamically computing a subset of neurons during inference, has been a promising area of research, as it exploits the observation that, for a particular input context and layer, only a small subset (dependent on the input) of MLP neurons may be needed to preserve the behavior of the dense model. Prior work such as DejaVu (Liu et al., 2023) and ShadowLLM (Akhauri et al., 2024) have largely focused on using lightweight "lookahead" predictors to guess which components will be needed for the next layer while the current layer is being computed. However, accurately predicting the set of important neurons remains difficult, and small errors in neuron ranking can accumulate across layers and substantially degrade language quality. Additionally, research such as TEAL (Liu et al., 2025) and CATS (Lee et al., 2024) compute activation magnitudes or thresholds, dependent on context, to identify specific neurons for selective computation without training a predictor. Even though they avoid computational overhead for training a predictor, they still discard the contribution of neurons that are not selected. This is especially damaging during inference because even when the selected set captures the largest individual activations, the omitted neurons can collectively make a large contribution to the MLP output. On Qwen2.5-7B, computing a quarter of

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.

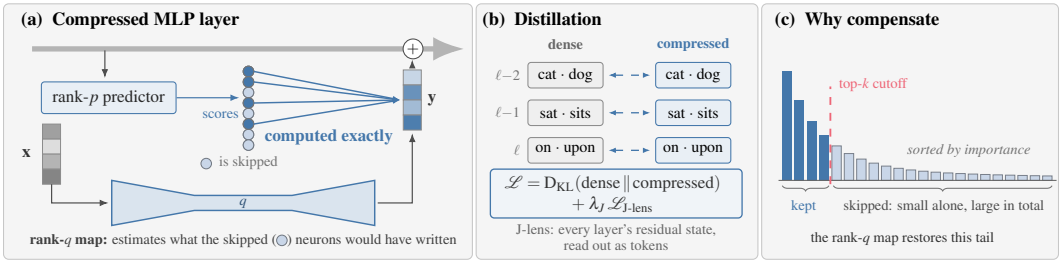

Figure 1: **Overview of ReComp.** (a) A rank-p predictor selects the top-k neurons, and a rank-q affine map estimates the combined output of the rest. (b) The compensators are distilled against the dense model (KL plus an optional J-lens term) with all other weights frozen. (c) Skipped neurons are individually small but collectively large.

each compressed MLP with TEAL and dropping the rest raises perplexity by 55%; ReComp, at the same arithmetic, raises it by 16% (Section 4.3).

We propose ReComp (Residual Compensation), which characterizes the output of uncomputed MLP neurons as a semi-predictable signal and not an irrecoverable error, unlike prior research. A lightweight low-rank predictor selects neurons for computation, and a low-rank affine map estimates the rest. We then distill the compensators against the dense model while keeping all pretrained weights frozen. Because the predictor, compensator, and selected neurons share one budget, we ask **whether a unit of computation is better spent on another neuron or on reconstruction**, and we find that the answer depends on the selector. We compare methods by reporting analytical FLOPs; measured latency is reported only for a single-layer kernel (Appendix A.6). Our main contributions can be summarized as follows:

- **Budgeted residual compensation:** Unlike prior contextual-sparsity methods, which discard the output of skipped neurons, ReComp reconstructs it with a rank-q affine compensator that is budgeted the same as the neurons it replaces. With our selector, a unit of computation is much better spent on reconstruction than on another neuron; reconstruction wins in all 20 model budget settings we test, with the largest gains at the sparsest budgets. The distilled compensator also works as an attachment for a training-free selector, lowering TEAL's relative perplexity from 1.42 to 1.16 on Qwen2.5-7B at the same amount of computation (Section 4.4).
- **Selector-based closed-form fitting:** Our compensator is a reduced-rank ridge regression of the output that one specific selector removes at one specific budget, and is computed in a single streaming pass, instead of static low-rank weight factorizations. We show that the selector decides whether this fit works without any training: on every model, it helps our neuron-level predictor (2.35 to 1.36 relative PPL on Qwen2.5-7B) but degrades a magnitude threshold, and on the same selector it beats a LoRA-initialized adapter matched in parameters, computation, and training.
- **Empirical Results:** At 25% density, ReComp outperforms TEAL, WINA and R-Sparse in perplexity on three 7-8B models, and its compensator does so before distillation (its predictor is trained). At 35-50% computation, the training-free methods are better in downstream accuracy and text generation, and we find that this discrepancy is due more to **selection** than to budget.

2 Related Work

Contextual activation sparsity: DejaVu (Liu et al., 2023) and ShadowLLM (Akhauri et al., 2024) train predictors that decide which neurons or attention heads to execute, and PowerInfer (Song et al., 2024) and LLM in a Flash (Alizadeh et al., 2024) use activation sparsity to reduce computation and memory traffic on constrained hardware. Methods not requiring training select from activations, with methods such as CATS (Lee et al., 2024) that threshold gate activations, TEAL (Liu et al., 2025) applying sparsification to hidden states based on their magnitudes, and WINA adding weight

information to the selection score (Chen et al., 2026). A separate line of work sparsifies the model through training, for example by converting MLPs into experts (Zhang et al., 2022), or by increasing activation sparsity through continued training (Song et al., 2025). These methods discard the output of unselected neurons. ReComp keeps a fixed selector and checks whether part of the budget should instead reconstruct that discarded output.

Sparse computation with low-rank recovery: R-Sparse (Zhang et al., 2025) is closest to ReComp as it uses activation sparsity and low-rank approximations of the projection weights and applies it to uncomputed activation components. Instead, ReComp uses regression on the removed output of the MLP from its input by using targets defined by a selector at a specific budget. Static sparsity and low-rank weight decompositions (Zhang & Papyan, 2025; Mozaffari et al., 2025; Makni et al., 2025) similarly use both a sparse component with a low-rank remainder, but the decomposition is fixed across tokens. RaNA (Garcia et al., 2025) learns adaptive-rank representations for transformer compression.

Low-rank compression and adapter adaptation: SVD methods that use activations and outputs as the basis for computation add low-rank weight factors to the distribution on which the weights operate (Yuan et al., 2025; Wang et al., 2025; Hsu et al., 2022; Abbasi et al., 2026). Adapter initializations such as PiSSA (Meng et al., 2025) start low-rank adapters from **principal directions** instead of random values. We additionally compare our method with a LoRA (Hu et al., 2022) adapter.

Low-rank error compensation: Several methods correct a static compression step with a low-rank term to fit the error that is introduced. For example, LoRC in ZeroQuant-V2 (Yao et al., 2023) and LQER (Zhang et al., 2024) for quantization, and EoRA (Liu et al., 2026) for quantization and pruning. ReComp applies the same concept to contextual sparsity, where the error changes as every new token is generated.

3 RECOMP

3.1 PRELIMINARIES

Let $\mathscr{M}$ be a L-layer decoder-only Transformer (Vaswani et al., 2017). For token t at layer ℓ, write $x_\ell^{(t)} \in \mathbb{R}^d$ for the MLP input after normalization and $y_\ell^{(t)} = \mathrm{MLP}_\ell(x_\ell^{(t)}) \in \mathbb{R}^d$ for the MLP write.

For ReComp, the Multilayer Perceptron (MLP) is in a form that is common to current Language Model architectures, $\mathrm{MLP}_\ell(x) = \phi_\ell(x) W_{\text{down}}^{(\ell)} + b_{\text{out}}^{(\ell)}$, $\phi_\ell : \mathbb{R}^d \to \mathbb{R}^m$, where the hidden map ϕ_ℓ is either:

$$\sigma\left(xW_{\text{in}}^{(\ell)} + b_{\text{in}}^{(\ell)}\right) \text{ (GPT-2, OPT and Pythia) or}$$

$$\sigma\left(xW_{\text{gate}}^{(\ell)}\right) \odot \left(xW_{\text{up}}^{(\ell)}\right) \text{ (Llama, Mistral, Gemma and Qwen)}$$

with $W_{\text{gate}}^{(\ell)}, W_{\text{up}}^{(\ell)} \in \mathbb{R}^{d\times m}$, $W_{\text{down}}^{(\ell)} \in \mathbb{R}^{m\times d}$. When writing $W_{\text{down},j}$ for row j, this decomposes into m neuron writes, which can be denoted as $y = \phi_1(x)W_{\text{down},1} + \phi_2(x)W_{\text{down},2} + \ldots + \phi_m(x)W_{\text{down},m} = \sum_{j=1}^m \phi_j(x) W_{\text{down},j}$..

A contextual sparsity method usually picks a gate $g(x) \in \{0,1\}^m$, where $g_j = 1$ means to compute the neuron, while $g_j = 0$ skips it, choosing only k neurons: $S = \{j : g_j = 1\}$, $|S| = k$. This returns the calculation of only those k selected neurons, or $\hat{y} = \sum_{j\in S} \phi_j(x) W_{\text{down},j}$.

For our method, this sparsity is only applied to a contiguous middle section of the Transformer's layers, $\ell \in [\lfloor 0.25L \rfloor, \lfloor 0.90L \rfloor)$, and the MLP is run densely for the first $T_0 = 16$ token positions of a given sequence.

3.2 WHY THE SKIPPED WRITE IS NOT NOISE

We define the **oracle importance** of a given neuron as $s_j(x) = |\phi_j(x)| \cdot \|W_{\text{down},j}\|_2$, or the L2 norm of neuron j's contribution to the output y. Only a small number of neurons have large individual contributions to y, which is the reasoning behind every activation-sparsity method (Liu et al., 2023;

Song et al., 2024; Alizadeh et al., 2024). However, these methods differ in what happens to the skipped neurons. We call their combined contribution the **residual**, and denote it as $r(x)$, where

$$r(x) = y(x) - \hat{y}(x) = \sum_{j \notin S(x)} \phi_j(x) W_{\text{down},j}, \tag{1}$$

and characterize it as a predictable signal. The predictability of the residual from x depends on how S was chosen, which we measure in Section 4.4. Our motivation for this is that an MLP's input-output map can be close to linear over the inputs in a specific layer. This means that an affine function of x may be able to characterize a part of $r(x)$. However, this does not mean that the MLP itself is linear, and in our measurements the best rank-q affine estimate removes a substantial but partial share of the residual (Section 4.4).

Because a few dimensions of x carry much larger activations than the rest, especially early in a sequence (Elhage et al., 2023; Sun et al., 2024a; Timkey & van Schijndel, 2021), and a predictor trained on typical tokens handles them poorly, we keep the first T_0 positions dense.

3.3 Rank-p importance prediction

Selectors like CATS (Lee et al., 2024) gate on the hidden activation itself, $g_j = \mathbb{1}[|\phi_j(x)| \geq t]$, so scoring them needs the computation of the gate projection that they aim to skip. TEAL avoids this by thresholding the MLP input by magnitude, and WINA weighs it by column norms (Liu et al., 2025; Chen et al., 2026). Both are cheap, but their selection is done per element. We score neurons with a learned low-rank surrogate that never forms $\phi(x)$:

$$\hat{s}(x) \;=\; \big((x-\mu) \oslash \varsigma\big) W_1 W_2, \qquad W_1 \in \mathbb{R}^{d\times p},\; W_2 \in \mathbb{R}^{p\times m}, \qquad S(x) \;=\; \text{top-}k\big(\hat{s}(x)\big), \tag{2}$$

with μ, ς as the mean per feature and standard deviation on calibration data, respectively, which removes the coordinate imbalance so no single outlier channel dominates the training process. Eq. 2 costs $p(d+m)$ operations against $(P-1)dm$ for the exact score (gate and up projections in the gated case) and Pdm for the full block ($P = 3$ gated, 2 ungated).

We fit $\hat{s}$ to rank neurons. With $e(x) = s(x)^2$ and $\pi(x) = e(x)/\sum_j e_j(x)$, the predictor's loss can be defined as:

$$\mathscr{L}_{\text{pred}} \;=\; -\mathbb{E}_x\Big[\textstyle\sum_j \pi_j(x) \log \text{softmax}_j\big(\hat{s}(x)\big)\Big], \tag{3}$$

This weights each neuron based on its contribution to the total importance mass. The target distribution π is unchanged by uniformly rescaling the oracle scores, while neuron selection depends only on the ranking of $\hat{s}$. We then select the predictor's checkpoint using captured importance on validation data, $\sum_{j\in S} s_j / \sum_{j\in S^\star} s_j$, evaluated against the oracle set $S^\star$ for multiple budgets. The predictor is trained once for each model and frozen afterward.

3.4 Rank-q compensation via reduced-rank ridge regression

We estimate $r(x)$ with a rank-q affine map $\mathscr{C}(x)$, which can be denoted as:

$$\mathscr{C}(x) \;=\; \big((x-\mu)A\big)B + c, \qquad A \in \mathbb{R}^{d\times q},\; B \in \mathbb{R}^{q\times d}, \tag{4}$$

This makes it so that the compressed block computes $\tilde{y}(x) = \hat{y}(x) + \mathscr{C}(x)$. On centered (where the mean was subtracted) calibration inputs X_c and residuals R_c (Eq. 1, at this layer's own k), we solve ridge regression and truncate it to rank q in the output space:

$$\Omega \;=\; \big(X_c^\top X_c + \lambda I\big)^{-1} X_c^\top R_c, \qquad V_q \;=\; \text{top-}q \text{ right singular vectors of } X_c\Omega, \tag{5}$$

In this, $A = \Omega V_q$, $B = V_q^\top$ and $c = \mu_r$, which is the mean calibration residual. This is reduced-rank ridge regression (Izenman, 1975; Mukherjee & Zhu, 2011), from $d \times d$ statistics in one streaming pass rather than an $N \times d$ matrix and its Singular Value Decomposition. This is similar to weight SVD (Yuan et al., 2025; Wang et al., 2025; Hsu et al., 2022; Abbasi et al., 2026), but this fits a regression of what one specific selection discards every token, rather than a single static factorization of the weight matrix, and it is refit for every (k, q) pair.

λ is chosen from $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$ (scaled by $\text{mean}\,\text{diag}(X_c^\top X_c)/N$) by error on validation data on the last 10% of calibration rows, and the fit is then redone on all rows. This is a simple approximation to bi-cross-validation for choosing the rank and shrinkage (Owen & Perry, 2009; Gavish

& Donoho, 2014). This is useful when there are not too many calibration rows as compared to the input dimension (an unregularized fit can overfit the noise). We use 48,000 calibration rows per layer (Section 4.1). If no value of λ performs better than simply predicting the mean residual on validation rows, we set $A = 0$ but keep the directions in B. This is so that distillation can still learn the map from zero.

3.5 Distillation with a Jacobian-lens term

The closed-form fit is greedy and per-layer in that it minimizes local reconstruction error against a residual that is defined by a dense upstream, not the model's actual output error once every compressed layer interacts. We distill $\{A, B, c\}$ end-to-end against the unmodified model, and predictors, gates, and the original weights stay frozen. We define our distillation loss as:

$$\mathscr{L} = \underbrace{\mathbb{E}_t\left[\mathrm{D}_{\mathrm{KL}}\left(p^{\mathrm{d}}(\cdot \mid x_{<t}) \,\middle\|\, p^{\mathrm{c}}(\cdot \mid x_{<t})\right)\right]}_{\text{output agreement}} + \lambda_J \kappa \underbrace{\frac{1}{|\mathscr{L}_c|} \sum_{\ell \in \mathscr{L}_c} \mathbb{E}_{t>T_0}\left[1 - \cos\left(\Lambda_\ell h_\ell^{\mathrm{c}}, \Lambda_\ell h_\ell^{\mathrm{d}}\right)\right]}_{\mathscr{L}_{\text{J-lens}}}, \quad (6)$$

where $\Lambda_\ell(h) = \mathrm{RMSNorm}(J_\ell h)$(Zhang & Sennrich, 2019) transports layer ℓ's residual state into the final-layer frame using the Jacobian-lens of Gurnee et al. (2026). J_ℓ is the Jacobian of the residual of the final frame with respect to layer ℓ's residual, averaged over the positions of 16 prompts and then frozen. J_ℓ is computed once per model ($\approx$ 22 minutes for Qwen2.5-7B on one RTX 5090). At step 0, κ is set to the ratio of the KL term to the J-lens term, so the two start equal. We use $\lambda_J = 1$ throughout (the term is optional). Replacing J_ℓ by the identity (plain hidden-state matching) leaves perplexity and MC accuracy unchanged, and the J-lens adds MMLU only at $\lambda_J = 10$ (+0.79, $p = 0.02$, paired). Distillation runs 3,000 steps of one 1,024-token sequence each ($\approx$ 3.1M token positions).

3.6 Budget allocation

The predictor and compensator are restricted to the same budget that they serve. For target fraction B of dense MLP FLOPs,

$$p(d+m) + 2qd + Pkd = BPdm \quad \Longrightarrow \quad k = \left\lfloor \frac{BPdm - p(d+m) - 2qd}{Pd} \right\rfloor. \quad (7)$$

So, raising p or q always costs neurons, and no configuration receives free overhead. For our experiments, we choose $(p, q) = (64, 384)$ with a uniform budget across compressed layers. The top-k selection over m scores and the input normalization are not charged since they cost $O(m)$ comparisons and $O(d)$ operations against $P \cdot dm$ operations, which is $< 0.01\%$ of a Qwen2.5-7B layer. A per-layer allocation search (Mi et al., 2025; Xv et al., 2025) and a per-token budget rather than a per-layer one would likely improve on this; however, we leave it out so that computation is independent of data (Liu et al., 2025; Zhang et al., 2025).

4 Experiments

In this section, we provide experimental results to answer the following four questions:

- **Q1.** At a fixed budget for MLP computation, is part of it better spent reconstructing the skipped neurons than computing more neurons?
- **Q2.** How does ReComp compare with training-free contextual sparsity methods?
- **Q3.** Why does the combination and pairing of a low-rank selector with a compensator work?
- **Q4.** Where do the gains stop and why?

4.1 Setup

Models, budgets and accounting. We evaluate Language Models from three families, all unquantized: Qwen2.5-0.5B Instruct, Qwen2.5-7B Instruct and Qwen2.5-14B Instruct (Qwen et al., 2025), Mistral-7B Instruct-v0.1 (Jiang et al., 2023) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024).

Layers ℓ for $\ell \in [\lfloor 0.25L \rfloor, \lfloor 0.90L \rfloor)$ are compressed, with the first $T_0 = 16$ tokens of every sequence kept dense. The budget $B \in \{0.25, 0.35, 0.50, 0.65\}$ is the fraction of dense arithmetic in a compressed MLP layer, with predictor and compensator charged against it (Eq. 7). In this, $B = 0.25$ is the most aggressive budget (most sparsity), while $B = 0.65$ is the least. The compensator costs $2qd$, or the price of 256 neurons, which is equivalent to 1.3-5.3% of dense MLP arithmetic depending on m. Because only middle-layer MLPs are compressed, $B = 0.25$ is $0.61 - 0.65$ of the entire model's arithmetic computation.

Adaptation. All models use $p = 64$ and $q = 384$. Calibration uses 48,000 token positions per compressed layer (200 sequences of 256 tokens, excluding each sequence's 16 dense-prefix positions, from WikiText-2 *train* (Merity et al., 2016), a C4 *train* (Raffel et al., 2023) shard, code, and chat data), with the last 10% held-out for choosing λ. The predictor is trained once per layer for 2,000 steps of 1,024 rows and frozen. The compensator is fit in closed form and distilled for 3,000 steps per budget of one 1,024-token sequence (3.07M token positions) with $\lambda_J = 1$, updating only compensator parameters with AdamW (learning rate 10^{-4}, or 10^{-5} for Mistral-7B (10^{-4} diverged) and Llama-3.1-8B, having a 20-step warmup and cosine decay. Qwen2.5-0.5B is the only model on which these choices were made, and every larger model uses them unchanged (apart from the learning rate).

Controls and baselines. Every control uses identical computation: *top-k* spends the compensator's $2qd$ arithmetic operations on $2q/P = 256$ extra exact neurons similar to DejaVu (Liu et al., 2023), though with our rank-64 linear predictor. *Ridge* is the compensator in closed form **before distillation**, and *oracle top-k* spends the whole budget on $k = Bm$ neurons that are ranked by their true write norm, with no predictor or compensator (making it a ceiling for our benchmarks (it needs the dense MLP)). We compare against TEAL (Liu et al., 2025), WINA (Chen et al., 2026) and R-Sparse (Zhang et al., 2025) using the authors' code, restricted to the same layers and dense prefix, and run with a uniform per-layer budget as ReComp is. TEAL runs without its greedy layer allocation and WINA without its column-orthogonal reparameterization, which understates both compared to their published numbers. Experiments that attach a compensator to TEAL's masks need hooks inside the MLP and use our re-implementation of its masking, which is denoted by TEAL†.

Evaluation. We report WikiText-2 test and C4 validation perplexity over 2,048-token windows relative to the dense model, and call a comparison a win **only when both datasets agree**. Downstream accuracy uses the lm-evaluation-harness (Gao et al., 2024), and **MC** is the mean over ARC-Easy, ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2019), WinoGrande (Sakaguchi et al., 2019), and BoolQ (Clark et al., 2019). MMLU is evaluated separately (Hendrycks et al., 2021). All positions after T_0 are sparsified in prefill and scoring. On Qwen2.5-0.5B, we test the differences with paired tests on the same questions (McNemar for MMLU, a paired bootstrap for MC).

4.2 Reconstruction Beats More Neurons (Responses to Q1)

With our selector, spending 256 neurons' worth of arithmetic (1.3–5.3% of dense MLP arithmetic) on the compensator beats spending it on 256 more exact neurons in all 20 model–budget settings (Table 1), and the margin grows as the budget shrinks. On Qwen2.5-7B at $B = 0.25$, relative perplexity falls from 2.347 to 1.158, closing 94% of the gap between top-k and the oracle ceiling (81% on Qwen2.5-14B, 76% on Llama-3.1-8B, 62% on Mistral-7B, where selection alone loses least). The closed-form fit does most of this by itself, and distillation improves on it in every WikiText-2 row. C4 agrees (Table 2), apart from Mistral-7B at $B \geq 0.50$, where distillation is marginally worse than ridge.

4.3 Comparison with Training-Free Contextual Sparsity (Responses to Q2)

At $B = 0.25$ ReComp outperforms R-Sparse, TEAL, and WINA on all three 7–8B models (Figure 2), by 0.28–0.39 relative perplexity on Qwen2.5-7B, 0.25–0.27 on Mistral-7B and 0.22–0.42 on Llama-3.1-8B. The closed-form compensator alone before distillation, also beats every baseline there (narrowly on Llama-3.1-8B: 1.580 versus 1.613 for WINA). At $B = 0.35$ ReComp remains best on Qwen2.5-7B and, on WikiText-2, on Mistral-7B, where on C4 it ties the baselines once their

Table 1: Relative WikiText-2 test perplexity (compressed/dense, lower is better). All ReComp columns share one predictor ($p = 64$) and the same analytical computation. Data labels are explained in Section 4.1. Reference dense WikiText-2 / C4 perplexity: 14.25/21.46 (0.5B), 7.46/12.55 (Qwen 7B), 6.91/10.93 (Mistral), 7.23/11.47 (Llama), 5.70/10.73 (14B). External baselines are run in uniform per-layer mode, and their realized computation is in Table 3. **Bold text**: best deployable column where WikiText-2 and C4 agree. Qwen2.5-0.5B rows are three-seed means (standard deviation ≤ 0.0021).

		Within-method (exact match)				External (uniform mode)		
Model	B	Top-k	Ridge	ReComp	Oracle	R-Sparse	TEAL	WINA
Qwen2.5-0.5B	.25	2.489	1.404	**1.189**	1.115	2.191	2.443	2.974
	.35	1.871	1.291	**1.140**	1.048	1.422	1.497	1.617
	.50	1.411	1.193	1.099	1.014	1.106	1.122	1.132
	.65	1.194	1.123	1.069	1.003	–	–	–
Qwen2.5-7B	.25	2.347	1.356	**1.158**	1.081	1.438	1.545	1.521
	.35	1.777	1.259	**1.115**	1.034	1.145	1.161	1.156
	.50	1.365	1.156	1.070	1.009	1.044	1.042	**1.040**
	.65	1.169	1.089	1.043	1.002	–	–	–
Mistral-7B	.25	1.335	1.175	**1.146**	1.030	1.406	1.413	1.400
	.35	1.229	1.126	**1.108**	1.013	1.137	1.141	1.137
	.50	1.135	1.079	1.073	1.004	**1.036**	1.037	**1.036**
	.65	1.075	1.048	1.046	1.001	–	–	–
Llama-3.1-8B	.25	2.317	1.580	**1.390**	1.093	1.812	1.628	1.613
	.35	1.740	1.414	1.297	1.038	1.290	1.234	**1.228**
	.50	1.373	1.255	1.196	1.010	1.083	1.070	**1.068**
	.65	1.193	1.150	1.122	1.002	–	–	–
Qwen2.5-14B	.25	2.375	1.570	1.353	1.118	–	–	–
	.35	1.785	1.426	1.276	1.054	–	–	–
	.50	1.407	1.276	1.194	1.015	–	–	–
	.65	1.221	1.178	1.133	1.003	–	–	–

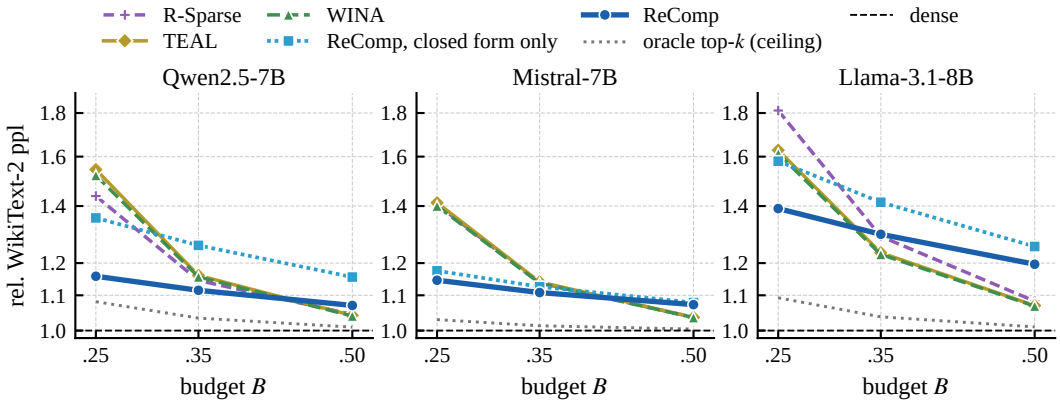

Figure 2: **ReComp against training-free contextual sparsity.** ReComp is best on every 7-8B model at $B = 0.25$, even before distillation; training-free methods are better at $B = 0.50$. Dotted: oracle ceiling. Realized arithmetic in Table 3.

arithmetic is matched to ReComp's; at $B = 0.50$ the training-free methods win on all three. On Qwen2.5-0.5B, at 0.40 and 0.45, ReComp (1.12 and 1.11) beats the best baseline, R-Sparse (1.26 and 1.17), on both datasets (Table 4).

Threshold selectors do not land exactly on their target. Measured per token with the dense prefix excluded (Table 3), the baselines receive *more* arithmetic than ReComp on Qwen2.5-7B, and TEAL

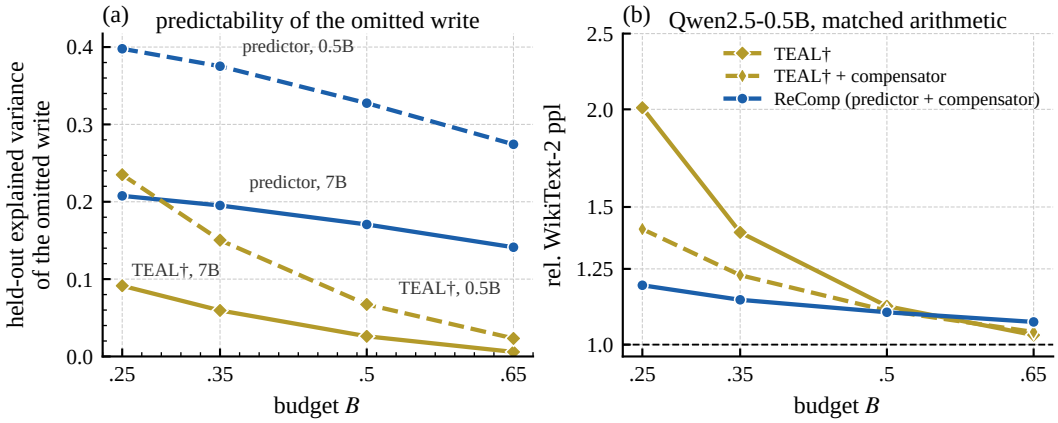

Figure 3: **A low-rank selector leaves a predictable residual; a magnitude threshold does not.** (a) Held-out fraction of the omitted write's variance explained by the same rank-q estimator, applied to what each selector leaves out at the same budget. (b) On Qwen2.5-0.5B at matched arithmetic, the same compensator helps TEAL† only where its residual is predictable. ReComp, whose residual stays predictable, is best at every budget up to 0.50.

and WINA do on Llama-3.1-8B, so those wins hold against better-resourced baselines. On Mistral-7B at $B \leq 0.35$, every baseline receives 1.0-3.2 points less computation, and so does R-Sparse on Llama-3.1-8B, so those margins are partly due to arithmetic. The gains are also not confined to the adaptation distribution: on PG-19 (Rae et al., 2020) and PubMed, which the predictor and compensator never saw, Qwen2.5-0.5B keeps the same ordering against TEAL† and the same crossover, although ReComp's lead at $B = 0.35$ narrows (Table 5); on Qwen2.5-7B at $B = 0.25$, ReComp reaches 1.23 and 1.22 on PG-19 and PubMed against 1.60 and 1.45 for TEAL†. Removing WikiText-2 and C4 from all adaptation data raises ReComp to 1.43 and 1.30 at $B = 0.25$ and 0.35 on Qwen2.5-0.5B, still ahead of the best baseline (2.19 and 1.42) on both datasets (Table 6): part of the margin is domain fit, but not the ordering. On Qwen2.5-7B the same ablation gives 1.34 and 1.22 at $B = 0.25$, still ahead of every baseline on both datasets (best 1.44 and 1.46), but 1.22 and 1.15 at $B = 0.35$, behind them (1.15 and 1.14), so the $B = 0.35$ advantage there depends on in-domain adaptation.

4.4 Co-design of Selector and Compensator (Responses to Q3)

Our rank-64 predictor is a weak selector on its own: plain predicted top-k is far worse than TEAL† (1.90 versus 1.39 at $B = 0.35$ on Qwen2.5-0.5B). Yet with the same compensator at the same arithmetic the order reverses: ReComp reaches 1.19 and 1.14 at $B = 0.25$ and 0.35, against 1.41 and 1.23 for TEAL† with the distilled compensator at matched measured arithmetic (Figure 3(b)). We attribute this to what each selector leaves out, although the two also differ in sparsity axis (TEAL† masks inputs and intermediates; the predictor drops whole neurons). A rank-p selector scores neurons with a low-rank function of x, so which neurons it misses is itself a systematic function of x, the thing an input-conditioned compensator can model. A per-element magnitude threshold omits a far less structured remainder. Measured directly, the predictor's omitted write is 1.7× (Qwen2.5-0.5B), 2.3× (Qwen2.5-7B), 1.9× (Mistral-7B) and 1.4× (Llama-3.1-8B) more predictable than TEAL†'s at $B = 0.25$, and far more at $B = 0.65$, where TEAL† leaves almost nothing predictable (0.023 and 0.006) (Figure 3(a)). This is not an artifact of normalizing by a larger residual. At $B = 0.35$ both selectors omit the same share of the dense write's energy (0.30), yet the compensator removes 40% of the predictor's omission and 17% of TEAL†'s (Table 8).

Three controls isolate the ingredients (Table 10). A constant bias equal to the mean skipped write recovers only 29-31% of the ridge fit's gain, so the input dependence carries the rest. A LoRA adapter with the same selector, rank, parameter count, arithmetic and training, differing only in its initialization, reaches 1.249 and 1.181 against 1.191 and 1.141 for ReComp at $B = 0.25$ and 0.35, with 3.4 and 1.8 points lower MMLU (29.85 and 32.77 against 33.21 and 34.54): the closed-form fit to the selector's residual matters beyond the added capacity and training. Finally, the selector decides whether the closed-form fit works, and distillation narrows the difference at scale. At matched

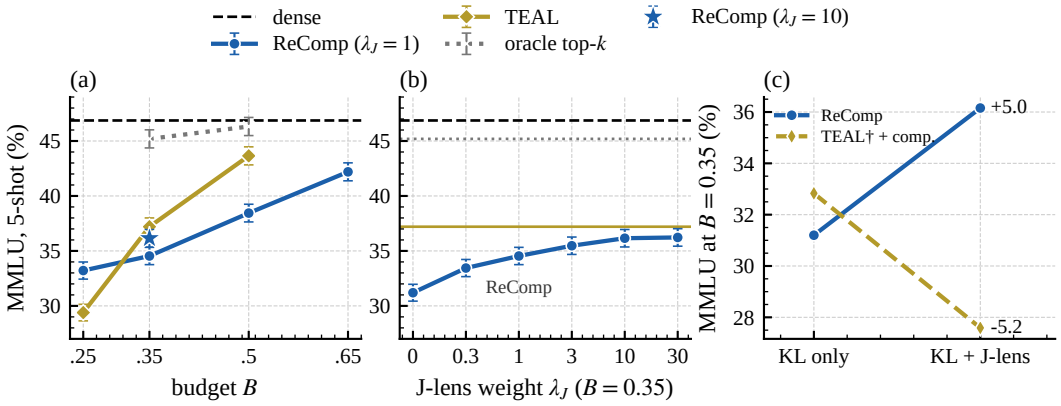

Figure 4: **Downstream accuracy on Qwen2.5-0.5B.** (a) Five-shot MMLU against the budget, with 95% intervals. ReComp beats TEAL at $B = 0.25$ while trailing at 0.35 and 0.50 (attributed to imperfect selection). The star is ReComp with $\lambda_J = 10$. (b) At $B = 0.35$ MMLU rises with the J-lens weight and saturates by $\lambda_J = 10$, where the gap to TEAL narrows to 1.05 points. (c) The same J-lens term raises MMLU on ReComp and lowers it on TEAL† with the compensator.

measured arithmetic, the ridge fit alone makes TEAL† worse on every model (at $B = 0.25$: 2.01 to 2.71 on Qwen2.5-0.5B, 1.42 to 2.06 on Qwen2.5-7B, 1.32 to 1.59 on Mistral-7B, 1.58 to 2.44 on Llama-3.1-8B), while for our selector it removes most of the gap to the oracle (2.35 to 1.36 on Qwen2.5-7B). Distilled, the compensator helps both: on Qwen2.5-0.5B, ReComp stays ahead, whereas on Qwen2.5-7B, TEAL† with the compensator matches ReComp at $B = 0.25$ (1.160 versus 1.158) and is better at $B = 0.35$ (1.079 versus 1.115; Table 9).

4.5 Downstream Accuracy and Generation (Responses to Q4)

Perplexity gains transfer to downstream accuracy only partially. On Qwen2.5-0.5B (Figure 4, Table 12), ReComp significantly beats TEAL at $B = 0.25$ on MC (+2.50) and MMLU (+3.82, paired McNemar test, $p < 10^{-13}$) but trails on MMLU at $B = 0.35$ and 0.50. On the 7B models at $B = 0.25$ it is within 0.6 MC points of the strongest baseline or ahead on all three 7–8B models, and on MMLU it leads on Mistral-7B (48.43 versus 46.21) but trails on Qwen2.5-7B and Llama-3.1-8B (59.23 versus 62.01, 54.56 versus 56.12); at $B \geq 0.35$ it trails on all three 7–8B models (Table 11). *Selection* explains the gap: on Qwen2.5-0.5B at $B = 0.35$, oracle top-k keeps MMLU within 1.7 points of dense, while TEAL and ReComp lose 9.7 and 12.3 (Figure 4(a)), suggesting that an aggregate compensator cannot stand in for the specific neurons factual recall uses. The *objective* matters a lot too: on Qwen2.5-0.5B, MMLU rises from 31.20 at $\lambda_J = 0$ to 36.16 at $\lambda_J = 10$, where ReComp is within 1.05 MMLU points of TEAL, with perplexity nearly unchanged. Because that sweep selects on MMLU, those numbers are tuned; applied unchanged at 7B, $\lambda_J = 10$ does not reproduce the gain (Table 11).

Free-running generation is ReComp's weakness (Table 13): at $B = 0.50$ GSM8K (Cobbe et al., 2021) falls from 86% to 67% on Qwen2.5-7B while the training-free baselines stay within about six points of dense. Sirius (Zhou et al., 2024) also finds that contextual sparsity hurts generation most, which makes generation-aware distillation the natural next step.

5 Conclusion

ReComp spends a small, fully charged part of a contextual-sparsity budget on reconstructing the output of skipped neurons, a cheap low-rank selector leaves a far more predictable residual than a magnitude threshold, and the pairing outperforms strong training-free methods in perplexity on every 7-8B model at the tightest budget. Its advantage is confined to aggressive budgets, which motivates further work for stronger selectors and generation-aware distillation.

AI USE STATEMENT

In this paper, we used LLM tools to help implement parts of the proposed method and its experimental code, propose and refine some hypotheses, interpret some results, give feedback on methodology/experiments, and propose fixes to issues encountered during development. All AI-assisted code was reviewed by the authors before use. Writing proofs, developing theoretical models, formulating mathematical claims, cleaning datasets, generating synthetic data, translation, qualitative/thematic data analysis and generating datasets do not apply to this work. Additionally, we used generative AI tools to identify relevant literature, with every reference verified by the authors against its source, and to edit/draft some areas of the text for readability. We have reviewed all AI-assisted work and take responsibility for all content.

ETHICS STATEMENT

This work studies inference efficiency for open-source language models via HuggingFace on public datasets and involves no new data collection or human subjects. Compressed models can behave differently from their original dense models, which is why we report where ReComp degrades quality.

REPRODUCIBILITY STATEMENT

We have taken the necessary steps to ensure our results are reproducible. All experiments were conducted on publicly available datasets and models. We used standard open-source evaluation pipelines such as lm-evaluation-harness for benchmarking. Detailed experiments can be found in Section 4. Our code was designed to be run on RTX 5090 and RTX 5060 Ti GPUs for several days, including the time for our method and baselines on different models. Our source code is available anonymously at our Anonymous GitHub and Supplementary Material.

REFERENCES

Ali Abbasi, Chayne Thrash, Haoran Qin, Hamed Pirsiavash, and Soheil Kolouri. Io-svd: Input-output whitened svd for adaptive-rank llm compression, 2026. URL https://arxiv.org/abs/2605.15626.

Yash Akhauri, Ahmed F. AbouElhamayed, Jordan Dotzel, Zhiru Zhang, Alexander M. Rush, Safeen Huda, and Mohamed S. Abdelfattah. ShadowLLM: Predictor-based contextual sparsity for large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 19154–19167, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1068. URL https://aclanthology.org/2024.emnlp-main.1068/.

Keivan Alizadeh, Seyed Iman Mirzadeh, Dmitry Belenko, S. Khatamifard, Minsik Cho, Carlo C Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. LLM in a flash: Efficient large language model inference with limited memory. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12562–12584, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.678. URL https://aclanthology.org/2024.acl-long.678/.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i10.28960. URL https://doi.org/10.1609/aaai.v38i10.28960.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Sihan Chen, Dan Zhao, Jongwoo Ko, Colby Banbury, Huiping Zhuang, Luming Liang, Pashmina Cameron, and Tianyi Chen. WINA: Weight informed neuron activation for accelerating large language model inference. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=l7Vb3yxmuz.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions, 2019. URL https://arxiv.org/abs/1905.10044.

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.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

CodeParrot Team. Codeparrot cleaned dataset. https://huggingface.co/datasets/codeparrot/codeparrot-clean, 2022. Accessed: 2026-09-25.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations, 2023. URL https://arxiv.org/abs/2305.14233.

Nelson Elhage, Robert Lasenby, and Christopher Olah. Privileged bases in the transformer residual stream. Transformer Circuits Thread, 2023. URL https://transformer-circuits.pub/2023/privileged-basis/index.html.

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 2024. URL https://zenodo.org/records/12608602.

Roberto Garcia, Jerry Weihong Liu, Daniel Sorvisto, and Sabri Eyuboglu. Adaptive rank allocation: Speeding up modern transformers with raNA adapters. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=uAtDga3q0r.

Matan Gavish and David L. Donoho. The optimal hard threshold for singular values is 4/sqrt(3), 2014. URL https://arxiv.org/abs/1305.5870.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Wes Gurnee, Nicholas Sofroniew, Adam Pearce, Mateusz Piotrowski, Isaac Kauvar, Runjin Chen, Anna Soligo, Paul Bogdan, Euan Ong, Rowan Wang, Ben Thompson, David Abrahams, Subhash Kantamneni, Emmanuel Ameisen, Joshua Batson, and Jack Lindsey. Verbalizable representations form a global workspace in language models. *Transformer Circuits Thread*, 2026. URL https://transformer-circuits.pub/2026/workspace/index.html.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. Language model compression with weighted low-rank factorization, 2022. URL https://arxiv.org/abs/2207.00112.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Alan Julian Izenman. Reduced-rank regression for the multivariate linear model. *Journal of Multivariate Analysis*, 5(2):248–264, 1975. ISSN 0047-259X. doi: https://doi.org/10.1016/0047-259X(75)90042-1. URL https://www.sciencedirect.com/science/article/pii/0047259X75900421.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Donghyun Lee, Jaeyong Lee, Genghan Zhang, Mo Tiwari, and Azalia Mirhoseini. CATS: Context-aware thresholding for sparsity in large language models. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=v3w2a7EInO.

James Liu, Pragaash Ponnusamy, Tianle Cai, Han Guo, Yoon Kim, and Ben Athiwaratkun. Training-free activation sparsity in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=dGVZwyq5tV.

Shih-Yang Liu, Maksim Khadkevich, Nai Chit FUNG, Charbel Sakr, Chao-Han Huck Yang, Chien-Yi Wang, Saurav Muralidharan, Hongxu Yin, Kwang-Ting Cheng, Jan Kautz, Yu-Chiang Frank Wang, Pavlo Molchanov, and Min-Hung Chen. EoRA: Fine-tuning-free compensation for compressed LLM with eigenspace low-rank approximation. In *Third Workshop on Test-Time Updates (Main Track)*, 2026. URL https://openreview.net/forum?id=CBVNZfPSGs.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Ré, and Beidi Chen. Deja vu: contextual sparsity for efficient llms at inference time. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Mehdi Makni, Xiang Meng, and Rahul Mazumder. 3BASil: An algorithmic framework for sparse plus low-rank compression of LLMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=byNNv5Et10.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models, 2025. URL https://arxiv.org/abs/2404.02948.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843.

Zhendong Mi, Bian Sun, Grace Li Zhang, and Shaoyi Huang. Layer-wise dynamic rank for compressing large language models, 2025. URL https://arxiv.org/abs/2509.25622.

Mohammad Mozaffari, Amir Yazdanbakhsh, and Maryam Mehri Dehnavi. SLim: One-shot quantization and sparsity with low-rank approximation for LLM weight compression. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=4UfRP8MopP.

Ashin Mukherjee and Ji Zhu. Reduced rank ridge regression and its kernel extensions. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 4:612–622, 11 2011. doi: 10.1002/sam.10138.

Art B. Owen and Patrick O. Perry. Bi-cross-validation of the svd and the nonnegative matrix factorization. *The Annals of Applied Statistics*, 3(2), 2009. ISSN 1932-6157. doi: 10.1214/08-aoas227. URL http://dx.doi.org/10.1214/08-AOAS227.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SylKikSYDH.

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, 2023. URL https://arxiv.org/abs/1910.10683.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Chenyang Song, Xu Han, Zhengyan Zhang, Shengding Hu, Xiyu Shi, Kuai Li, Chen Chen, Zhiyuan Liu, Guangli Li, Tao Yang, and Maosong Sun. Prosparse: Introducing and enhancing intrinsic activation sparsity within large language models, 2025. URL https://arxiv.org/abs/2402.13516.

Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. Powerinfer: Fast large language model serving with a consumer-grade gpu, 2024. URL https://arxiv.org/abs/2312.12456.

Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. Massive activations in large language models, 2024a. URL https://arxiv.org/abs/2402.17762.

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*, 2024b. URL https://openreview.net/forum?id=PxoFut3dWW.

Philippe Tillet, H. T. Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, MAPL 2019, pp. 10–19, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367196. doi: 10.1145/3315508.3329973. URL https://doi.org/10.1145/3315508.3329973.

William Timkey and Marten van Schijndel. All bark and no bite: Rogue dimensions in transformer language models obscure representational quality, 2021. URL https://arxiv.org/abs/2109.04404.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

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

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Lin Xv, Jingsheng Gao, Xian Gao, Ting Liu, and Yuzhuo Fu. Ara: Adaptive rank allocation for efficient large language model svd compression, 2025. URL https://arxiv.org/abs/2510.19389.

Zhewei Yao, Xiaoxia Wu, Cheng Li, Stephen Youn, and Yuxiong He. Zeroquant-v2: Exploring post-training quantization in llms from comprehensive study to low rank compensation, 2023. URL https://arxiv.org/abs/2303.08302.

Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: Activation-aware singular value decomposition for compressing large language models, 2025. URL https://openreview.net/forum?id=HyPofygOCT.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Biao Zhang and Rico Sennrich. Root mean square layer normalization, 2019. URL https://arxiv.org/abs/1910.07467.

Cheng Zhang, Jianyi Cheng, George Anthony Constantinides, and Yiren Zhao. LQER: Low-rank quantization error reconstruction for LLMs. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 58763–58779. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhang24j.html.

Stephen Zhang and Vardan Papyan. Oats: Outlier-aware pruning through sparse and low rank decomposition, 2025. URL https://arxiv.org/abs/2409.13652.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Moefication: Transformer feed-forward layers are mixtures of experts, 2022. URL https://arxiv.org/abs/2110.01786.

Zhenyu Zhang, Zechun Liu, Yuandong Tian, Harshit Khaitan, Zhangyang Wang, and Steven Li. R-sparse: Rank-aware activation sparsity for efficient LLM inference. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=9VMW4iXfKt.

Yang Zhou, Zhuoming Chen, Zhaozhuo Xu, Xi Victoria Lin, and Beidi Chen. Sirius : Contexual sparisty with correction for efficient llms. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 24046–24080. Curran Associates, Inc., 2024. doi: 10.52202/079017-0758. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/2ae6b2bdf3a179e3e24129e2c54bd871-Paper-Conference.pdf.

A Additional Experimental Results

A.1 Perplexity: C4, per-model curves and realized arithmetic

C4 mirrors WikiText-2 (Table 2). Figure 5 plots the within-method comparison of Table 1 for every model. Threshold selectors do not land exactly on their configured sparsity, so Table 3 reports each baseline row's realized arithmetic against ReComp's on one basis. RaNA (Garcia et al., 2025), run on Qwen2.5-0.5B only (its 7B run did not complete), sets its own arithmetic through rank allocation: at 0.373 and 0.525 of dense it reaches relative WikiText-2 perplexity 2.270 and 1.258 (C4 1.702 and 1.202), against 1.140 and 1.099 for ReComp at the lower budgets of 0.35 and 0.50; its MMLU at 0.525 (39.1) is comparable to ReComp's at 0.50 (38.4). We do not include CATS (Lee et al., 2024) as it computes the gate projection densely for every neuron before thresholding and sparsifies only the up and down projections. This means that its MLP arithmetic cannot fall below $1/3$ of dense, so $B = 0.25$ is infeasible, and at $B = 0.35$ it could keep only 2.5% of neurons in the up and down projections.

Table 2: Relative C4 validation perplexity (compressed / dense; lower is better). All ReComp columns share one predictor ($p = 64$) and identical analytical arithmetic; Top-k spends the compensator's $2qd$ on $2q/P = 256$ extra neurons, Ridge is the closed-form compensator before distillation. *Oracle* spends the whole budget on $k = Bm$ neurons ranked by their true write norm, with no predictor or compensator; it needs the dense MLP, so it is a ceiling. External baselines are run in uniform per-layer mode at $B \leq 0.50$ (a dash marks a setting not run); their realised arithmetic is in Table 3. **Bold**: best deployable method where WikiText-2 and C4 agree. Qwen2.5-0.5B rows are three-seed means.

		Within-method (exact match)				External (uniform mode)		
Model	B	Top-k	Ridge	ReComp	Oracle	R-Sparse	TEAL	WINA
Qwen2.5-0.5B	.25	2.359	1.437	**1.237**	1.120	2.101	2.321	3.142
	.35	1.859	1.302	**1.174**	1.053	1.390	1.453	1.667
	.50	1.433	1.187	1.122	1.015	1.100	1.113	1.125
	.65	1.212	1.114	1.084	1.004	–	–	–
Qwen2.5-7B	.25	1.816	1.310	**1.162**	1.068	1.462	1.597	1.572
	.35	1.529	1.225	**1.116**	1.028	1.140	1.162	1.158
	.50	1.275	1.137	1.071	1.008	1.037	1.036	**1.035**
	.65	1.134	1.077	1.041	1.002	–	–	–
Mistral-7B	.25	1.258	1.146	**1.132**	1.028	1.320	1.327	1.318
	.35	1.181	1.103	**1.097**	1.013	1.104	1.106	1.103
	.50	1.110	1.063	1.064	1.004	**1.024**	**1.025**	**1.024**
	.65	1.064	1.039	1.041	1.001	–	–	–
Llama-3.1-8B	.25	2.079	1.581	**1.425**	1.117	1.758	1.607	1.591
	.35	1.680	1.432	1.334	1.056	1.297	1.248	**1.244**
	.50	1.394	1.290	1.237	1.017	1.104	1.086	**1.084**
	.65	1.236	1.192	1.163	1.004	–	–	–
Qwen2.5-14B	.25	1.771	1.379	1.219	1.061	–	–	–
	.35	1.478	1.269	1.159	1.026	–	–	–
	.50	1.251	1.161	1.101	1.008	–	–	–
	.65	1.127	1.093	1.061	1.002	–	–	–

A.2 Out-of-distribution perplexity

On PG-19 and PubMed, ReComp is best at $B \leq 0.35$ on Qwen2.5-0.5B and at $B = 0.25$ on Qwen2.5-7B and TEAL† at $B \geq 0.50$, the same crossover as in-distribution. ReComp's absolute degradation is larger off-distribution, while TEAL†'s is not: at $B = 0.35$, ReComp's lead over TEAL† at configured sparsity lessens from 0.33 on WikiText-2 to 0.27 on PG-19 and 0.17 on PubMed. Part of the adaptation is therefore domain-specific. Table 6 removes the evaluation domains from adaptation altogether: the predictor, the closed-form fit, and distillation all use code and chat data only.

Table 3: **Realised MLP arithmetic of every external baseline row, against ReComp's.** Both columns are per compressed token with the $T_0 = 16$ dense-prefix positions excluded, the basis on which the baselines' achieved sparsity is measured (held-out rows, mean over gate, up and down inputs); on this basis ReComp's arithmetic is exactly its charged budget. Threshold selectors do not land on their configured sparsity. Δ is the baseline's arithmetic minus ReComp's, in points: positive means the baseline was given *more* compute than ReComp. Baseline rows with more arithmetic than ReComp: Qwen2.5-7B 8 of 9; Mistral-7B 1 of 9; Llama-3.1-8B 6 of 9.

Model	Method	B	configured s	achieved s	baseline arith.	ReComp arith.	Δ (pts)
Qwen2.5-0.5B	R-Sparse	.25	0.7556	0.7495	0.2505	0.2498	+0.1
	R-Sparse	.35	0.6550	0.6501	0.3499	0.3499	-0.0
	R-Sparse	.50	0.5033	0.4993	0.5007	0.4998	+0.1
	TEAL	.25	0.7559	0.7654	0.2346	0.2498	-1.5
	TEAL	.35	0.6551	0.6636	0.3364	0.3499	-1.4
	TEAL	.50	0.5039	0.5105	0.4895	0.4998	-1.0
	WINA	.25	0.7559	0.7557	0.2443	0.2498	-0.6
	WINA	.35	0.6551	0.6576	0.3424	0.3499	-0.8
	WINA	.50	0.5039	0.5062	0.4938	0.4998	-0.6
Qwen2.5-7B	R-Sparse	.25	0.7678	0.7390	0.2610	0.2500	+1.1
	R-Sparse	.35	0.6722	0.6376	0.3624	0.3500	+1.2
	R-Sparse	.50	0.5286	0.5016	0.4984	0.5000	-0.2
	TEAL	.25	0.7559	0.7391	0.2609	0.2500	+1.1
	TEAL	.35	0.6551	0.6293	0.3707	0.3500	+2.1
	TEAL	.50	0.5039	0.4820	0.5180	0.5000	+1.8
	WINA	.25	0.7559	0.7360	0.2640	0.2500	+1.4
	WINA	.35	0.6551	0.6278	0.3722	0.3500	+2.2
	WINA	.50	0.5039	0.4792	0.5208	0.5000	+2.1
Mistral-7B	R-Sparse	.25	0.7559	0.7817	0.2183	0.2500	-3.2
	R-Sparse	.35	0.6550	0.6662	0.3338	0.3500	-1.6
	R-Sparse	.50	0.5039	0.5061	0.4939	0.5000	-0.6
	TEAL	.25	0.7559	0.7786	0.2214	0.2500	-2.9
	TEAL	.35	0.6551	0.6626	0.3374	0.3500	-1.3
	TEAL	.50	0.5039	0.5014	0.4986	0.5000	-0.1
	WINA	.25	0.7559	0.7762	0.2238	0.2500	-2.6
	WINA	.35	0.6551	0.6598	0.3402	0.3500	-1.0
	WINA	.50	0.5039	0.4992	0.5008	0.5000	+0.1
Llama-3.1-8B	R-Sparse	.25	0.7559	0.7712	0.2288	0.2500	-2.1
	R-Sparse	.35	0.6550	0.6665	0.3335	0.3500	-1.7
	R-Sparse	.50	0.5039	0.5165	0.4835	0.5000	-1.6
	TEAL	.25	0.7559	0.7425	0.2575	0.2500	+0.7
	TEAL	.35	0.6551	0.6354	0.3646	0.3500	+1.5
	TEAL	.50	0.5039	0.4871	0.5129	0.5000	+1.3
	WINA	.25	0.7559	0.7402	0.2598	0.2500	+1.0
	WINA	.35	0.6551	0.6326	0.3674	0.3500	+1.7
	WINA	.50	0.5039	0.4844	0.5156	0.5000	+1.6

Table 4: **A finer budget grid around the crossover** (Qwen2.5-0.5B, relative perplexity, WikiText-2 / C4). Rows at $B = .35$ and .50 repeat Tables 1 and 2 (three-seed means for the ReComp columns); rows at .40 and .45 are single-seed. Measured baseline arithmetic at .40 and .45 relative to ReComp's: R-Sparse $+0.1$, TEAL -1.2, WINA -0.6 to -0.7 points. **Bold**: best method where WikiText-2 and C4 agree.

B	Top-k	Ridge	ReComp	R-Sparse	TEAL	WINA
.35	1.871 / 1.859	1.291 / 1.302	**1.140 / 1.174**	1.422 / 1.390	1.497 / 1.453	1.617 / 1.667
.40	1.695 / 1.689	1.250 / 1.255	**1.124 / 1.154**	1.262 / 1.241	1.308 / 1.280	1.359 / 1.365
.45	1.540 / 1.548	1.219 / 1.219	**1.112 / 1.137**	1.166 / 1.155	1.195 / 1.177	1.212 / 1.208
.50	1.411 / 1.433	1.193 / 1.187	1.099 / 1.122	1.106 / 1.100	1.122 / 1.113	1.132 / 1.125

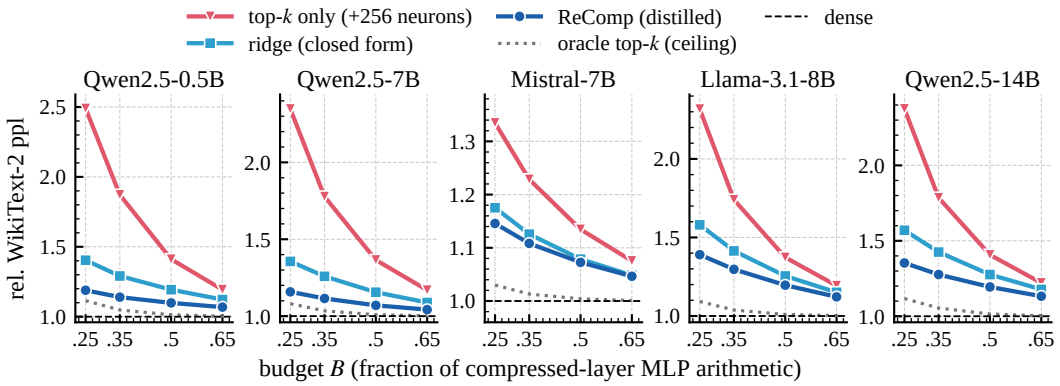

Figure 5: **At matched arithmetic, reconstructing the skipped write beats computing more neurons in every model and budget.** Top-k spends the compensator's arithmetic on 256 extra exact neurons; ridge is the closed-form compensator; ReComp adds distillation; the dotted oracle ranks neurons by their true write norm and is a ceiling.

Table 5: **Perplexity outside the adaptation mixture.** PG-19 (public-domain books) and PubMed abstracts appear in none of the four adaptation sources (WikiText-2, C4, CodeParrot (CodeParrot Team, 2022), UltraChat Ding et al. (2023)), so they are out-of-distribution for the predictor and the compensator. 128 sequences of 2048 tokens each, relative to the dense model's perplexity on the same set. TEAL† is our reimplementation of TEAL's masking (Section 4.1). TEAL† at $B = .25$ is 2.2555 on PG-19 and 2.2547 on PubMed; both round to 2.255. Bold: best non-oracle method.

		PG-19					PubMed				
Model	B	Top-k	Ridge	ReComp	TEAL†	Oracle	Top-k	Ridge	ReComp	TEAL†	Oracle
Qwen2.5-0.5B	.25	2.084	1.448	**1.267**	2.255	1.117	2.338	1.622	**1.369**	2.255	1.116
	.35	1.712	1.319	**1.201**	1.469	1.051	1.821	1.416	**1.268**	1.442	1.051
	.50	1.377	1.206	1.140	**1.123**	1.015	1.402	1.246	1.181	**1.114**	1.015
	.65	1.192	1.126	1.099	**1.031**	1.004	1.195	1.143	1.120	**1.030**	1.004
Qwen2.5-7B	.25	1.889	1.390	**1.231**	1.604	1.078	1.664	1.295	**1.224**	1.453	1.053

Table 6: **Adaptation without the evaluation domains** (relative perplexity, WikiText-2 / C4). *Mixture*: the paper's calibration and distillation data (WikiText-2 train, a C4 train shard, code, chat). *Code + chat*: identical pipeline with neither evaluation domain in the predictor's, compensator's or distillation's data. *Best baseline*: the strongest of R-Sparse, TEAL and WINA, which use no adaptation data. Single seed.

Model	B	Method	Top-k	Ridge	ReComp
Qwen2.5-0.5B	.25	ReComp, Mixture	2.506 / 2.361	1.399 / 1.435	1.191 / 1.238
		ReComp, Code + chat	2.540 / 2.246	1.777 / 1.635	1.430 / 1.334
		Best baseline (R-Sparse)		2.191 / 2.101	
	.35	ReComp, Mixture	1.897 / 1.867	1.288 / 1.301	1.141 / 1.176
		ReComp, Code + chat	1.909 / 1.795	1.511 / 1.414	1.300 / 1.242
		Best baseline (R-Sparse)		1.422 / 1.390	
Qwen2.5-7B	.25	ReComp, Mixture	2.347 / 1.816	1.356 / 1.310	1.158 / 1.162
		ReComp, Code + chat	2.151 / 1.766	1.532 / 1.343	1.343 / 1.222
		Best baseline (R-Sparse)		1.438 / 1.462	
	.35	ReComp, Mixture	1.777 / 1.529	1.259 / 1.225	1.115 / 1.116
		ReComp, Code + chat	1.719 / 1.505	1.364 / 1.245	1.222 / 1.154
		Best baseline (R-Sparse)		1.145 / 1.140	

A.3 CO-DESIGN: PREDICTABILITY AND ABSOLUTE RESIDUAL ENERGY

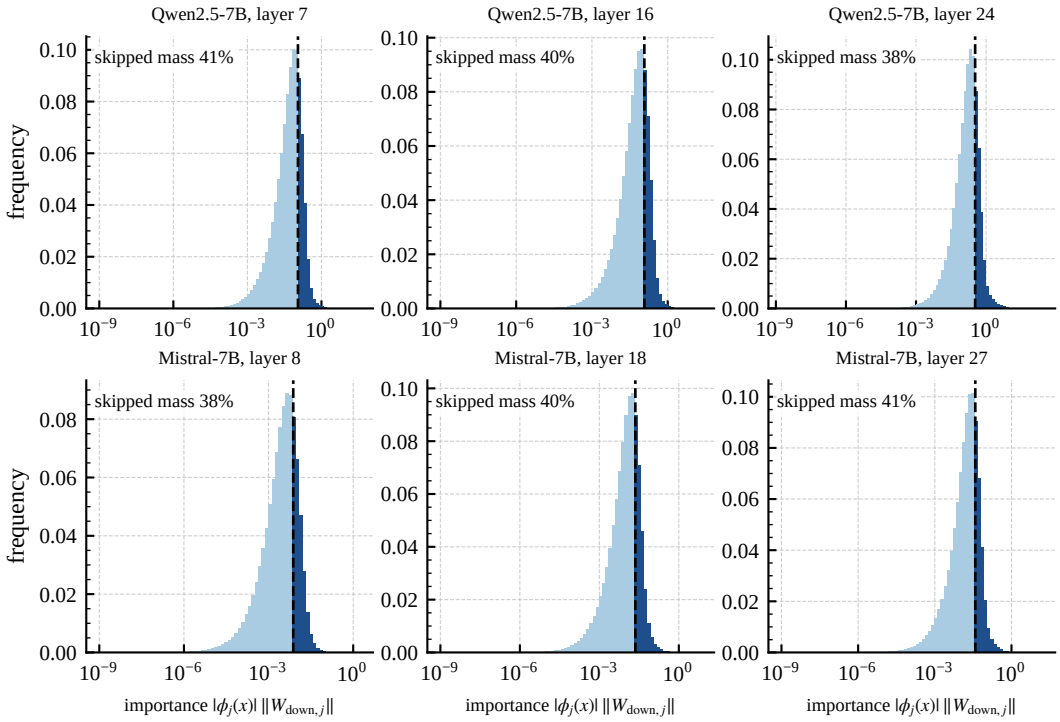

Figure 6: **The skipped neurons carry a large share of the output.** Distribution of neuron importance $|\phi_j(x)| \, \|W_{\text{down},j}\|$ over all (token, neuron) pairs at the first, middle and last compressed layer of Qwen2.5-7B and Mistral-7B. Dashed: the mean top-k cutoff at the operating neuron count; dark: kept, light: skipped. Although each skipped neuron is small, together they carry 38-41% of the importance mass.

Table 7 gives the numbers behind Figure 3(a). Table 8 reports energies as fractions of the dense write rather than of the residual, so a larger residual cannot look more compensable merely by being larger. The low-rank-residual intuition gets only modest direct support: the predictor's omitted write puts 0.83–0.85 of its energy in its top 384 directions against 0.77–0.82 for TEAL†, so the difference lies mainly in how predictable the residual is from x rather than in its intrinsic rank.

Table 7: **A low-rank selector leaves a more predictable residual than a magnitude threshold.** Held-out fraction of the omitted write's variance explained by the same rank-q reduced-rank ridge estimator (same rows, same held-out split, same λ grid), applied to the residual each selector leaves at the same budget; mean over compressed layers, range in parentheses. Our rank-p predictor scores with a rank-p map, so what it omits is a systematic function of x, which an input-conditioned compensator can model. A per-element magnitude threshold omits a far less structured object, and the gap widens as the budget grows. This is why the closed-form fit helps our selector and hurts TEAL† (Table 9).

Model	B	rank-p predictor top-k	TEAL† threshold	ratio
Qwen2.5-0.5B	.25	**0.398** (0.316, 0.489)	0.235 (0.176, 0.283)	1.7×
	.35	**0.375** (0.301, 0.455)	0.150 (0.115, 0.190)	2.5×
	.50	**0.327** (0.265, 0.394)	0.067 (0.051, 0.095)	4.9×
	.65	**0.274** (0.225, 0.328)	0.023 (0.018, 0.037)	11.8×
Qwen2.5-7B	.25	**0.208** (0.120, 0.296)	0.091 (0.069, 0.105)	2.3×
	.35	**0.195** (0.121, 0.271)	0.060 (0.049, 0.071)	3.3×
	.50	**0.171** (0.111, 0.235)	0.026 (0.021, 0.033)	6.5×
	.65	**0.141** (0.095, 0.195)	0.006 (0.004, 0.010)	24.0×
Mistral-7B	.25	**0.138** (0.088, 0.173)	0.071 (0.038, 0.095)	1.9×
	.35	**0.132** (0.085, 0.169)	0.048 (0.024, 0.070)	2.7×
Llama-3.1-8B	.25	**0.139** (0.060, 0.195)	0.097 (0.061, 0.123)	1.4×
	.35	**0.125** (0.053, 0.176)	0.063 (0.039, 0.079)	2.0×

Table 8: **Absolute energy of the omitted write, before and after compensation.** Fractions of the dense MLP write's energy, $\mathbb{E}\|r\|^2/\mathbb{E}\|y\|^2$ (*omitted*) and $\mathbb{E}\|r-\mathscr{C}(x)\|^2/\mathbb{E}\|y\|^2$ (*after*), for our rank-64 predictor and for TEAL at the same arithmetic, on calibration rows that no fit has seen. Because these are normalized by y, not by r, a larger residual cannot look more compensable merely by being larger. *rank-q* is the share of the omitted write's energy in its top $q = 384$ principal directions, a property of the residual alone. Mean over compressed layers.

		rank-64 predictor			TEAL† threshold		
Model	B	omitted	after	rank-q	omitted	after	rank-q
Qwen2.5-0.5B	.25	0.382	**0.218**	0.850	0.501	0.371	0.818
	.35	0.301	**0.180**	0.845	0.296	0.246	0.798
	.50	0.212	0.137	0.837	0.120	**0.111**	0.780
	.65	0.142	0.100	0.829	0.040	**0.039**	0.772

A.4 Compensation on TEAL and component controls

TEAL† is our reimplementation of TEAL's per-element magnitude masking on the gate/up and down inputs. Uncompensated, it is slightly *stronger* than the reference code on Qwen2.5-0.5B (relative WikiText-2 perplexity 2.34 versus 2.44 at $B = 0.25$, 1.47 versus 1.50 at $B = 0.35$, 1.120 versus 1.122 at $B = 0.50$), partly because the reference overshoots its target sparsity. On TEAL† the closed-form fit alone *degrades* perplexity at every budget, although on held-out calibration rows it lowers each layer's relative write error from 0.74 to 0.61: with TEAL† installed upstream, a layer's inputs drift from the dense calibration data (the cosine between their mean and the calibration mean falls to 0.80–0.90), so the affine map is evaluated off-distribution and its error compounds across layers. Distillation removes this mis-specification (Table 9).

Table 9: **Compensation on top of a strong training-free selector, at matched measured arithmetic.** The same rank-q compensator, fit to what TEAL† omits and paid for by raising its sparsity, with TEAL†'s sparsity set so that arithmetic measured on held-out rows equals the budget in every arm. Relative perplexity, WikiText-2 / C4. The closed-form fit alone makes TEAL† worse on every model; distillation (KL only) reverses this. At 0.5B ReComp stays well ahead; at 7B the two are tied at $B = .25$ and TEAL† is ahead at $B = .35$. The 0.5B rows at $B \geq .50$ are from an earlier run whose compensated arms had slightly less arithmetic (WikiText-2 only).

Model	B	arith.	TEAL†	+ mean	+ ridge	+ ridge, distilled	ReComp
Qwen2.5-0.5B	.25	0.247	2.009 / 1.924	3.442 / 3.126	2.711 / 2.803	1.405 / 1.460	**1.189 / 1.237**
	.35	0.347	1.392 / 1.359	1.802 / 1.730	1.775 / 1.828	1.227 / 1.264	**1.140 / 1.174**
	.50	0.487	1.120	1.217	1.228	1.105	**1.099**
	.65	0.644	**1.029**	1.051	1.051	1.038	1.069
Qwen2.5-7B	.25	0.243	1.417 / 1.450	1.825 / 1.925	2.064 / 2.263	1.160 / 1.162	**1.158 / 1.162**
	.35	0.343	1.143 / 1.141	1.206 / 1.219	1.296 / 1.343	**1.079 / 1.082**	1.115 / 1.116
Mistral-7B	.25	0.246	1.317 / 1.250	1.465 / 1.385	1.593 / 1.584	–*	1.146 / 1.132
	.35	0.346	1.124 / 1.093	1.157 / 1.122	1.192 / 1.175	–*	1.108 / 1.097
Llama-3.1-8B	.25	0.241	1.581 / 1.561	2.521 / 2.457	2.442 / 2.406	1.389 / 1.410	1.390 / 1.425
	.35	0.340	1.245 / 1.256	1.390 / 1.397	1.455 / 1.459	1.230 / 1.255	1.297 / 1.334

* Mistral-7B diverged at learning rate 10^{-4} when distilled

Table 10: Component controls on Qwen2.5-0.5B, relative WikiText-2 perplexity. Every row inside a selector uses identical analytical arithmetic. The mean-only row keeps the Top-k neuron count and adds a constant bias $c = \mu_r$ refit for that count, so it isolates the input-independent part of the compensator, the bias compensation of FLAP (An et al., 2024). The LoRA-init row is a parameter-, arithmetic- and training-matched adapter: the same k and rank-384 map, distilled identically, but started from LoRA's initialization instead of the closed-form fit. Oracle top-k selects on true write norms and is a ceiling. All rows come from one single-seed run so that the controls are internally comparable, which is why the ReComp row differs in the third decimal from the three-seed mean in Table 1. TEAL† rows are at matched measured arithmetic.

Selector	Compensation	Trained	$B = .25$	$B = .35$
Predicted top-k	none (+256 neurons)	no	2.506	1.897
Predicted top-k	mean only ($c = \mu_r$)	no	2.183	1.708
Predicted top-k	ridge	no	1.399	1.288
Predicted top-k	ridge init (ReComp)	yes	1.191	1.141
Predicted top-k	LoRA init (matched params)	yes	1.249	1.181
TEAL†	none	no	2.009	1.392
TEAL†	mean only	no	3.442	1.802
TEAL†	ridge	no	2.711	1.775
TEAL†	ridge init, distilled	yes	1.405	1.227
Oracle top-k	none	no	1.114	1.048

A.5 Downstream accuracy and generation

Table 11: Downstream accuracy (%). MC: mean of ARC-E, ARC-C, HellaSwag, PIQA, WinoGrande and BoolQ. MMLU is five-shot. All methods share the compressed-layer range and the $T_0 = 16$ dense prefix. At $B = .25$, ReComp leads on MC on Qwen2.5-7B and Mistral-7B and on MMLU on Mistral-7B, trails on MMLU on Qwen2.5-7B, and trails slightly on both on Llama-3.1-8B; at $B = 0.35$ and 0.50 its perplexity advantage does *not* transfer (Section 4.5). λ_J=10 rows apply the 0.5B setting unchanged. No baseline was run on Qwen2.5-14B, so its rows show retention against dense only.

		$B = .25$		$B = .35$		$B = .50$	
Model	Method	MC	MMLU	MC	MMLU	MC	MMLU
Qwen2.5-7B	Dense	75.68	74.27	75.68	74.27	75.68	74.27
	ReComp	71.05	59.23	72.48	63.42	74.48	66.94
	ReComp (λ_J=10)	–	–	72.97	62.82	–	–
	R-Sparse	70.10	61.70	73.42	68.42	74.96	71.78
	TEAL	70.34	61.49	73.63	68.38	75.34	72.17
	WINA	70.36	62.01	73.89	68.59	75.18	72.20
Mistral-7B	Dense	72.93	53.75	72.93	53.75	72.93	53.75
	ReComp	69.53	48.43	70.52	49.88	71.31	51.38
	ReComp (λ_J=10)	–	–	70.31	50.48	–	–
	R-Sparse	67.98	45.59	70.84	50.66	72.37	52.71
	TEAL	67.86	45.90	70.98	50.88	72.36	52.58
	WINA	68.09	46.21	70.98	50.50	72.22	52.61
Llama-3.1-8B	Dense	75.83	68.25	75.83	68.25	75.83	68.25
	ReComp	69.03	54.56	71.40	58.65	73.05	62.96
	R-Sparse	68.35	55.33	72.72	62.53	74.79	66.30
	TEAL	69.58	56.07	73.22	63.09	74.88	66.26
	WINA	69.07	56.12	73.21	63.25	75.24	66.69
Qwen2.5-14B	Dense	78.85	79.86	78.85	79.86	78.85	79.86
	ReComp	–	–	75.30	68.99	77.15	73.75

Table 12: Qwen2.5-0.5B downstream accuracy (%), ReComp at every budget and J-lens weight against TEAL at the same budget. Δ is ReComp minus TEAL; * marks a difference larger than two binomial standard errors. Dense: MC 56.57, MMLU 46.87. Paired tests (Section 4.1) also find the $+0.89$ MC difference at $B = 0.35$, $\lambda_J = 1$ and the -1.05 MMLU difference at $\lambda_J = 10$ significant.

B	ReComp λ_J	MC	MMLU	TEAL MC	TEAL MMLU	ΔMC	ΔMMLU
.25	1	51.99	33.21	49.49	29.39	+2.50*	+3.82*
.35	0	52.77	31.20	51.95	37.20	+0.81	-6.00*
.35	0.3	52.91	33.44	51.95	37.20	+0.96	-3.76*
.35	1	52.85	34.54	51.95	37.20	+0.89	-2.66*
.35	3	52.91	35.47	51.95	37.20	+0.96	-1.73*
.35	10	52.26	36.16	51.95	37.20	+0.31	-1.05
.35	30	52.34	36.23	51.95	37.20	+0.39	-0.98
.50	0	54.34	37.81	55.09	43.65	-0.75	-5.84*
.50	0.3	54.06	38.01	55.09	43.65	-1.03	-5.63*
.50	1	53.94	38.44	55.09	43.65	-1.15	-5.21*
.50	3	53.64	38.85	55.09	43.65	-1.45*	-4.79*
.65	1	54.93	42.19	–	–	–	–

Table 13: Free-running generation (%): pass@1 for HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021), strict exact match for GSM8K. Every row uses the same first 100 examples of each task, so rows are directly comparable (95% intervals are about ± 10 points); baseline runs on fewer examples are omitted. Mistral-7B HumanEval is excluded: our HumanEval setup fails for Mistral-7B-Instruct-v0.1 even for the dense model (2–3%), so the column carries no information about compression. At $B = 0.50$ every training-free baseline preserves generation better than ReComp.

Model	Setting	n	HumanEval	MBPP	GSM8K
Qwen2.5-7B	Dense	100	78	–	86
Qwen2.5-7B	ReComp $B = .35$	100	60	–	54
Qwen2.5-7B	ReComp $B = .50$	100	62	–	67
Qwen2.5-7B	WINA $B = .50$	100	77	–	80
Qwen2.5-14B	Dense	100	80	–	92
Qwen2.5-14B	ReComp $B = .35$	100	57	–	69
Qwen2.5-14B	ReComp $B = .50$	100	69	–	84
Mistral-7B	Dense	100	–	33	28
Mistral-7B	ReComp $B = .35$	100	–	31	17
Mistral-7B	ReComp $B = .50$	100	–	32	22
Mistral-7B	TEAL $B = .50$	100	–	39	30
Mistral-7B	WINA $B = .50$	100	–	38	32
Llama-3.1-8B	Dense	100	73	62	70
Llama-3.1-8B	ReComp $B = .35$	100	41	33	33
Llama-3.1-8B	ReComp $B = .50$	100	54	41	52
Llama-3.1-8B	R-Sparse $B = .50$	100	67	58	71
Llama-3.1-8B	TEAL $B = .50$	100	72	64	70
Llama-3.1-8B	WINA $B = .50$	100	72	65	69

A.6 Memory traffic and latency

Cost ReComp is not training-free: per budget, it updates 49.6M parameters on Qwen2.5-7B (under 1% of the model) over 3.07M token positions, 1.8–7.7 GPU-hours on one RTX 5090 for all four budgets of a 7–14B model. At batch size one, decoding is memory-bound, and a fused kernel realizes the saving: a compressed Qwen2.5-7B MLP layer runs 2.91× faster than dense at $k/m = 0.25$ (Table 14). Because only the middle MLPs are compressed and attention is untouched, $B = 0.25$ is 0.61–0.65 of whole-model arithmetic, so end-to-end decoding gains are far smaller. Because selected rows are not shared across a batch, the kernel is slower than dense at batch 16.

Table 14: **Memory traffic and measured decode latency for one MLP layer**, fp16, on an RTX 5060 Ti whose sustained bandwidth we measure at 397 GB/s, under CUDA graphs. Bytes moved are counted term by term from the implementation. *Prototype*: Triton (Tillet et al., 2019) gate/up rows with PyTorch glue and an `embedding_bag` down projection. *Fused*: gate/up rows with the SwiGLU product fused, and a split-K row-gather down projection. Times in ms at batch 1; *realis.* is roofline time over measured time for the fused path (dense runs at 1.04, the calibration check). Random weights at the paper's ranks $p = 64$, $q = 384$; the fused path matches a dense masked computation to fp16 rounding. At batch 4 and 16 the speedup is 0.46–1.02× and 0.13–0.33×, because selected rows are not shared across the batch. Qwen2.5-0.5B is omitted: its 26 MB layer fits in the 32 MB L2 cache, so its timings do not reflect decoding. Rows are indexed by the exact-neuron fraction k/m, not by B; the benchmark uses an earlier bilinear predictor with separate rank-p gate and up scorers, costing $2p(d+m)$ rather than Eq. 7's $p(d+m)$, so its arithmetic is 0.007 above the charged budget and the timings are slightly conservative.

Model	k/m	arith.	bytes	dense	prototype	fused	speedup	realis.
Qwen2.5-7B	0.25	0.278	0.279	0.989	1.847	0.339	2.91×	0.85
Qwen2.5-7B	0.35	0.378	0.379	0.989	2.713	0.435	2.27×	0.90
Qwen2.5-7B	0.50	0.528	0.529	0.989	3.950	0.582	1.70×	0.94
Qwen2.5-7B	0.65	0.678	0.679	0.989	5.150	0.728	1.36×	0.96

A.7 Further analysis

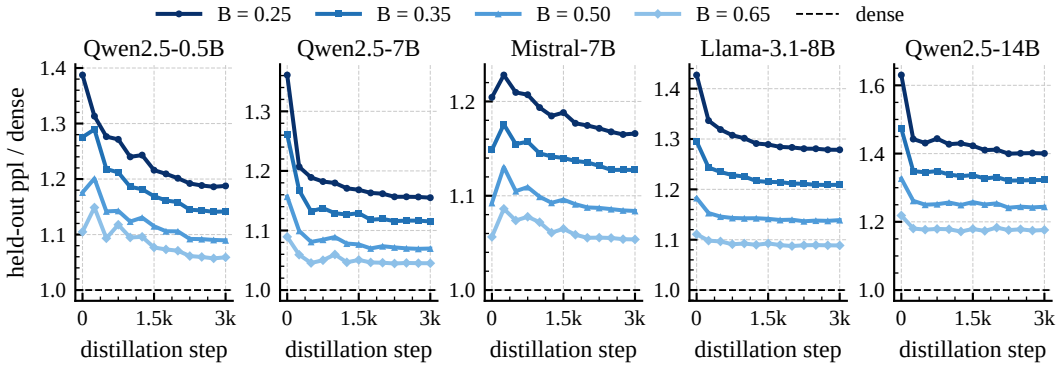

Figure 7: **Distillation dynamics for every model and budget** ($\lambda_J = 1$): Perplexity on held-out WikiText-2 validation text, relative to dense, from the closed-form initialization (step 0) through 3,000 distillation steps. Distillation improves on the closed-form fit in every case, most at the tightest budget (e.g. Qwen2.5-14B at $B = 0.25$: 1.63 to 1.40), and most of the gain arrives within the first 1,000 steps. On Mistral-7B (learning rate 10^{-5}; 10^{-4} diverged), perplexity rises briefly over the first few hundred steps before falling.

Rank allocation: At matched arithmetic, larger ranks trade exact neurons for selection and reconstruction quality. On Qwen2.5-0.5B $(p,q) = (64,384)$ gives the lowest local error among the settings we tested; on Qwen2.5-7B the error keeps falling through $(256,1024)$, so the fixed default is not optimal at 7B and the main results may understate per-model tuning.

Per-layer allocation: A per-layer budget search is orthogonal and would likely help every method. The one criterion we tried, a J-lens-driven per-layer neuron count, was *worse* than a uniform budget at identical arithmetic on Qwen2.5-0.5B (1.276 versus 1.251, in an earlier configuration of the pipeline whose ratios are not comparable to Table 1), so we report uniform allocation for every method.

Predictor calibration data: The main runs train the predictor on 43,200 rows per layer. On one Qwen2.5-7B layer, quadrupling the rows from 10,800 to 43,200 raises held-out captured importance only from 0.680 to 0.708, while the train–held-out gap closes (0.054 to 0.014): the predictor is limited by capacity, not data. Since the downstream gap is a selection gap with about ten MMLU points of oracle headroom at $B = 0.35$, stronger predictors are the most direct route to improving ReComp at moderate budgets.

Reproducibility across machines: The 7B pipeline was rebuilt from scratch on a second machine (a 16 GB GPU with layer streaming) and reproduces the reported Qwen2.5-7B numbers within 0.2%: at $B = 0.25$, oracle 1.081, top-k 2.344 and ridge 1.354, against 1.081, 2.347 and 1.356. The second run also reproduces downstream accuracy at $B = 0.25$ (MMLU 59.44 against 59.23), and a compact reimplementation in the supplementary material and Anonymous Github reproduces every Qwen2.5-0.5B perplexity within 0.3%.