REFIT: RECOVERING BLOCK-SPARSE FFN QUALITY AT NO EXTRA INFERENCE COST

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context tasks require processing many input tokens before generation, making efficient LLM prefill increasingly important. Sharing FFN neuron sets across tokens preserves efficient batched computation, but limits adaptation to each token and degrades quality. Individual-neuron energy scores omit interactions between discarded contributions, while a block-constant correction cannot recover token-dependent errors. We propose **ReFiT**, a closed-form refit of the FFN down-projection weights to reconstruct missing output from surviving activations under a fixed selection policy. The update performs ridge regression of the discarded output on retained activations, regularized toward the original weights. These activations remain token-specific under a shared mask, allowing one fitted down-projection matrix to produce different corrections for different tokens. The update folds into the existing matrix, adding no compensation branch or matrix multiplication to sparse inference. An exact mean/within-block error decomposition and held-out measurements verify recovery of both block-average and token-dependent residuals. With predictor parameters fixed, ReFiT improves block-shared diagnostic perplexity at all 24 operating points across four models and improves generation quality on LongBench. On LLaMA2-7B at 90% sparsity with 128-token blocks, LongBench improves from 12.90 to 21.11, recovering 51.8% of the score lost relative to the dense model.

1 INTRODUCTION

Long-context tasks require processing many input tokens before generation, making efficient prefill increasingly important. Activation sparsity reduces FFN computation by retaining only selected channels for each input (Liu et al., 2023; 2025a). For multi-token processing, however, different supports across tokens fragment batched matrix multiplication. Sharing a support across a token block restores regular computation, but forces tokens with different activation patterns to use the same channels. FastForward addresses this setting with predictive intermediate-neuron selection (Gautam et al., 2026); LaRoSA makes projection inputs more amenable to sparsification through rotation (Liu et al., 2025b). These approaches provide effective ways to select or represent sparse activations, while leaving room to understand and repair the token-dependent errors introduced when supports are shared.

The difficulty is that a channel unimportant to most tokens can still be useful to another token in the block. A shared mask therefore discards different contributions for different tokens, and larger blocks or higher sparsity can magnify the quality loss. Existing output compensators use mean, low-rank, or learned auxiliary contributions (Lee et al., 2024b; Zhang et al., 2025; Gautam et al., 2026); weight reconstruction offers a complementary route by adapting the existing projections (Frantar et al., 2022; Frantar & Alistarh, 2023; Yin et al., 2025). The opportunity is that sharing a mask still leaves different activation values for each token. These values may retain information about the missing output, even when the selected channels stay unchanged.

We propose **ReFiT**, a tool for refitting down-projection weights that uses this information to mitigate block-sparsity error. At each operating point, ReFiT fits one down-projection weight matrix per layer across calibration blocks, anchoring the solution to the original weights. Its closed-form update regresses the sparse FFN's output residual on its actual intermediate activations. The same matrix thus produces token-dependent corrections and folds into the existing projection, without an

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.

auxiliary inference branch. Local reconstruction analyses show recovery beyond a block-average correction, supporting the use of retained activation values to repair part of the error. ReFiT complements sparsity selection and output compensation by adapting how the existing sparse features are read out.

We evaluate this approach in two representative activation-sparsity families: FastForward-based predictive intermediate sparsity across token block sizes, and a block-shared extension of LaRoSA-based rotated input sparsity. Paper-reference implementation checks precede the controlled ReFiT comparisons (§5.1). With frozen predictors, ReFiT improves diagnostic perplexity in all 24 operating points across four models. On LLaMA2-7B at 90% sparsity with 128-token blocks, LongBench rises from 12.90 to 21.11, recovering 51.8% of the gap to dense quality. Down-projection adaptation also lowers diagnostic perplexity in 16 of 18 tested input-sparsity pairs, including settings without an intermediate mask. The results establish useful, though incomplete, quality recovery under both forms of block-sparse computation.

Our contributions are:

- **Refitting down-projection weights for block sparsity.** We formulate an anchored fit that repairs sparse FFN outputs through the existing down-projection, preserving the sparse operation structure.
- **Understanding token-dependent repair.** We characterize shared-support errors and use local reconstruction controls to show that retained activations support more than a block-average correction.
- **Validation across sparsity families.** We demonstrate quality recovery with FastForward-based predictors and a LaRoSA-based block input-sparsity extension, across block sizes, sparsities, and models.

2 Related Work

Prefill acceleration and FFN computation. Prompt processing benefits from KV-cache reuse and attention optimization (Zheng et al., 2024; Yao et al., 2025; Jones et al., 2026), while FFN projections remain a substantial cost at short-to-moderate context lengths (Gautam et al., 2026). Activation sparsity offers complementary savings, but prefill requires efficient batched computation across tokens as well as reduced arithmetic.

Activation sparsity and shared supports. Deja Vu predicts active neurons (Liu et al., 2023), while CATS and TEAL use activation thresholding (Lee et al., 2024a; Liu et al., 2025a). GRIFFIN reuses prompt-selected neurons during generation (Dong et al., 2024), and BlockFFN promotes chunk-level locality (Song et al., 2025). FastForward predicts a common intermediate-neuron set for each prefill block (Gautam et al., 2026); LaRoSA rotates projection inputs to make them more suitable for sparsification (Liu et al., 2025b). We use these last two families to study down-projection refitting under block-shared supports: the common support enables regular computation, but cannot retain every token's preferred contributions.

Weight reconstruction and output compensation. Optimal Brain Compression and SparseGPT reconstruct outputs after weight compression (Frantar et al., 2022; Frantar & Alistarh, 2023); DuoGPT incorporates activation-aware calibration into joint weight and activation sparsity (Yin et al., 2025). Output compensation instead adds expert-mean, low-rank, or learned contributions (Lee et al., 2024b; Zhang et al., 2025; Gautam et al., 2026). ReFiT applies reconstruction to a fixed block-sparse policy, fitting the existing down-projection across changing supports. Its corrections depend on the actual activations supplied by that policy and are absorbed into the down-projection. This provides a complementary repair mechanism without an additional inference branch; the following analysis examines why those activations can support token-dependent correction.

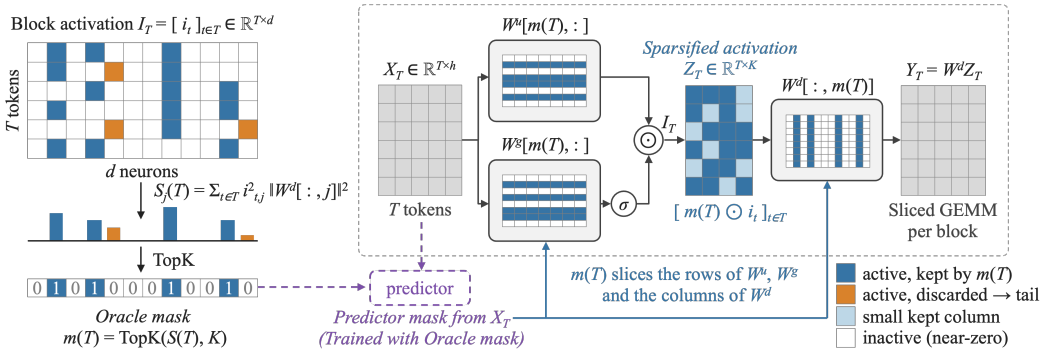

Figure 1: Block-shared intermediate sparsity. Actual activations determine the oracle support (left); predicting it before the FFN also enables up/gate row sparsification (right). Tokens share selected neurons but retain different activation values. Orange entries contribute discarded output; blue shades indicate retained values. Here T denotes block size and labels the current block in the schematic.

3 Understanding Errors under Block Sparsity

3.1 Shared Support and Token-Specific Activations

FFN computation and support selection. A LLaMA-style FFN computes, for one token with input $x_t \in \mathbb{R}^h$,

$$u_t = W^u x_t, \qquad a_t = \sigma(W^g x_t)\,, \qquad i_t = u_t \odot a_t \in \mathbb{R}^d, \qquad y_t^* = W^d\, i_t, \tag{1}$$

where $W^u, W^g \in \mathbb{R}^{d\times h}$ and $W^d \in \mathbb{R}^{h\times d}$ are the up, gate, and down projections, and σ is the activation function. Intermediate sparsity retains selected entries of i_t (Dong et al., 2024; Gautam et al., 2026; Szatkowski et al., 2026). As shown in Figure 1, one mask is shared across a block $\mathcal{B}$ of T consecutive tokens, retaining $K = \lfloor (1-s)d \rfloor$ neurons at sparsity s. Our oracle reference selects by weight-aware activation energy:

$$\xi_{t,j} = i_{t,j}^2 \left\| W^d[:, j] \right\|_2^2, \qquad S_j(\mathcal{B}) = \sum_{t\in\mathcal{B}} \xi_{t,j}, \qquad m(\mathcal{B}) = \mathrm{TopK}(S(\mathcal{B}), K)\,, \tag{2}$$

where TopK returns a binary mask with support $S_{\mathcal{B}} = \{j : m_j(\mathcal{B}) = 1\}$.

Oracle selection sparsifies W^d columns after computing i_t; predicting support also permits row sparsification of W^u, W^g. FastForward uses attention pooling and an MLP trained on dense-activation targets (Gautam et al., 2026), whereas LaRoSA sparsifies rotated projection inputs (Liu et al., 2025b). Both supply a down-projection input z_t: $m(\mathcal{B}) \odot i_t$ for intermediate masking, or activations computed from sparse inputs. ReFiT fits the down-projection to reduce their residual $e_t = y_t^* - W^d z_t$.

Token-specific activation patterns. Figure 2 uses fixed LLaMA2-7B dense-model activations from 166 non-overlapping 2048-token WikiText-2 test windows, excluding the first and last 128 tokens. Masks follow Eq. (2) without being installed in the model. For tokenwise top-K set S_t and block support $S_{\mathcal{B}(t)}$, define mean support recall R and pooled discarded energy fraction D over N evaluated tokens:

$$R(T, s) = \frac{1}{N} \sum_t \frac{|S_t \cap S_{\mathcal{B}(t)}|}{K}, \tag{3}$$

$$D(T, s) = \frac{\sum_t \sum_{j\notin S_{\mathcal{B}(t)}} \xi_{t,j}}{\sum_t \sum_j \xi_{t,j}}. \tag{4}$$

Heatmaps show 128 tokens from the first window; statistics use all windows.

At layer 16 and $s = 0.9$, the union of tokenwise top-K sets averages $9.9K$ neurons per 128-token block, far exceeding its shared budget K. Recall consequently falls from 1 at $T = 1$ to 0.26 at

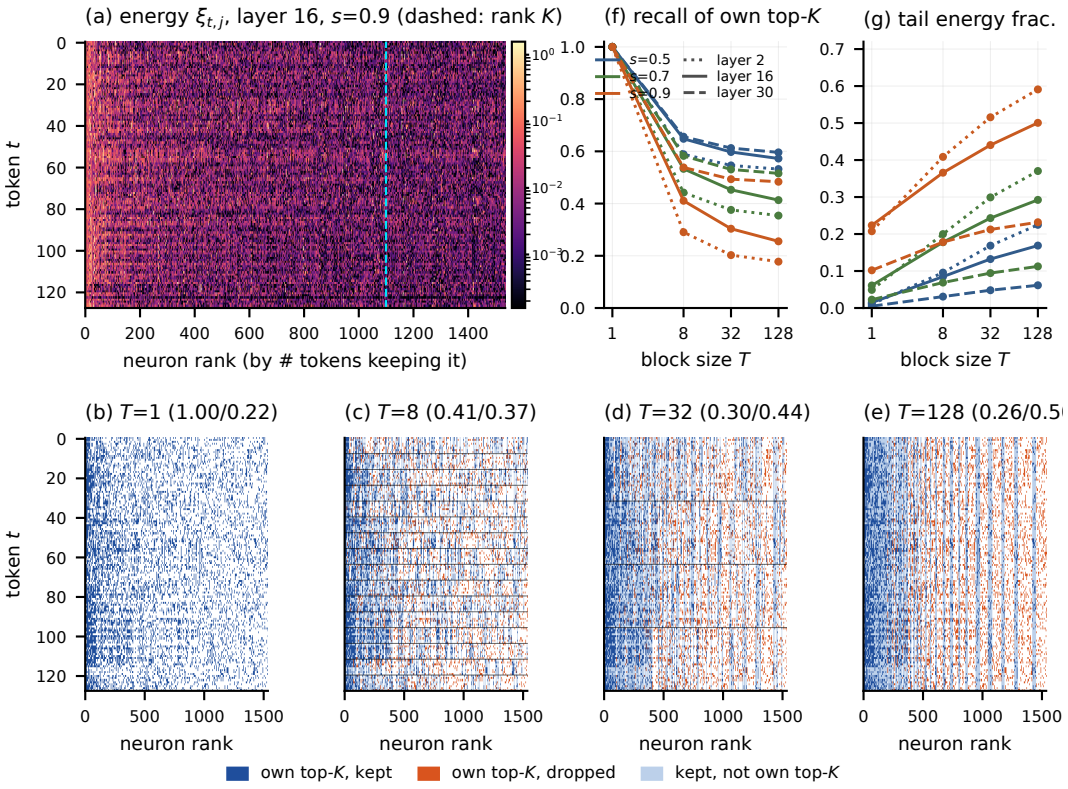

Figure 2: Token-specific support under block sharing (LLaMA2-7B, WikiText-2). **(a)** Activation energy and **(b–e)** shared masks for $T = 1, 8, 32, 128$ at layer 16, $s = 0.9$. Dark blue/orange mark retained/dropped tokenwise top-K neurons; light blue marks retained neurons outside that set. The dashed line marks frequency rank K, not an oracle boundary. **(f,g)** Support recall and discarded diagonal energy across layers and sparsities, aggregated over 166 windows. Larger blocks lose more token-specific support and energy.

$T = 128$. Discarded energy also increases with block size and sparsity. These selection statistics expose the cost of shared support, but actual output error depends on interactions between discarded contributions. We next examine this residual and its token-dependent structure.

3.2 The reconstruction problem under shared support

Sharing a support drops the same channels, but their contributions differ across tokens. For intermediate masking, the residual is the discarded output, or *tail*:

$$e_t = y_t^* - W^d z_t = W^d[(1 - m(\mathcal{B})) \odot i_t] =: \text{tail}_t. \tag{5}$$

The mask fixes which channels are discarded, but their values vary across tokens, as illustrated in Figure 2. With $q = 1 - m(\mathcal{B})$ and block means $\bar{i}_\mathcal{B}$ and $\bar{e}_\mathcal{B}$ for activations and residuals, respectively,

$$e_t - \bar{e}_\mathcal{B} = W^d[q \odot (i_t - \bar{i}_\mathcal{B})]\,. \tag{6}$$

This variation is the within-block component measured in §3.3. Its magnitude depends on output directions and interactions, which the diagonal-energy statistic D omits. Retained activations z_t also vary, providing possible predictors of the residual; whether they support token-dependent correction is tested in §4.2 and Appendix B.2.

Block sparsity characteristics. Figure 3 shows two trends in the unrepaired LLaMA2-7B curves with FastForward-based predictors. **(1) Larger blocks increase PPL degradation:** sharing one support across more tokens limits adaptation to their activation patterns; at $s = 0.7$, PPL rises from

15.44 to 20.81 as T grows from 8 to 128. **(2) Higher sparsity amplifies degradation:** retaining fewer neurons worsens PPL and increases its absolute sensitivity to block size; at $T = 128$, PPL rises from 8.91 to 20.81 to 614.03 for $s = 0.5, 0.7, 0.9$. These observations use the common WikiText-2 evaluation protocol in §5.1.

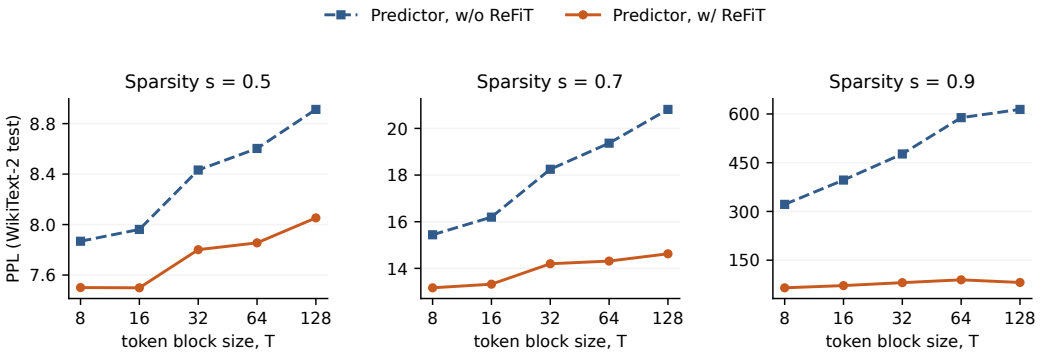

Figure 3: Block-size and sparsity trends with FastForward-based predictors on LLaMA2-7B. Dense PPL is 5.47 (not plotted). Unrepaired PPL increases with both T and s; paired curves use the same predictor before and after ReFiT. Each panel uses a separate linear y-axis spanning its observed PPL range, so vertical distances are not directly comparable across panels. Values are block-shared diagnostic PPL on WikiText-2 test with fixed 128-token dense prefix/suffix spans and one calibration seed; protocol details are in §5.1.

These trends motivate reducing block-sparsity error while preserving shared computation. Our proposed ReFiT substantially lowers PPL and narrows its range across block sizes in Figure 3. We next examine the error structure that motivates this reconstruction, before presenting the method in §4.

3.3 The Error Structure of Shared Neuron Sets

Most discarded output varies across tokens. Fix one layer's inputs and original weights. For a block $\mathcal{B}$, let e_t denote its residual and $\bar{e}_{\mathcal{B}}$ its mean over $n_{\mathcal{B}}$ valid tokens. The squared error decomposes exactly as

$$E_{\mathcal{B}} = \underbrace{n_{\mathcal{B}} \|\bar{e}_{\mathcal{B}}\|^2}_{E^0_{\text{mean},\mathcal{B}}} + \underbrace{\sum_{t \in \mathcal{B}} \|e_t - \bar{e}_{\mathcal{B}}\|^2}_{E^0_{\text{within},\mathcal{B}}}. \tag{7}$$

Table 1 shows that within-block variation accounts for 69–91% of local error under the actual-activation score policy, using the separate validation protocol in Appendix B. A constant correction per block cannot remove this component. Dense inputs come from the original dense model; raw-sparse inputs come from the unrefitted sparse model. Each local comparison holds these inputs fixed.

Individual energies do not capture the full error. Discarded neuron outputs are vectors whose contributions can reinforce or cancel. Writing $E_{\mathcal{B}} = E_{\text{diag},\mathcal{B}} + E_{\text{cross},\mathcal{B}}$, the first term sums individual energies from Eq. (2), while the second contains signed interactions. Cross terms account for 8–24% of the measured error (Table 1; derivation in Appendix C). This motivates fitting the full output residual rather than approximating it through individual energies. The cross and within shares are separate decompositions and are not additive.

These findings identify what a correction must capture, but not whether it is predictable. ReFiT asks whether the retained activation values contain information about this residual, then tests that possibility through local reconstruction (§4.2).

		Cross / total error (%)		Within / total error (%)	
s	T	Dense	Raw-sparse	Dense	Raw-sparse
0.7	16	9.89	9.13	81.62	81.36
0.7	32	9.67	8.91	85.89	85.52
0.7	64	9.29	8.60	88.66	88.17
0.7	128	8.89	8.31	90.52	89.87
0.9	16	23.76	19.59	71.66	68.67
0.9	32	23.14	19.45	76.69	73.12
0.9	64	22.61	19.74	79.98	75.70
0.9	128	22.20	19.96	82.16	77.25

Table 1: Error structure before down-projection refitting on LLaMA2-7B, WikiText-2 validation. Cross and within shares are respectively $100 \sum E_{\text{cross},\mathcal{B}} / \sum E_{\mathcal{B}}$ and $100 \sum E^{0}_{\text{within},\mathcal{B}} / \sum E_{\mathcal{B}}$, pooling native blocks across 32 layers and 130 reporting windows before division. Dense and raw-sparse refer to layer-input streams. The two shares describe different decompositions and must not be added. These are local squared-error statistics, not PPL.

4 REFIT

4.1 DOWN-PROJECTION FITTING AND RESIDUAL REGRESSION

For the sparse activation z_t defined in §3.1, ReFiT fits the down-projection to the original unmasked output y_t^* at the same local input. The selection policy is fixed, and one matrix per layer is fitted across calibration blocks at each operating point:

$$\tilde{W}^d = \arg\min_W \sum_t \|y_t^* - W z_t\|^2 + \mu \|W - W^d\|_F^2. \tag{8}$$

The anchor $\mu > 0$ controls departure from the original down-projection weights. With $G = \sum_t z_t z_t^\top$ and $C = \sum_t y_t^* z_t^\top$, the solution is

$$\tilde{W}^d = (C + \mu W^d)(G + \mu I)^{-1}. \tag{9}$$

Calibration activations and targets are fixed during this solve; Appendix A gives its derivation and scope.

Token-dependent correction through a shared matrix. Writing $e_t = y_t^* - W^d z_t$ gives

$$\tilde{W}^d = W^d + \Delta W, \qquad \Delta W = \Big(\sum_t e_t z_t^\top\Big)(G + \mu I)^{-1}. \tag{10}$$

Thus ReFiT regresses the full residual on the actual sparse-computation features. For intermediate masking, the residual is the discarded tail; for input sparsity, it also reflects changes to upstream activations. Although the support is shared, z_t varies across tokens, allowing $c_t = \Delta W z_t$ to provide different corrections. The fitted matrix replaces the existing down-projection, with no added sparse-path matrix multiplication. Calibration and deployment settings are given in §5.1.

4.2 HOW REFIT REDUCES RECONSTRUCTION ERROR

We evaluate local reconstruction with actual-activation masks on LLaMA2-7B, using the same layer inputs and masks before and after repair. Applying Eq. (7) to $r_t = e_t - c_t$ measures how much block-mean and within-block error remain. The analysis covers dense and raw-sparse inputs; the evaluation protocol and full policy grid are detailed in Appendix B.

Correction beyond the block average. Table 2 compares full ReFiT with its block-average correction and a within-block shuffle of token corrections. At $T = 32$, $s = 0.9$, full ReFiT reduces total error by 26–27%. Keeping only its average correction loses part of this gain; shuffling preserves the average but disrupts the token-specific alignment and removes much of the benefit. ReFiT recovers

Local reconstruction	Dense input	Raw-sparse input
No correction (SSE)	15.971	15.257
Block-average correction (SSE)	13.666	12.841
Shuffled correction (expected SSE)	15.697	14.469
Full ReFiT (SSE)	**11.829**	**11.116**
Block-mean error reduction (%)	61.91	58.92
Within-block error reduction (%)	15.00	15.46

Table 2: Local correction at $T = 32$, $s = 0.9$ on LLaMA2-7B, WikiText-2 validation. SSE is normalized by summed unmasked output energy and multiplied by 100 (lower is better). Controls use the same ReFiT checkpoint: block-average correction removes token dependence; shuffle reports the analytical permutation expectation. Reduction rates use each component's unrepaired energy.

59–62% of block-mean error and about 15% of within-block error, showing that retained features predict more than a common offset. Both components improve across the tested policies.

Recovery is concentrated in the final layer, but excluding it still leaves 5.68–7.89% within-block recovery. These are local output measurements; model-level quality is evaluated separately in Sec. 5. Appendix B.2 includes the layer analysis and an optimized constant-correction reference.

5 Experiments

5.1 Setup and evaluation protocols

Baselines and controlled comparisons. We build on paper-reference implementations of FastForward and LaRoSA (Gautam et al., 2026; Liu et al., 2025b). At 50% sparsity, checks obtain LaRoSA PPL 5.83 (reported 5.87) and FastForward LongBench 29.31 (reported 28.37); evaluation differences and execution specifications are documented in Appendix D. We then isolate ReFiT under block-shared policies: FastForward-based predictors use uniform sparsity without an auxiliary compensator, and LaRoSA-based input selection uses shared block masks. Each before/after pair fixes the policy, block size, and sparsity, changing only the down-projection. This tests a complementary repair tool rather than a new selector.

Models and fitting. Experiment 1 uses LLaMA2-7B with $T \in \{8, 16, 32, 64, 128\}$; Experiment 2 uses LLaMA3.2-1B, LLaMA3.2-3B (both Instruct), and Qwen3-4B at $T = 128$. Experiment 3 tests rotated input sparsity on LLaMA2-7B at $T \in \{8, 16, 32\}$. All use $s \in \{0.5, 0.7, 0.9\}$. At each operating point, we collect (z_t, y_t^*) from the original sparse trajectory, solve Eq. (9) per layer, and install the fitted down-projection matrices. Targets use the same local inputs as the sparse features. E1/E2 use 128 sequences of 2048 WikiText-2 train tokens, seed 42, and relative regularization 1; E3 uses 16 sequences, relative regularization 0.001, and a 0.1-scaled update. Original weights serve the fixed 128-token dense prefix/suffix spans in E1/E2; E3 has no dense boundary exceptions. Detailed collection, damping, and inference rules are in Appendices D.1 and D.2.

Quality metrics. We report WikiText-2 test *diagnostic PPL*: current-block selection can use later tokens in that block, so this is not standard causal perplexity. LongBench measures generation after sparse prefill with dense decoding, using the unweighted mean over 16 English tasks and matched dense controls. E1 and E2 use different context caps and prompt templates; comparisons are within each model/setup. E3 is evaluated by PPL only. All fits use one calibration seed. Full scoring protocols and sampling analysis are in Appendix D.3.

5.2 Quality recovery across block sizes and sparsities

Recovering generation quality. All nine measured LongBench pairs in Table 3 improve. At $s = 0.9$, ReFiT gains 4.65–8.21 points across $T = 8, 32, 128$. For $T = 128$, the score rises from 12.90 to 21.11 against dense 28.75, recovering 51.8% of the lost score, computed as $(L_{\mathrm{ReFiT}} - L_{\mathrm{sparse}})/(L_{\mathrm{dense}} - L_{\mathrm{sparse}})$. At $s = 0.7$, gains are 1.66–2.13 points. The benefit therefore extends

		$s = 0.5$		$s = 0.7$		$s = 0.9$	
T	Method	PPL ↓	LB ↑	PPL ↓	LB ↑	PPL ↓	LB ↑
8	w/o ReFiT	7.87	27.80	15.44	24.91	321.71	15.84
	w/ ReFiT	7.50	28.22	13.17	26.59	64.90	20.49
32	w/o ReFiT	8.43	27.86	18.25	24.14	476.93	14.67
	w/ ReFiT	7.80	27.91	14.20	25.80	80.79	20.66
128	w/o ReFiT	8.91	27.76	20.81	23.74	614.03	12.90
	w/ ReFiT	8.05	28.05	14.63	25.87	81.37	21.11

Table 3: LLaMA2-7B with FastForward-based predictors: WikiText-2 diagnostic PPL and LongBench (LB, 16-task mean). Dense: PPL 5.47, LB 28.75. All nine combinations of $T = 8, 32, 128$ and $s = 0.5, 0.7, 0.9$ are evaluated. LB uses a 3,500-token prompt cap, sparse prefill, and dense decoding.

		$s = 0.5$		$s = 0.7$		$s = 0.9$	
Model	Method	PPL ↓	LB ↑	PPL ↓	LB ↑	PPL ↓	LB ↑
LLaMA3.2-1B	w/o ReFiT	26.72	29.86	62.50	25.53	1039.95	17.22
	w/ ReFiT	22.84	30.18	41.79	27.95	193.07	21.47
LLaMA3.2-3B	w/o ReFiT	19.73	42.85	50.43	32.54	1794.52	19.47
	w/ ReFiT	18.63	44.93	40.37	38.60	224.39	23.48
Qwen3-4B	w/o ReFiT	21.60	45.51	45.49	36.69	1207.12	20.26
	w/ ReFiT	16.88	46.65	24.91	42.81	184.44	25.84

Table 4: FastForward-based predictors, $T = 128$: WikiText-2 diagnostic PPL and LongBench (LB, 16-task mean). Dense PPL/LB in model order: 13.16/32.00, 11.05/45.79, 13.66/47.88. LB uses a 31,500-token prompt cap, model chat templates, sparse prefill, and dense decoding. The LLaMA3.2 models use Instruct checkpoints.

to generation, while leaving a gap to dense quality. At $s = 0.5$, gains are 0.42, 0.05, and 0.29 points for $T = 8, 32, 128$, respectively. The $T = 32$, $s = 0.5$ paired prompt-bootstrap interval includes zero, unlike its $s = 0.7, 0.9$ intervals (Appendix D.3).

Reduced sensitivity to block size. At $s = 0.9$, the LongBench range across $T = 8, 32, 128$ narrows from 2.94 to 0.62 points after refitting; at $s = 0.7$, it narrows from 1.17 to 0.79. At $s = 0.5$, it instead increases from 0.102 to 0.312 points, so reduced generation-score variation is specific to the higher sparsities here. ReFiT also improves PPL at all 15 operating points in Figure 3, with narrower PPL and mean-NLL ranges (Appendix D.3). These observations show that down-projection adaptation can mitigate the quality cost of sharing larger blocks, without requiring monotone fitted curves or complete recovery.

Actual-activation control. The effect also appears without learned prediction: ReFiT improves all 15 PPL pairs using the actual-activation score of Eq. (2). Thus its benefit is not restricted to learned-predictor errors; the control and full values are in Appendix D.1.

5.3 Applicability across models

The same procedure for refitting down-projection weights improves quality beyond LLaMA2-7B (Table 4). At $s = 0.7$, LongBench rises from 25.53 to 27.95 for LLaMA3.2-1B, 32.54 to 38.60 for LLaMA3.2-3B, and 36.69 to 42.81 for Qwen3-4B, recovering 37.3%, 45.7%, and 54.7% of their respective dense-score gaps. Each model uses its own fitted down-projection matrices, with the same objective and calibration recipe. Gains depend on the operating point, and substantial loss remains at $s = 0.9$; PPL alone does not determine generation quality. The comparison tests applicability across models rather than a model-size scaling law.

	UG		UGD	
T	w/o ReFiT	D	w/o ReFiT	D
8	32.32	**28.55**	38.64	**35.48**
16	45.36	**38.89**	60.19	**52.10**
32	55.36	**46.49**	83.27	**69.13**

Table 5: Down-projection refitting across block sizes.

	Refitted projections ($T = 32$)			
Inputs	w/o ReFiT	SEP	D	SEP-D
UG	55.36	51.90	46.49	44.18
UGD	83.27	79.00	69.13	66.58

Table 6: Projection-wise refit ablation.

Tables 5–6: LLaMA2-7B, $s = 0.7$, $\alpha = 0.001$, $\beta = 0.1$; diagnostic PPL $\downarrow$. Dense PPL is 5.47. w/o ReFiT retains the original weights; SEP refits up/gate, D (ReFiT-D) refits down, and SEP-D refits down on the SEP parent's recollected activations. Bold marks lower PPL within a pair.

5.4 Down-projection refitting for rotation-based input sparsity

We next apply ReFiT to a block-shared LaRoSA-based policy. After a fixed FFN-input rotation, **UG** masks the up/gate inputs using one shared support; **UGD** also masks the intermediate down input. **D** fits only down, **SEP** fits up/gate, and **SEP-D** fits down on activations recollected after SEP. Each paired comparison keeps rotation and selection fixed; equal nominal sparsity in UG and UGD does not imply equal compute. Protocols and complete grids are in Appendix D.2.

At $s = 0.7$, D lowers PPL at all three block sizes in both policies (Table 5). UG benefits even without an intermediate mask: at $T = 32$, PPL falls from 55.36 to 46.49, showing that the down-projection can partly compensate for activations changed by upstream sparsity. At this operating point, D improves more than SEP, while SEP-D further reduces PPL to 44.18 for UG and 66.58 for UGD (Table 6). This supports a complementary role for down-projection refitting alongside upstream adaptation.

Across $T = 8, 16, 32$, D improves nine of nine UG and seven of nine UGD pairs. The two small UGD regressions at $s = 0.5$ and the larger-block $s = 0.9$ failures remain in Appendix D.2. The results extend ReFiT's usefulness to block input sparsity, with recovery conditional on the sparse policy and operating point.

6 Conclusion

ReFiT complements sparse selection by refitting FFN down-projection weights at no extra inference cost. It recovers token-dependent errors and improves PPL and LongBench quality with FastForward-based intermediate sparsity, with further PPL gains under LaRoSA-based input sparsity. Recovery remains partial and configuration-dependent.

AI use statement

Generative AI tools assisted with GPU resource management and experiment orchestration, code implementation and debugging, analysis and presentation of experimental results, feedback on experimental procedures, and manuscript drafting and revision, including proof exposition. The core research ideas and overall research direction originated with the authors, who retained control over scientific decisions and conclusions. The authors are responsible for verifying AI-assisted material and for the correctness, originality, and integrity of the final work.

Reproducibility statement

Our description of ReFiT connects the objective for refitting down-projection weights and deployment procedure in Section 4 with the models, calibration settings, sparsity configurations, and evaluation protocols in Section 5. Appendix A supplies the derivation and its assumptions, while Appendix B specifies the local reconstruction measurements and controls. Detailed calibration recipes, numerical precision, runtime environments, full quality grids, and LongBench scoring and sampling procedures appear in Appendix D.

REFERENCES

Harry Dong, Beidi Chen, and Yuejie Chi. Prompt-prompted adaptive structured pruning for efficient LLM generation. *arXiv preprint arXiv:2404.01365*, 2024.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning (ICML)*, 2023. arXiv:2301.00774.

Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2208.11580.

Aayush Gautam, Mukul Gagrani, Junyoung Park, Mingu Lee, Chris Lott, and Narasimha Reddy. FastForward: Accelerating LLM prefill with predictive FFN sparsity. *arXiv preprint arXiv:2602.00397*, 2026.

Dalton Jones, Junyoung Park, Matthew Morse, Mingu Lee, Chris Lott, and M. Harper Langston. QuOKA: Query-oriented KV selection for efficient LLM prefill. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2602.08722.

Donghyun Lee, Je-Yong Lee, Genghan Zhang, Mo Tiwari, and Azalia Mirhoseini. CATS: Contextually-aware thresholding for sparsity in large language models. In *Conference on Language Modeling (COLM)*, 2024a. arXiv:2404.08763.

Jaeseong Lee, Seung-won Hwang, Wonpyo Park, and Mingi Ji. Breaking ReLU barrier: Generalized MoEfication for dense pretrained models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024b.

James Liu, Pragaash Ponnusamy, Tianle Cai, Han Guo, Yoon Kim, and Ben Athiwaratkun. Training-free activation sparsity in large language models. In *International Conference on Learning Representations (ICLR)*, 2025a. arXiv:2408.14690.

Kai Liu, Bowen Xu, Shaoyu Wu, Xin Chen, Hao Zhou, Yongliang Tao, and Lulu Hu. La RoSA: Enhancing LLM efficiency via layerwise rotated sparse activation. In *International Conference on Machine Learning (ICML)*, 2025b. arXiv:2507.01299.

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 *International Conference on Machine Learning (ICML)*, 2023. arXiv:2310.17157.

Chenyang Song, Weilin Zhao, Xu Han, Chaojun Xiao, Yingfa Chen, Yuxuan Li, Zhiyuan Liu, and Maosong Sun. BlockFFN: Towards end-side acceleration-friendly mixture-of-experts with chunk-level activation sparsity. *arXiv preprint arXiv:2507.08771*, 2025.

Filip Szatkowski, Patryk Będkowski, Alessio Devoto, Jan Dubiński, Pasquale Minervini, Mikołaj Piórczyński, Simone Scardapane, and Bartosz Wójcik. Universal properties of activation sparsity in modern large language models. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2509.00454.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. CacheBlend: Fast large language model serving for RAG with cached knowledge fusion. In *European Conference on Computer Systems (EuroSys)*, 2025. arXiv:2405.16444.

Ruokai Yin, Yuhang Li, Donghyun Lee, and Priyadarshini Panda. DuoGPT: Training-free dual sparsity through activation-aware pruning in LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2506.20194.

Zhenyu Zhang, Zechun Liu, Yuandong Tian, Harshit Khaitan, Zhangyang Wang, and Steven Li. R-Sparse: Rank-aware activation sparsity for efficient LLM inference. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2504.19449.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2312.07104.

A Derivation and Scope of the Down-projection Fit

A.1 Fixed calibration problem

For each calibration token, let $z_t \in \mathbb{R}^d$ be the actual sparse down-projection input and $y_t^* \in \mathbb{R}^h$ the original unmasked FFN output at the same local input. With intermediate masking, $z_t = m(\mathcal{B}) \odot i_t$ and $y_t^* = W^d i_t$.

Assumption 1 (Fixed regressors)**.** *Calibration inputs, targets, and masks are collected before the solve. Selection uses original scoring weights or a frozen predictor, and the regressors z_t do not depend on the matrix being optimized.*

Assumption 2 (Positive anchor)**.** *The regularization parameter satisfies $\mu > 0$.*

Define the objective and sufficient statistics as

$$J(W) = \sum_t \|y_t^* - W z_t\|^2 + \mu \|W - W^d\|_F^2, \tag{11}$$

$$G = \sum_t z_t z_t^\top, \qquad C = \sum_t y_t^* z_t^\top. \tag{12}$$

Theorem 1 (Anchored down-projection fit)**.** *Under Assumptions 1–2, J has the unique minimizer*

$$\tilde{W}^d = (C + \mu W^d)(G + \mu I)^{-1}. \tag{13}$$

Proof. For any v, $v^\top G v = \sum_t (z_t^\top v)^2 \geq 0$, so $G + \mu I \succ 0$. The objective separates over output rows, each with Hessian $2(G+\mu I)$, and is strictly convex. Its gradient vanishes when $W(G+\mu I) = C + \mu W^d$, giving the stated unique solution. □

A.2 Residual regression and the anchor

Let $e_t = y_t^* - W^d z_t$. This is the discarded intermediate contribution for intermediate masking; for input sparsity it includes the effect of changed upstream activations.

Corollary 1 (Residual-regression identity)**.** *Writing $\tilde{W}^d = W^d + \Delta W$ gives*

$$\Delta W = \Big(\sum_t e_t z_t^\top\Big)(G + \mu I)^{-1}. \tag{14}$$

Proof. Substitute $C = W^d G + \sum_t e_t z_t^\top$ into Eq. (13). □

Corollary 2 (Anchor limits)**.** *As $\mu \to \infty$, $\tilde{W}^d \to W^d$. If $G \succ 0$, then $\tilde{W}^d \to CG^{-1}$ as $\mu \to 0^+$. Replacing the anchor by $\mu\|W\|_F^2$ instead gives $C(G+\mu I)^{-1}$, whose large-μ limit is the zero matrix.*

Proof. For the large-μ limit, divide the two factors in Eq. (13) by μ; the small-μ limit follows by continuity of inversion when $G \succ 0$. The zero-anchor solution follows from the same normal equation with the μW^d term removed. □

These limits specify the regularization prior; they do not guarantee monotone improvement of held-out quality as the anchor is weakened.

A.3 Recoverable components and deployment scope

Proposition 1 (Calibration residual equation). *For* $r_t = y_t^* - \tilde{W}^d z_t$,

$$\sum_t r_t z_t^\top = \mu \Delta W. \tag{15}$$

Proof. The left side is $\sum_t e_t z_t^\top - \Delta W G = \Delta W (G + \mu I) - \Delta W G$. □

At an unregularized least-squares solution, the residual is orthogonal to the calibration design. With $\mu > 0$, Eq. (15) generally leaves a nonzero residual inner product with the regressors. A single shared linear map cannot perfectly predict distinct residuals from identical survivor vectors. More generally, if Z stacks the rows $z_t^\top$ and E stacks $e_t^\top$, exact unregularized reconstruction requires every column of E to lie in the column space of Z. These are finite-sample statements, not guarantees of held-out recovery.

A frozen policy does not freeze a network rollout: upstream refitted layers can change later inputs and realized masks. The theorem applies to the collected calibration problem. The oracle diagnostics below and the predictor-based quality evaluations test different aspects of deployment.

Numerical recipe and cost. Experiments 1 and 2 use $\mu = \mu_{\mathrm{rel}} \operatorname{mean}(\operatorname{diag} G)$ with $\mu_{\mathrm{rel}} = 1$. Both statistics and Cholesky solves use FP64. One factorization is shared across output rows; factorization and solves cost $O(d^3 + hd^2)$ per layer, in addition to collecting the statistics. Experiment 3 uses its separate regularization and damping recipe (Appendix D); its damped deployment is not the full ridge minimizer. ReFiT adds no compensation matrix multiplication on the sparse path, but calibration and storage have nonzero cost. DENSE-D0 retains original and fitted down-projection matrices for different token spans.

B Local Reconstruction and Token-Dependent Corrections

B.1 Protocol and recovery across policies

We evaluate existing policy-matched oracle ReFiT checkpoints on LLaMA2-7B at $T \in \{16, 32, 64, 128\}$ and $s \in \{0.7, 0.9\}$, using all 32 layers. WikiText-2 validation provides 130 reporting windows after reserving the historical 16 development windows. Each window has 2048 tokens; the common scored region is $[128, 1920)$, giving 232,960 tokens per condition. These windows have historical experiment exposure and are not a new confirmation set. No additional down-projection fitting or PPL evaluation is part of this analysis.

These historical checkpoints were fitted on 128 packed 2048-token WikiText-2 train sequences (seed 42), using each policy's original sparse trajectory and $\mu_{\mathrm{rel}} = 1$. Unlike the current E1/E2 recipe, their dense prefix and suffix each contain T tokens. The fit therefore uses 258,048, 253,952, 245,760, and 229,376 sparse tokens per layer for $T = 16, 32, 64, 128$, respectively. Historical statistics were accumulated in FP32 and solved by FP64 Cholesky, then stored in FP16 and loaded in BF16. Restricting evaluation to $[128, 1920)$ equalizes the reporting region, but does not equalize these calibration boundaries or fitted checkpoints.

We use two input streams. Common dense-model inputs hold each layer's input fixed across policies. Raw-sparse inputs come from each policy's rollout with the original down-projection matrices. In both streams, raw and corrected outputs are computed at the same local input and mask, against $y^* = W^d i_t$. In particular, the corrected raw-sparse measurement is a local counterfactual, not a rollout with ReFiT installed throughout the network. Different raw-sparse policies need not encounter the same layer inputs.

Table 7 reports native-block within-error recovery. Every condition has positive mean, within, and total gain in all 32 layer aggregates; this does not imply improvement in every block. At $s = 0.9$, dense-input within recovery decreases from 16.32% to 11.90% over the listed block sizes, while raw-sparse recovery stays near 15.4–15.8%. The qualitative stream difference remains with fixed 128-token analysis bins, so changing the decomposition's bin size alone does not explain it. Block-dependent historical boundary rules and upstream input changes preclude a causal interpretation

	$s = 0.7$		$s = 0.9$	
T	Dense	Raw-sparse	Dense	Raw-sparse
16	11.82	11.27	16.32	15.40
32	11.60	11.33	15.00	15.46
64	11.10	11.21	13.52	15.80
128	10.41	10.97	11.90	15.80

Table 7: Within-block local error reduction (%), aggregated across all 32 layers and reporting windows before taking ratios. Dense and raw-sparse columns evaluate different input distributions. These are observed values, not tests of monotone trends or end-to-end quality.

Local correction	Dense input	Raw-sparse input
w/o ReFiT	15.971	15.257
ReFiT block mean	13.666	12.841
Full ReFiT	11.829	11.116
Shuffled ReFiT (expectation)	15.697	14.469
Best-constant oracle	12.248	11.156

Table 8: Local SSE at $T = 32$, $s = 0.9$, normalized by summed unmasked output energy and multiplied by 100 (lower is better). Each comparison uses identical local inputs, masks, and targets. The best-constant oracle uses the true residual mean separately in every block; it is an analytical reference, not a deployable compensation method.

of these trends. The result supports recovery under shared masks across the tested settings, not a universal increase of recoverability with block size.

For $T = 32$, $s = 0.9$, window-cluster bootstrap intervals for within recovery are $[14.73, 15.25]\%$ on dense inputs and $[15.04, 15.89]\%$ on raw-sparse inputs (95%, 2000 resamples; clusters use historical window ID divided by 8, rounded down). These quantify sensitivity to the WT2 sample, not training-seed uncertainty or a significant difference between streams.

B.2 Mean-only and token-pairing controls

To isolate the contribution of token-dependent corrections, we compare full ReFiT with the same checkpoint's block-average correction and with a within-block shuffle. All variants use the local inputs and checkpoints described above; no additional down-projection is fitted.

Let $\tilde{e}_t = e_t - \bar{e}_{\mathcal{B}}$ and $\tilde{c}_t = c_t - \bar{c}_{\mathcal{B}}$ denote centered residuals and corrections. A mean-only ablation replaces c_t by $\bar{c}_{\mathcal{B}}$ from the same checkpoint. Its error is $E^R_{\text{mean}} + E^0_{\text{within}}$, so its excess over full ReFiT is exactly G_{within}. It is not a separately optimized block-bias model. For a uniformly random permutation of the corrections within the block, including permutations with fixed points,

$$A_{\mathcal{B}} = \sum_{t \in \mathcal{B}} \langle \tilde{e}_t, \tilde{c}_t \rangle, \qquad V_{\mathcal{B}} = \sum_{t \in \mathcal{B}} \|\tilde{c}_t\|^2, \tag{16}$$

$$G_{\text{within}} = 2A_{\mathcal{B}} - V_{\mathcal{B}}, \tag{17}$$

$$\mathbb{E}_{\text{perm}}[E_{\text{shuffle}}] = E^R_{\text{mean}} + E^0_{\text{within}} + V_{\mathcal{B}}. \tag{18}$$

Hence correctly paired corrections outperform this shuffle by $2A_{\mathcal{B}}$, but improve on the raw within-block error only if $2A_{\mathcal{B}} > V_{\mathcal{B}}$. Positive alignment alone does not rule out overcorrection.

Table 8 gives the representative $T = 32$, $s = 0.9$ comparison. Full correction beats mean-only and shuffled correction. The shuffle preserves the block mean, so it need not be worse than the unrepaired output. Within-block repair supplies 44.35% and 41.65% of the total recovered energy for dense and raw-sparse inputs, respectively; these gain shares differ from the within-error recovery rates in Table 2.

Best constant versus checkpoint mean. The mean-only diagnostic is not an optimized block-bias model. Allowing one constant vector $b_{\mathcal{B}}$ per block gives

$$\min_{b_{\mathcal{B}}} \sum_{t \in \mathcal{B}} \|e_t - b_{\mathcal{B}}\|^2 = E^0_{\text{within},\mathcal{B}}, \qquad b^*_{\mathcal{B}} = \bar{e}_{\mathcal{B}}. \tag{19}$$

At $T = 32$, $s = 0.9$, full ReFiT's SSE divided by this oracle's SSE is 0.965817 on dense inputs and 0.996383 on raw-sparse inputs. The latter margin is small. At $T = 16$, full ReFiT does not beat the best constant in either sparsity or input stream. Thus positive within-block recovery and superiority to the best constant in total SSE are distinct claims.

Layer concentration and scope. The pooled gains in Table 2 are not uniform across layers. The final layer contributes 26.50% of raw within-block error but 72.17% of its reduction on dense inputs; the corresponding raw-sparse shares are 22.86% and 60.61%. Excluding the final layer leaves 5.68% and 7.89% within-block recovery, respectively; median layer recovery is 5.52% and 11.54%. Both its error scale and its higher recovery rate contribute to the concentration. These local measurements do not determine how much that layer contributes to generation quality. They use oracle policies and historical calibration boundaries, and are not a mechanism measurement on the current RES/BCE checkpoints or a matched $T = 1$ comparison.

C Selection-Score Ranking Diagnostic

Individual energies and set-output error. For a block, let $I_{\mathcal{B}}$ collect intermediate activations and $q = 1 - m$ indicate discarded neurons. The local squared error is

$$E_{\mathcal{B}}(m) = \sum_{t \in \mathcal{B}} \left\| \sum_j q_j i_{t,j} W^d[:, j] \right\|^2 = q^\top Q_{\mathcal{B}} q, \tag{20}$$

$$Q_{\mathcal{B}} = ((W^d)^\top W^d) \odot (I_{\mathcal{B}}^\top I_{\mathcal{B}}). \tag{21}$$

Its diagonal is the individual contribution energy used by Eq. (2). We define

$$E_{\text{diag},\mathcal{B}} = \sum_j q_j^2 Q_{\mathcal{B},jj}, \qquad E_{\text{cross},\mathcal{B}} = E_{\mathcal{B}} - E_{\text{diag},\mathcal{B}}. \tag{22}$$

The interaction term is signed and can reflect reinforcement or cancellation. It is distinct from the mean/within decomposition over tokens in Eq. (7).

Equation (21) and Table 1 show that individual contribution energies omit signed interaction terms. Here we test whether those terms can change the ranking of nearby sets, keeping the number of selected neurons fixed.

Fixed candidate neighborhood. The candidate experiment uses a separate, preselected subset of 32 WT2 reporting windows, dense inputs only, $T = 32$, and all 32 layers. Eight blocks per window are selected by a deterministic hash, one from each of eight strata within the common $[128, 1920)$ region. For each block and sparsity, the baseline is stable top-K by diagonal energy, with neuron-index ordering for ties. The lowest-scoring 64 kept neurons form a removal pool; the highest-scoring 64 discarded neurons form an addition pool. We generate 32 unique one-neuron swaps and 32 unique four-neuron swaps using a fixed hash rule, before evaluating their output errors. All masks have the same keep count. Including the baseline yields 65 masks per observation, 8,192 block–layer observations per sparsity, and 1,064,960 candidate rows.

Observed reversals and their size. A strict reversal increases diagonal error and decreases actual output SSE beyond the numerical tolerances below. Table 9 reports whether at least one such candidate exists in each observation, not the fraction of individual candidates that improve. Best-candidate reduction is $100(E_0 - \min_{m \in \mathcal{C}} E(m))/E_0$, where E_0 is the baseline output SSE and $\mathcal{C}$ contains the 65 fixed masks. We average these blockwise reductions; this differs from the energy-pooled ratios in Table 1. Candidate selection and evaluation use the same block, so this is not held-out performance of a learned selector.

s	Reversal count	Mean reduction (%)	95% CI (%)
0.7	8,192 / 8,192	0.088960	[0.087932, 0.090041]
0.9	8,192 / 8,192	0.121289	[0.119880, 0.122772]

Table 9: Ranking disagreement in the fixed candidate neighborhood. CIs for mean reduction use 2,000 window-bootstrap resamples with seed 260924. Observations and candidates within a window are not independent samples. Complete observed coverage does not establish universal reversal prevalence.

Reversals remain present in every sampled observation after excluding the final layer; mean best-candidate reductions become 0.08154% and 0.11299% at the two sparsities. These small, retrospective gains establish ranking disagreement in the sampled neighborhood, not a practical selector improvement or an explanation of ReFiT's reconstruction gains.

Numerical validation and scope. Diagnostic products use FP32 with TF32 disabled and FP64 reductions. Strict reversals require output-SSE changes greater than $10^{-5}\max(E_0, E, 10^{-12})$ and diagonal changes greater than $10^{-10}\max(D_0, D, 10^{-12})$, where D denotes diagonal error. A fixed hash-selected one-eighth of blocks additionally checks the baseline, a fixed candidate, and the strongest reversal by direct FP64 evaluation. There are 6,143 direct checks, with maximum relative SSE error 2.706×10^{-8}; all 4,095 checked ranking comparisons agree. The remaining candidates use the specified indirect arithmetic, not direct FP64 recomputation. Deterministic FP64-score top-K differs from the FP32 reference in two of the 16,384 observations. Thus this candidate experiment and the 130-window cross-term analysis differ in sampling and baseline construction and are not an improvement-before/after pair. No $T = 1$ control, predictor training, down-projection fitting, or PPL evaluation is part of this experiment; it does not establish a block-sharing-specific repair effect.

D Full Quality Results and Reproducibility

We retain the full evaluated grids, including conditions that worsen. Table 10 completes the RES grid, Table 11 gives the true-activation score reference, and Tables 12–13 cover input sparsity. The BCE grid is complete in Table 4. Each pair uses one calibration seed; counts of improving pairs do not estimate an improvement probability over calibration samples.

Paper-reference implementation checks. The FastForward implementation follows the reported predictor, compensator, and sparsity-scheduling architecture (Gautam et al., 2026). In historical LLaMA3.2-1B-Instruct LongBench checks, dense scores were 31.92 versus the reported 29.88, and the full 50%-sparse configuration scored 29.31 versus 28.37. These establish a paper-reference comparison, not exact numerical reproduction: prompt/evaluation settings differ, and the local compensator gain was not reproduced. The paper does not report a PPL counterpart. Current E1/E2 deliberately fix the predictor and use uniform sparsity without a compensator to isolate down-projection refitting; their scores use the protocols below, not these historical endpoints. For LaRoSA, the paper's 50% schedule gives LLaMA2-7B PPL 5.9592 without rotation and 5.8339 with rotation, versus 6.02 and 5.87 reported (Liu et al., 2025b). Both the quality and the direction of the rotation benefit are reproduced approximately under the common evaluation protocol (166 windows rather than the paper's 128 random samples). The rotation generator was also checked against the upstream implementation. E3 then extends this implementation to FFN-input rotation and exact-K block masks, so its paired results evaluate that specified adaptation.

D.1 Intermediate-neuron selection

Predictor training variants. All predictors use attention pooling followed by an MLP, with weights frozen during ReFiT fitting and evaluation. E2 uses the weighted binary cross-entropy (BCE) objective with dense-activation labels following FastForward. E1 uses the residual-score regression variant (RES): the MLP predicts blockwise log-standardized neuron residual energies relative to fixed rank-1376 projection sketches, using mean-squared loss. The sketches supply training targets only; no sketch compensation or affine correction is active in the evaluated FFN. These

	$s = 0.5$		$s = 0.7$		$s = 0.9$	
T	w/o ReFiT	ReFiT	w/o ReFiT	ReFiT	w/o ReFiT	ReFiT
8	7.8679	7.5008	15.4405	13.1668	321.7133	64.8995
16	7.9614	7.4987	16.1966	13.3223	396.5894	71.7338
32	8.4322	7.8010	18.2481	14.1985	476.9326	80.7892
64	8.6032	7.8550	19.3665	14.3129	588.6753	89.4562
128	8.9127	8.0530	20.8142	14.6255	614.0278	81.3719

Table 10: LLaMA2-7B, RES predictor: full diagnostic PPL grid. Fixed 128-token dense boundaries and DENSE-D0. Single measurements; lower is better.

	$s = 0.5$		$s = 0.7$		$s = 0.9$	
T	w/o ReFiT	ReFiT	w/o ReFiT	ReFiT	w/o ReFiT	ReFiT
8	6.0033	5.9161	6.9557	6.5169	15.3493	8.5826
16	6.1722	6.0306	7.3910	6.7588	20.3244	9.4423
32	6.3202	6.1292	7.8463	6.9634	26.2240	10.7394
64	6.4609	6.2126	8.3856	7.1810	32.6794	12.8012
128	6.6028	6.3009	9.0393	7.5101	39.0099	15.9357

Table 11: LLaMA2-7B, true-activation score policy: full diagnostic PPL grid. Fixed 128-token dense boundaries and DENSE-D0. Single measurements; lower is better.

objectives are not the same as the actual-activation score reference of Eq. (2); we do not compare their relative quality as a separate contribution.

FastForward-family protocol. RES, BCE, and true-activation score evaluations use 2048-token windows with 128-token dense prefix and suffix spans, leaving 1792 sparse tokens per window. LLaMA2-7B, LLaMA3.2-1B, LLaMA3.2-3B, and Qwen3-4B use 166, 141, 141, and 146 test windows, respectively. Calibration uses 128 such sequences (262,144 packed tokens; 229,376 sparse tokens per layer enter the fit), target y^*, $\mu_{\text{rel}} = 1$, and seed 42, without a compensator or affine predictor correction. Statistics and Cholesky solves use FP64; fitted weights are stored in FP16 and loaded into the BF16 model. Original weights are retained on dense spans. We collect all layer statistics on one forward pass of the original sparse model, before installing fitted matrices. Selection uses the original scoring weights or frozen predictor; upstream refits can still change downstream inputs and realized masks. RES $T = 8, 16, 64, 128$ uses 655.36M training tokens; $T = 32$ uses 163.84M. Each before/after pair uses the identical predictor checkpoint. Across-T comparisons therefore also involve different predictor training histories. All models use their own tokenizers and dense references. The test data have historical exposure within this research programme, so this is not a new held-out confirmation study.

Runtime and aggregation. Both arms of every main pair were evaluated on the same GPU family. RES $T = 8, s = 0.9$, all $T = 16$ pairs, and $T = 128, s = 0.9$ used B200 with PyTorch 2.8 and Transformers 4.46.3; the other main pairs used A100 with PyTorch 2.6 and Transformers 4.53.2. Model weights and implementation files were checked across hosts, but absolute cross-family PPL parity was not measured. Each 2048-token window contributes 2047 scored tokens; NLL is summed in FP64 before exponentiation. Window PPLs are never averaged. There is no KV-cache sharing between independent PPL windows.

True-activation score reference. The score policy ranks neurons by $\sum_{t \in \mathcal{B}} i_{t,j}^2 \|W_{0,:,j}^d\|^2$ using the block's actual intermediate activations and original scoring weights. ReFiT improves all 15 pairs in this grid. It follows the same fixed-boundary and down-projection fitting recipe as the RES grid, but is not a deployable pre-FFN predictor. Its energy ranking does not minimize set-output error or optimize the subsequently fitted down-projection; it is therefore not an upper bound on selection quality. At $T = 128, s = 0.9$, diagnostic PPL changes from 39.01 to 15.94. The gap from the RES result is an observed policy comparison, not an additive decomposition into predictor and block-sharing errors.

Inputs	T	$s = 0.5$	$s = 0.7$	$s = 0.9$
UG	8	$8.3745 \to 8.2979$	$32.3167 \to 28.5521$	$1531.63 \to 851.81$
UG	16	$10.1561 \to 9.9987$	$45.3567 \to 38.8884$	$1967.88 \to 1039.79$
UG	32	$11.8963 \to 11.6571$	$55.3587 \to 46.4937$	$2436.48 \to 1298.51$
UGD	8	$8.9761 \to 8.9919$	$38.6390 \to 35.4776$	$2556.80 \to 1722.43$
UGD	16	$11.1253 \to 11.1357$	$60.1925 \to 52.0984$	$3060.89 \to 2762.99$
UGD	32	$13.3742 \to 13.3372$	$83.2730 \to 69.1286$	$3340.61 \to 3193.91$

Table 12: Full representative Experiment 3 grid: w/o ReFiT $\to$ ReFiT-D diagnostic PPL, fixed $\alpha = 0.001$, $\beta = 0.1$, C-spread16, no dense boundary exceptions. UG improves in nine of nine pairs and UGD in seven of nine; the $s = 0.5$, $T = 8, 16$ UGD pairs slightly worsen. All values use one calibration seed. The larger-block UGD conditions appear separately in Table 13.

D.2 Input sparsity and applicability boundaries

Table 12 reports UG and UGD at $T \in \{8, 16, 32\}$ and $s \in \{0.5, 0.7, 0.9\}$. Table 13 adds the $T = 64, 128$ UGD stress conditions. Q is fitted at the post-attention layer-normalization output. UG shares an exact-K_{UG} block-energy mask across the rotated up/gate inputs, where $K_{\mathrm{UG}} = \lfloor (1-s)h \rfloor$; UGD additionally uses an exact-K_{D} block-energy mask on the unrotated intermediate, where $K_{\mathrm{D}} = \lfloor (1-s)d \rfloor$. Here $h = 4096$ and $d = 11008$; ties are resolved in favor of the lower channel index. At $s = 0.7$, UG and UGD retain approximately 53.3% and 30.0% of dense FFN projection FLOPs, respectively, before rotation and selection costs. The same nominal sparsity is therefore not a matched-compute comparison.

Both use C-spread16 calibration (16×2048 tokens), no dense boundary exceptions, FP64 statistics/Cholesky, relative regularization $\mu = \alpha \operatorname{tr}(G)/d$ with $\alpha = 0.001$ for the down-projection, damping $\beta = 0.1$, and a final BF16 cast. Damped deployment uses

$$W^d_{\mathrm{deploy}} = W^d + \beta(\tilde{W}^d - W^d), \tag{23}$$

where $\tilde{W}^d$ is the anchored ridge solution. For $\beta < 1$, the installed matrix is not the full minimizer of Eq. (8). SEP fits up/gate projections; SEP-D recollects their resulting sparse activations before fitting the down-projection. E3 reports PPL only. The unmasked teacher and sparse regressor use the same local input from the parent sparse trajectory. Fits are collected independently for each operating point. The table reports one calibration seed and diagnostic PPL, not a probability of improvement over seeds. The $s = 0.7$ main-text table is an illustrative slice of an already observed grid; all sparsities, including the two small UGD degradations, are retained here.

Calibration sample and rotation. C-spread16 contains one sequence from each of 16 WikiText-2 raw train articles. Articles are delimited by top-level headings; those overlapping zero-indexed rows 0–1499, 1582–1935, or 2555–3115 are excluded because of earlier calibration or diagnostic use. Each article's rows are joined with two newlines and tokenized without special tokens; eligible articles have at least 2047 tokens. Python's `random.Random(0)` shuffles eligible articles in row order. For each of the first 16 articles, the same RNG draws a uniform valid start; the sequence is BOS followed by 2047 consecutive content tokens, with no padding. This frozen sample is shared across operating points. The fixed layerwise rotation uses eigenvectors of the uncentered input Gram matrix collected on this sample from the dense model at the stated normalization output. The rotation generator accumulates in FP64, casts the Gram matrix to FP32 to add $0.01 \operatorname{mean}(\operatorname{diag} G) I$, and computes eigenvectors in FP64 in ascending eigenvalue order. This rotation construction is separate from fitting the sparse down-projection on each parent's trajectory.

Projection-wise refitting ablation. SEP fits up and gate separately on the original sparse parent's inputs and installs both updates. D fits only down on that parent. SEP-D collects new statistics on the SEP parent's trajectory before fitting down; it does not combine independently fitted SEP and D checkpoints. Table 6 uses the same fixed regularization and damping as Experiment 3. Validation-selected $T = 8$ recipes are not mixed into these block-size comparisons.

Larger-block UGD and failure cases. The larger-block UGD runs use the same rotation, exact-K selection, C-spread16 calibration, regularization, damping, and no-boundary-exception recipe

	$s = 0.5$		$s = 0.7$		$s = 0.9$	
T	w/o ReFiT	ReFiT	w/o ReFiT	ReFiT	w/o ReFiT	ReFiT
64	15.4056	15.2297	107.1766	85.6071	3659.8865	3801.1681
128	17.1968	16.8663	128.4922	99.9646	3853.0271	4564.2925

Table 13: LLaMA2-7B, F-Q UGD input sparsification: larger-block diagnostic PPL. The $T = 8, 16, 32$ grid is in Table 12. No dense-boundary exception; C-spread16 calibration and down-only repair. Single measurements; lower is better.

		$s = 0.5$		$s = 0.7$		$s = 0.9$	
Task	Dense	w/o ReFiT	ReFiT	w/o ReFiT	ReFiT	w/o ReFiT	ReFiT
narrativeqa	17.74	16.26	15.89	12.62	13.90	2.98	12.97
qasper	9.91	8.85	9.07	6.42	7.81	2.60	5.49
multifieldqa_en	24.34	25.18	22.86	21.58	19.25	11.66	13.37
hotpotqa	7.95	7.38	7.96	6.74	7.78	5.19	7.87
2wikimqa	10.92	10.29	10.76	8.68	10.42	8.43	9.95
musique	4.43	3.42	3.66	2.53	2.93	2.08	3.22
gov_report	26.81	24.41	25.84	20.93	23.62	9.44	14.57
qmsum	20.59	19.68	20.35	18.40	19.12	14.17	18.20
multi_news	2.93	3.04	3.00	5.06	4.50	1.59	6.23
trec	67.50	63.50	64.00	46.00	52.00	31.50	33.00
triviaqa	88.42	86.28	87.18	79.18	84.72	56.70	66.61
samsum	41.02	42.47	40.80	33.72	38.29	6.71	23.05
passage_count	1.58	1.00	0.17	0.43	0.79	1.48	1.42
passage_retrieval_en	5.64	7.46	7.54	5.83	5.00	3.75	4.00
lcc	68.24	66.36	66.79	62.51	64.36	37.15	58.70
repobench-p	61.93	60.19	60.73	55.66	58.30	39.24	51.93

Table 14: Per-task LongBench scores for LLaMA2-7B, RES $T = 32$. Each task uses its recorded benchmark metric on a 0–100 scale (higher is better). All 16 tasks, including synthetic tasks with low dense scores, are retained in the macro average in Table 3.

as the UGD rows above. Four of the 15 UGD pairs worsen. At $s = 0.5$, $T = 8, 16$ worsen by approximately 0.0018 and 0.0009 nats/token. At $s = 0.9$, $T = 64, 128$ worsen by 0.0379 and 0.1694. These observations do not by themselves distinguish upstream information loss, input-distribution shift, or calibration effects. No matched UGD LongBench grid is reported here.

D.3 Generation quality and sampling sensitivity

LongBench protocol. All suites use the same 16 English tasks, 3,750 prompts, task-specific generation caps, and greedy decoding. Middle truncation caps prompts at 3,500 tokens without a chat template for LLaMA2-7B, and at 31,500 tokens with model-specific chat templates for Experiment 2 (Qwen3 thinking disabled). Sparse prefill keeps first/last 128-token dense spans and uses DENSE-D0; decoding uses original dense weights. Dense controls share each model's prompt and context settings. Scores use each task's benchmark metric, including F1, ROUGE-L, classification/retrieval accuracy, and code edit similarity. The reported average weights the 16 tasks equally, rather than weighting all examples equally. There are nine RES pairs ($T = 8, 32, 128$ crossed with $s = 0.5, 0.7, 0.9$), nine BCE pairs, and four dense controls. Archived shards ran on A100, A6000, and B200 GPUs; archived Qwen3 runs used A100 only. Table 14 retains both improvements and regressions for the representative RES $T = 32$ grid. Task-level records for all 40 suites are archived in the campaign artifacts. The nine E1 pairs agree with the author-confirmed grid to two decimal places; analyses use the archived full-precision means. E3 is evaluated by PPL only.

Block-size sensitivity on both quality scales. For each sparsity and arm, we measure the max–min range over the evaluated block sizes. In the E1 PPL grid ($T = 8, 16, 32, 64, 128$), the mean-NLL ranges before/after ReFiT are 0.125/0.071 at $s = 0.5$, 0.299/0.105 at $s = 0.7$, and 0.646/0.321 at $s = 0.9$, in nats/token. Here mean NLL is $\log(\mathrm{PPL})$, so these comparisons show that the reduced

s	Mean gain (points)	95% interval (points)	Tasks improved
0.5	$+0.051$	$[-0.378, +0.471]$	11/16
0.7	$+1.656$	$[+1.053, +2.270]$	13/16
0.9	$+5.993$	$[+5.395, +6.590]$	15/16

Table 15: Paired LongBench sampling sensitivity, LLaMA2-7B RES, $T = 32$. Gain is ReFiT minus unrepaired quality on the 0–100 task-score scale. Intervals use within-task resampling with equal task weights; they are not calibration-seed confidence intervals.

variation is not solely a consequence of PPL's exponential scale. In LongBench ($T = 8, 32, 128$), the corresponding score ranges shrink from 1.17 to 0.79 points at $s = 0.7$ and from 2.94 to 0.62 at $s = 0.9$. At $s = 0.5$, however, the LongBench range increases from 0.102 to 0.312 points despite improvements at every operating point. Thus reduced generation-score variation is observed at the two higher sparsities, not across the entire grid. Predictor training histories also differ across T (Appendix D.1). The block-size coverage differs between metrics; these single-run ranges are descriptive, not uncertainty estimates.

Paired sampling sensitivity. For the RES $T = 32$ grid, we resample paired prompt scores within each of the 16 tasks and recompute their equally weighted mean (10,000 bootstrap replicates). Table 15 distinguishes the small observed $s = 0.5$ gain from the larger $s = 0.7, 0.9$ gains. Each pair contains 3,750 prompts. Recorded row order is checked against reference answers, original length, and context-token count; explicit example IDs were not stored. All task means reproduce the source scores used by the manuscript. These post-hoc percentile intervals describe prompt-sampling sensitivity conditional on the fitted checkpoints and fixed task set. They do not include calibration-seed variability, new-task generalization, or correlations among prompts that share a source document.

D.4 Scope of the retained evidence

The quality grids above use the stated matched protocols. Earlier pre-anchor sweeps, mask-mismatch tests, compensation stacks, and analytic Pareto comparisons used other solver, mask, or boundary conventions and are excluded from the current numerical evidence. The oracle local-reconstruction analysis in Appendix B retains its separate protocol; it is not pooled with the current predictor-based quality results. No measured latency frontier or matched-budget compensation advantage is established by these quality grids.