ATOMSHARE: ONE SHARED ATOM SET ACROSS A MIXTURE-OF-EXPERTS STACK FOR BUDGET-CONSTRAINED WEIGHT COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

A sparse mixture-of-experts stack stores one weight matrix per expert at every depth, and its stored size therefore grows with the expert count while only a few experts run per token. However, existing compressors fit each matrix in a basis of its own and price its error by how far that matrix moves; they therefore cannot reach the redundancy that recurs across the experts of a layer and they spend the budget against a distance the router never applies. We propose AtomShare, which stores a whole stack as one atom set shared across depths, carried into each layer by a norm-preserving map, with every expert held as a sparse code against that dictionary. AtomShare achieves this by fitting one atom set against every expert of the stack in each layer's recorded input geometry, by setting each unit's code length from the deviation it delivers at the layer output on the tokens the router sends it, and by dividing one stored-size budget across the layers by the deviation each layer's injection propagates to the end of the stack. Sharing the dictionary is what makes code length a single divisible pool, and the atoms, maps and codes are therefore fit together as one alternation rather than matrix by matrix. On an eight-layer, forty-eight-expert weight stack AtomShare reaches a retained score of 0.5483 against 0.5150 for the strongest per-expert baseline while holding fewer bits per parameter, and its compression ratio rises with the expert count, from 6.149 at eight experts to 8.385 at sixteen, where every per-expert store but one holds its own ratio fixed across that range. A mechanism ablation read at the routed output puts the recorded input geometry first: removing it raises the delivered deviation by 67 to 93 per cent.

1 INTRODUCTION

A sparse mixture-of-experts (MoE) layer holds one weight matrix per expert and routes each token to a few of them; its memory therefore grows with the expert count while its arithmetic does not (Shen et al., 2023; Cai et al., 2024). Released MoE language models already place eight or more experts in every MoE layer (Jiang et al., 2024; Dai et al., 2024). Adding experts is how these models buy capacity, and it is also what makes them expensive to hold on a device that runs only a handful of them per token (Lu et al., 2024; Wang et al., 2025a). The deployment cost is therefore the stored size of the whole stack. One budget is fixed for that stack, and every expert placed inside it competes for the same bytes.

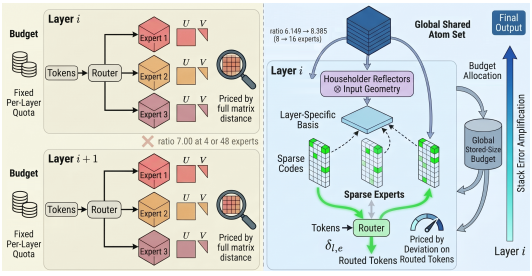

Figure 1: **Left:** a fixed per-layer quota prices each expert by its full-matrix distance; the ratio stays 7.00 at 4 or 48 experts. **Right:** one shared atom set prices each expert by its routed-token deviation; the ratio rises from 6.149 to 8.385.

Weight compression for large models is built around the single matrix: low-rank truncation and factorization fit each one with a few directions and recover most of its energy at a fraction of its size (Ren & Zhu, 2024; Saxena et al., 2025). One-shot sparsification keeps the entries that carry the reconstruction and discards the rest, with saliency read off a calibration batch such that the retained pattern reflects the inputs the layer actually sees (Li et al., 2023; Meng et al., 2024). Quantization

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.

drives the same matrix to a few bits and restores what the grid loses through a low-rank or sparse residual, which buys back the outlier directions that plain rounding destroys (Dettmers et al., 2024; Jeong et al., 2024). Work specific to MoE models acts at the level of the expert instead, pruning or skipping the ones a workload rarely selects and rescheduling the remainder for serving (Lu et al., 2024). Codebook methods go further and let one dictionary serve many weights, paying once for a direction that recurs across matrices (van der Ouderaa et al., 2026). Together these lines have made a single expert cheap to store and made the choice of which experts to keep an explicit part of the design. Despite these advances, the price an expert pays is still set inside its own matrix.

However, each of these stores is fit one matrix at a time, and its compression ratio is therefore a property of the matrix rather than of the stack: on the weight stacks we study, per-expert low-rank holds a ratio of 7.00 whether a layer carries four experts or forty-eight. The redundancy an MoE stack creates is the redundancy a per-matrix store cannot reach, because it lives across the experts of a layer and grows with exactly the quantity that makes the stack expensive. A second reading is that the budget is spent against the wrong price: an expert acts only on the tokens the router sends it, and across forty-eight readings their order by matrix error and by delivered layer-output deviation agree at a rank correlation of 0.149. What an expert is worth storing and what its error costs are set by the same object, the layer's routed input, and a per-matrix store reads neither: it fits each expert in a basis no other expert may use, and prices that fit by a distance the router never applies.

To address these challenges, we propose AtomShare, a weight store fit across an entire MoE stack under a single stored-size budget. The key insight is that the structure worth sharing and the price worth paying are read at the same place, the routed layer output: one atom set fit in each layer's recorded input geometry can serve every expert in the stack, and each unit's code length can be bought with the deviation it delivers on the tokens the router sends it. AtomShare has three parts: one atom set shared across depths, carried into each layer by a product of Householder reflectors and the layer's recorded input geometry; a per-unit sparse code whose length is priced by the deviation that unit delivers on its routed tokens; and a byte allocation that spends the single budget where a layer's injected error is most amplified to the end of the stack. The atoms, maps and codes are fit together as one alternation rather than matrix by matrix, and the atoms move against every expert of the stack at once. Our contributions are as follows:

- We identify three gaps that per-matrix stores leave in an MoE stack: cross-expert redundancy, code length priced by weight error, and per-layer quotas across coupled layers.
- We propose AtomShare, storing a stack as one atom set shared across depths, pricing each code by routed-token deviation, and dividing one budget by measured amplification.
- AtomShare reaches a retained score of 0.5483 against 0.5150 for the strongest per-expert baseline at fewer bits per parameter, and its compression ratio rises from 6.149 at eight experts to 8.385 at sixteen, while every per-expert store but one stays flat.

2 MOTIVATION

When compressing an MoE weight stack under a single stored-size budget, we ran a series of measurements over expert counts, layer depths and perturbation sizes, and identified three challenges.

Challenge #1: cross-expert redundancy. Every per-expert compressor fits an expert in a basis of its own, which assumes that the directions an expert needs are its own. To investigate, we compare the energy a set of directions fitted jointly across the experts of a layer captures against the energy a per-matrix truncation captures at equal stored size, over eight depths and eight expert counts. As shown in the left panel of Fig. 2, the ratio runs 0.730 to 0.849 at four experts, where the shared dictionary does not pay for itself, straddles one at sixteen, and reaches 1.302 to 1.363 at every depth by forty-eight. The redundancy is therefore not a property of the expert but of how many experts share the fit, and a per-matrix store pays for it once per matrix by construction, which is why the per-expert low-rank ratio holds at 7.00 from four experts to forty-eight. This raises a key question: *what must an expert store beyond its code once one atom set serves the whole stack?*

Challenge #2: mispriced code length. A budget spent to minimize each expert's matrix distance treats that distance as the harm. An expert acts only on the tokens the router sends it; the stack therefore pays for the deviation delivered at the layer output. To investigate, we perturb every expert of a layer and compare each expert's Frobenius reconstruction error against the deviation it induces

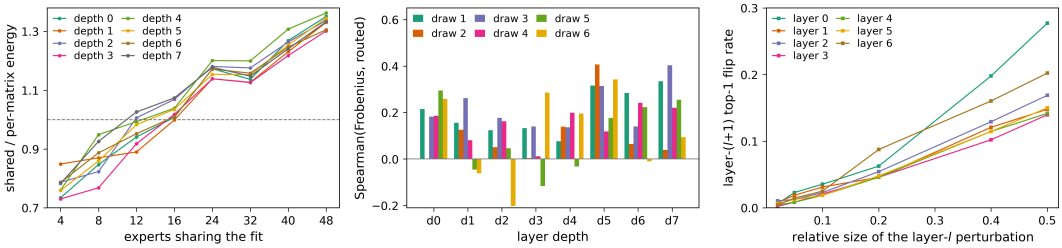

Figure 2: **(Left)** Shared against per-matrix energy, one line per depth. **(Middle)** Matrix error against routed deviation, six draws. **(Right)** Routing flips below a perturbed layer. Details in Appendix A.2.

at the layer output on its routed tokens, at eight depths and over six independent isotropic Gaussian draws. As shown in the middle panel of Fig. 2, the rank correlation between the two stays far below one everywhere: across the forty-eight readings the mean is 0.149, none exceeds 0.407, six fall below zero and the lowest is -0.202. A matrix norm weighs the directions the layer never receives equally with the ones it does. This raises a key question: *what changes when each unit's code is priced by the deviation it delivers rather than by how far its matrix moves?*

Challenge #3: cross-layer coupling. Splitting a budget into per-matrix quotas treats the layers as independent. A compressed layer's output is the next layer's router input, and error injected at one depth can change which experts run below it. To investigate, we perturb only the layer-l weights, rerun the stack from that layer, and measure the fraction of tokens whose top-1 expert at layer $l{+}1$ changes, at seven depths and six perturbation sizes. As shown in the right panel of Fig. 2, the flip rate runs 0.008 to 0.023 at a relative size of 0.05 and 0.102 to 0.198 at 0.40, rising with the perturbation at every depth. A per-layer quota therefore prices a layer as if its error stopped there. This raises a key question: *how should one stored-size budget be divided across layers that feed one another?*

Sharing one atom set turns a per-expert basis into a single pool of code length, and the only free quantity is how many atoms each unit spends: the second challenge prices that pool and the third divides it across coupled layers, which places all three in one budget for the whole stack.

3 Problem Formulation

To address the three challenges of Section 2 together, we formulate MoE weight-stack compression as a budget-constrained minimization over stored representations:

$$\min_{\Theta \in \mathcal{A}} \Gamma(\Theta) \quad \text{s.t.} \quad \mathcal{M}(\Theta) \leq B \tag{1}$$

where Θ is the stored representation of the whole stack, $\mathcal{A}$ the class of admissible representations, $\Gamma(\Theta)$ the deviation between the compressed and uncompressed run measured at the end of the stack on a fixed reference set, $\mathcal{M}(\Theta)$ the stored size in words, and B the deployment's stored-size budget.

Challenge #1 enters through $\mathcal{A}$, which must hold representations whose shared cost does not grow once per expert. Challenge #2 enters through Γ, which is read at the output the stack delivers and not as a distance in weight space. Challenge #3 enters through the single scalar B, which binds the stack as a whole and not each matrix on its own.

4 Method

4.1 Overall Framework

To solve Eq. (1), we propose AtomShare, which stores a whole sparse MoE weight stack (Shen et al., 2023) as one atom set shared across depths. Fig. 3 shows its three components: shared atoms across depths make a direction that recurs across the experts of a layer cost one entry for the stack; per-unit code length priced by routed-token deviation sets each unit's code from what it delivers at the layer output; and budget allocation across the coupled stack divides one stored-size budget where a layer's injected error is most amplified.

The three form one system. Sharing the atom set is what turns code length into a single divisible pool: the atom, map and geometry terms are paid independently of the expert count, and the only free quantity left is how many atoms each unit spends, which the second component prices and the

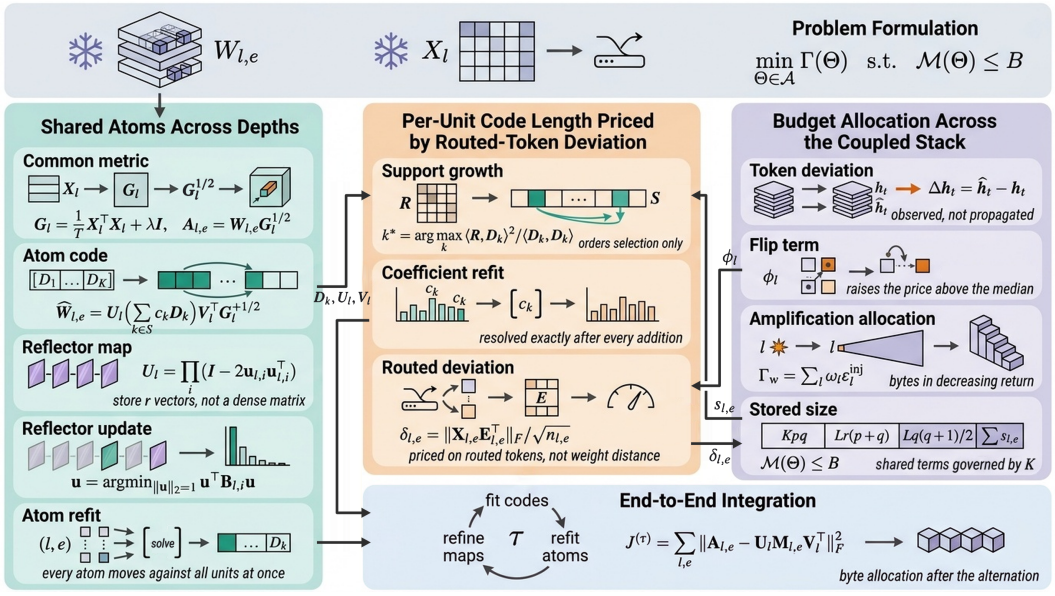

Figure 3: **Left:** shared atoms across depths, carried into each layer by Householder reflectors. **Middle:** each unit's code grows atom by atom, priced by routed-token deviation. **Right:** one stored-size budget divided by measured amplification. **Bottom:** codes, atoms and maps fit in one alternation.

third divides. The representation is therefore fit as one alternation rather than matrix by matrix, and Eq. (1) minimizes over

$$\Theta = \Big(\{D_k\}_{k=1}^{K},\ \{U_l, V_l\}_{l=1}^{L},\ \{G_l^{1/2}\}_{l=1}^{L},\ \{(S_{l,e}, c_{l,e,\cdot})\}_{l,e} \Big) \tag{2}$$

where $\{D_k\}$ is the shared atom set, U_l, V_l the per-layer maps, $G_l^{1/2}$ the recorded layer geometry, stored and charged for in Eq. (14), and $(S_{l,e}, c_{l,e,\cdot})$ the code of unit (l, e).

4.2 Shared Atoms Across Depths

To address Challenge #1, this component builds the class $\mathcal{A}$: one atom set shared across depths, carried into each layer by a norm-preserving map. Fig. 3 (left) traces its five steps, from recording each layer's input geometry to refitting every shared atom jointly against all units at once.

Common metric. Each layer's input geometry is recorded once and its experts placed in the metric it induces:

$$G_l = \tfrac{1}{T} X_l^\top X_l + \lambda I, \qquad A_{l,e} = W_{l,e} G_l^{1/2} \tag{3}$$

where X_l stacks the T rows entering layer l, λ ridges the second moment to keep the metric positive definite, and $A_{l,e}$ is expert (l, e) in that metric, which we compute once per layer and reuse for its experts, such that a fit minimizes the error the layer output sees.

Atom code. Each expert is a sparse combination of the shared atoms carried into its layer:

$$\widehat{W}_{l,e} = U_l \Big(\sum_{k \in S_{l,e}} c_{l,e,k} D_k \Big) V_l^\top G_l^{+1/2} \tag{4}$$

where $S_{l,e}$ is the support and $G_l^{+1/2}$ the inverse square root of G_l, taken on floored eigenvalues, and we apply U_l and V_l to the atom combination and return it to weight coordinates.

Reflector map. Each per-layer map is a product of r Householder reflectors, orthogonal by construction and costing only those vectors:

$$U_l = \prod_{i=1}^{r} H_{l,i}, \qquad H_{l,i} = I - 2\, u_{l,i} u_{l,i}^\top \tag{5}$$

where $H_{l,i}$ is the i-th reflector and $u_{l,i}$ its unit vector, which we set and store as those r vectors, not a dense matrix (V_l likewise).

Reflector update. One reflector is refined at a time with the others fixed, as the smallest-eigenvalue eigenvector of a form built from the layer's alignment target:

$$u_{l,i} = \arg\min_{\|u\|_2=1} u^\top B_{l,i} u, \qquad B_{l,i} = \mathrm{sym}\big(P^-_{l,i} \Sigma_l^\top P^+_{l,i}\big) \tag{6}$$

where $\Sigma_l = \sum_e A_{l,e} V_l M_{l,e}^\top$ is the layer's alignment target with $M_{l,e}$ the atom combination of Eq. (15), $P^+_{l,i}$ and $P^-_{l,i}$ are the products of the reflectors before and after i, and $\mathrm{sym}(\cdot)$ the symmetric part; the r reflectors are set in turn and then refined over a fixed number of passes.

Atom refit. The atoms are refit against every unit at once, solving the system that zeroes the atom-coordinate gradient:

$$\{D_k\}_{k=1}^K = \arg\min_{\{D_k\}} \sum_{l,e} \Big\| A_{l,e} - U_l \big(\textstyle\sum_{k \in S_{l,e}} c_{l,e,k} D_k\big) V_l^\top \Big\|_F^2 \tag{7}$$

where the sum runs over every unit (l, e), and we solve this system in closed form, and every atom thus moves against all units at once, coupling the atoms across depths. One atom set is handed on with its maps.

4.3 Per-Unit Code Length Priced by Routed-Token Deviation

To address Challenge #2, this component sets how many atoms each unit's code holds, from the deviation it delivers at the layer output on its routed tokens. Fig. 3 (middle) traces its three steps: support growth, coefficient refit after every addition, and routed deviation as the price of the code.

Support growth. The support grows one atom at a time, taking the atom whose single coefficient most reduces the residual:

$$k^\star = \arg\max_{k \notin S_{l,e}} g_{l,e}(k), \qquad g_{l,e}(k) = \frac{\langle R_{l,e}, D_k \rangle^2}{\langle D_k, D_k \rangle} \tag{8}$$

where $R_{l,e}$ is the unit's current residual, $g_{l,e}(k)$ the reduction one coefficient on atom k would give, a computed lower bound that orders selection only; we select $k^\star$ by it, never by the deviation the unit costs.

Coefficient refit. The coefficients on the whole support are resolved exactly after each atom is added:

$$c_{l,e,\cdot} = \arg\min_c \Big\| A_{l,e} - U_l \big(\textstyle\sum_{k \in S_{l,e}} c_k D_k\big) V_l^\top \Big\|_F^2 \tag{9}$$

where the minimization runs over the coefficients of $S_{l,e}$, and we resolve them after every addition, which keeps the code the least-squares fit on its support.

Routed deviation. A unit is priced by a linear proxy for the deviation it induces at the layer output on its routed tokens:

$$\delta_{l,e} = \tfrac{1}{\sqrt{n_{l,e}}} \big\| X_{l,e} E_{l,e}^\top \big\|_F, \qquad E_{l,e} = W_{l,e} - \widehat{W}_{l,e} \tag{10}$$

where $X_{l,e}$ is the layer-l input routed to expert e, $n_{l,e}$ its row count, and $E_{l,e}$ the replacement error in the same weight coordinates the forward pass applies. It is a linear proxy: the executed layer applies a pointwise nonlinearity and the layer-to-layer transform after this product, and $\delta_{l,e}$ therefore is not the layer output difference itself. A unit routed fewer rows than a fixed floor is priced on that many rows nearest its centre instead, and an expert with no traffic thus still carries a price. Each unit's code is priced by $\delta_{l,e}$, and the budget is thus spent against the routed proxy, not weight distance.

4.4 Budget Allocation Across the Coupled Stack

To address Challenge #3, this component divides one stored-size budget across layers by the deviation each layer's injection propagates to the stack end. Fig. 3 (right) traces its four steps: token deviation from one pass, flip term, amplification allocation and the single stored-size account.

Token deviation. The per-token deviation is read off a pass over the reference set, not inferred from per-unit errors:

$$\Delta h_t = \widehat{h}_t - h_t \tag{11}$$

where h_t and $\widehat{h}_t$ are the uncompressed and compressed hidden states of token t, which we compute from the two runs; the reading is thus observed rather than propagated.

Flip term. A layer whose perturbation reroutes tokens downstream is charged for it, at the rate its own routing turns over:

$$\phi_l = \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}\big[a_{l+1}(\tilde{h}_t) \neq a_{l+1}(h_t)\big] \tag{12}$$

where a_{l+1} is the layer-$(l+1)$ top-1 assignment and $\tilde{h}_t$ the state that reaches it when layer l is perturbed at a stated relative size; ϕ_l is thus a measured turnover rate of the stack. It raises the price of the units whose read sits above their layer's median by that rate, once per stack.

Amplification allocation. We measure how far each layer's injection is amplified to the stack end and take bytes in return order within the budget:

$$\widetilde{\Gamma} = \sum_l \omega_l \varepsilon_l^{\mathrm{inj}}, \qquad \omega_l = \frac{\|\Delta h_{\mathrm{end}}^{(l)}\|}{\varepsilon_l^{\mathrm{inj}}}, \qquad \mathcal{M}(\Theta) \le B \tag{13}$$

We then take bytes in decreasing return up to the atom count that B admits at a uniform per-unit length, where $\Delta h_{\mathrm{end}}^{(l)}$ is the end deviation from compressing only layer l, $\varepsilon_l^{\mathrm{inj}}$ the size of the error injected at layer l and ω_l its amplification; the last layer has no layer after it to displace, and its amplification is therefore taken as 1. The greedy order is a heuristic, run once after the alternation.

Stored size. The whole representation is accounted against one budget, not a per-matrix quota:

$$\mathcal{M}(\Theta) = Kpq + Lr(p+q) + L\frac{q(q+1)}{2} + \sum_{l,e} s_{l,e}\big(1 + \lceil \log_2 K \rceil / w\big) \tag{14}$$

where $s_{l,e}$ is the code length of unit (l,e) and w the word width; the atom, map, and geometry terms are shared over a layer's E experts, which keeps the shared terms independent of the expert count. Each $G_l^{1/2}$ is symmetric; a layer's geometry term is thus its triangle, $q(q+1)/2$ and not q^2. The component returns a byte allocation read off the measured ω_l and ϕ_l, held to $\mathcal{M}(\Theta) \le B$.

4.5 End-to-End Integration

End-to-end integration. The method runs as one alternation over atoms, maps, and codes, shown in the bottom panel of Fig. 3; Algorithm 1 states it end to end, including the per-unit fit. Each sweep fits the codes, then refits the shared atoms, then refines the per-layer maps against the current residual; the three parts are thus fit together and the cost of each step is charged to the fit rather than to the budget. The residual at sweep τ is

$$J^{(\tau)} = \sum_{l,e} \big\| A_{l,e} - U_l M_{l,e} V_l^{\top} \big\|_F^2, \qquad M_{l,e} = \sum_{k \in S_{l,e}} c_{l,e,k} D_k \tag{15}$$

where $M_{l,e}$ is the atom combination of unit (l,e).

Framework summary. AtomShare stores one atom set for the whole stack, prices each unit's code by its routed-token deviation, and spends the single budget where a layer's injection is amplified; the store thus answers to the delivered output, not weight error. The byte allocation is settled after the alternation, once the atoms, the maps and every unit's price curve are fixed.

5 Experiments

5.1 Implementation Details

Weight stacks. The primary stack has eight layers of forty-eight 14×14 experts read on 480 reference tokens, 75264 weights in all, whose experts share 0.548 of their energy. Every stack is

Table 1: Main benchmark on the eight-layer, forty-eight-expert stack at three budget settings. Cells give retained score (↑) / compression ratio (↑), the last column the mean retained score.

System	$b = 0.35$	$b = 0.55$	$b = 0.80$	Avg retained
Uncompressed	1.0000 / 1.0	1.0000 / 1.0	1.0000 / 1.0	1.0000
Per-expert low-rank	0.4708 / 7.0	0.4708 / 7.0	0.4779 / 6.4	0.4732
Mixed-precision low-rank residual	0.5054 / 7.0	0.5054 / 7.0	0.5150 / 6.4	0.5086
Low-rank plus sparse adapter	0.3444 / 6.6	0.3444 / 6.6	0.2810 / 6.3	0.3233
Magnitude sparsification	0.0210 / 8.2	0.0207 / 7.3	0.0207 / 6.5	0.0208
Saliency-weighted sparsification	0.0311 / 8.2	0.0308 / 7.3	0.0294 / 6.5	0.0304
Structured 2:4 sparsification	0.0069 / 8.0	0.0061 / 7.1	0.0057 / 6.2	0.0062
Per-matrix quantisation	0.0011 / 8.0	0.0122 / 7.2	0.0666 / 6.4	0.0266
Per-expert vector quantisation	0.4484 / 5.1	0.4514 / 5.1	0.4558 / 5.1	0.4519
AtomShare (ours)	**0.5135 / 8.3**	**0.5235 / 7.5**	**0.5483** / 6.5	**0.5284**

Table 2: Settings chosen on the primary stack applied unchanged to two further stacks. Retained score (↑) at $b = 0.45$, with the compression ratio (↑) on the fewer-experts column.

System	Primary retained	Fewer experts retained	Fewer experts CR	Greater depth retained
Uncompressed	1.0000	1.0000	1.00	1.0000
Per-expert low-rank	0.4708	0.4697	7.00	0.4702
Mixed-precision low-rank residual	0.5054	0.5047	7.00	0.5064
Low-rank plus sparse adapter	0.3444	0.3834	6.64	0.3574
Magnitude sparsification	0.0210	0.0664	7.69	0.0242
Saliency-weighted sparsification	0.0311	0.0767	7.69	0.0332
Structured 2:4 sparsification	0.0069	0.0302	8.00	0.0080
Per-matrix quantisation	0.0049	0.0101	7.60	0.0025
Per-expert vector quantisation	0.4514	0.4485	5.03	0.4512
AtomShare (ours)	**0.5158**	0.4890	7.66	**0.5220**

drawn at its own fixed seed. Two further settings hold for generalization, thirty-two experts at the same depth and ten layers at forty-eight. Routing is nearest-centre. Details in Appendix A.4.

Baselines. We compare against eight per-expert compressors, per-expert low-rank, mixed-precision low-rank residual, a low-rank-plus-sparse adapter, magnitude and saliency-weighted sparsification, structured 2:4 sparsity, per-matrix quantization and per-expert vector quantization, and against the uncompressed reference, each fitted to the same stored-size target. Details in Appendix A.5.

Budgets. A budget setting b names the stored-size target $B = LEpq(0.10 + 0.07b)$ words and is thus a position on that scale, not a retained fraction. The methods are not compared at an equal stored size; since a budget is met by an integer count of entries, adjacent settings can land on one stored size and return one reading twice. The full grid holds eight settings. Details in Appendix A.6.

Evaluation metrics. We report the retained score (↑), the end-of-stack deviation Γ (↓), the routed-token relative deviation (↓), the compression ratio (↑) and the effective bits per parameter (↓). Γ is the objective of Eq. (1). The retained score is read in the geometry that one component records; the variants are therefore separated on the routed-token deviation instead. Details in Appendix A.7.

Hyperparameters. We use up to twenty-eight shared atoms and three Householder reflectors per layer, a metric ridge $\lambda = 0.05$, eigenvalues of G_l floored at 10^{-9}, $\tau_{\max} = 6$ sweeps with four refinement passes per map, sixteen-bit words and a rank-three probe store for ω_l. Details in Appendix A.8.

5.2 Main Results

Budget benchmark. Table 1 reports the retained score and the compression ratio at three stored-size budget settings. AtomShare reaches 0.5135 at the tightest setting and 0.5483 at the loosest against the strongest per-expert baseline at 0.5054 and 0.5150, leading at all three by 0.0081, 0.0181 and 0.0333, and holding fewer bits per parameter at each of them, 1.917, 2.131 and 2.452 against 2.286, 2.286 and 2.494. The two families land on different stored sizes, AtomShare within budget. The lead follows the store: one atom set carries the directions that recur across a layer's experts, leaving each unit to spend its code only on what it does not share. The lead widens with the budget because code length is one divisible pool: the mixed-precision residual holds 0.5054 at both tighter settings, whereas AtomShare turns each looser one into longer codes. Details in Appendix A.9.

Stack transfer. Table 2 evaluates the settings chosen on the primary stack applied unchanged to a second stack setting and a larger one. AtomShare reads 0.5158 on the primary setting, 0.4890 on the second and 0.5220 on the larger one, each read against its own stack's residual; the three levels

Table 3: Mechanism ablation on the primary stack, one component removed at a time. Retained score (Ret., ↑), end-of-stack deviation Γ (↓) and routed-token deviation (Rout., ↓) at budget b.

	$b = 0.45$			$b = 0.65$			$b = 0.85$		
Variant	Ret.	Γ	Rout.	Ret.	Γ	Rout.	Ret.	Γ	Rout.
Full method	0.5158	1.0786	0.8282	0.5335	1.1140	0.8099	0.5483	1.0755	0.8022
w/o recorded geometry	0.1862	1.0475	1.3796	0.1502	1.0801	1.4808	0.1265	1.0672	1.5488
w/o shared atoms across depth	0.4859	1.0425	0.8918	0.4859	1.0425	0.8918	0.5048	1.0595	0.8669
w/o routed-deviation pricing	0.5154	1.0638	0.8607	0.5334	1.0718	0.8362	0.5487	1.0675	0.8178
w/o stack-coupled allocation	0.5196	1.0783	0.8209	0.5316	1.0798	0.8118	0.5461	1.0901	0.8011
uniform code length	0.5328	1.0611	0.8375	0.5508	1.1080	0.8140	0.5638	1.1189	0.7963

are thus read within their own column. AtomShare leads by 0.0104 on the primary setting and by 0.0156 on the larger one, and on the second it reads 0.4890 against 0.5047 for the mixed-precision low-rank residual while holding the higher compression ratio, 7.66 against 7.00. The settings carry to a deeper stack and cost the method its lead on a narrower one. The narrower stack shares the same 0.548 of its energy; its budget shrinks with the expert count, while the atom and map terms do not shrink in step, leaving fewer words to the codes. Details in Appendix A.10.

5.3 Ablation Study and Analysis

Component removal. Table 3 isolates each component by removing one at a time at three budget settings. The variants separate on the routed-token relative deviation, which is read on the tokens the router sends each expert and uses no recorded geometry; Γ sits between 1.04 and 1.12 for every variant and no ordering is taken from it. On the routed reading the full method is lowest at $b = 0.65$ against the four removals and at $b = 0.45$ and $b = 0.85$ against three of them. Removing the recorded geometry costs the most, 0.8282 to 1.3796 at $b = 0.45$, 0.8099 to 1.4808 at $b = 0.65$ and 0.8022 to 1.5488 at $b = 0.85$, a rise of 67 to 93 per cent in the deviation the store delivers at the routed output; removing the shared atom set costs next, 0.8918 against 0.8099 at $b = 0.65$; the routed-deviation pricing costs 0.0156 to 0.0325 while leaving the retained score within 0.0004 of the full method; and the stack-coupled allocation reads 0.8209 against 0.8282 at $b = 0.45$ and 0.8011 against 0.8022 at $b = 0.85$. The ranking follows the challenges: the shared store, answering cross-expert redundancy, costs most, the pricing, answering mispriced code length, next, and the coupled allocation least, within 0.0073 either way. Details in Appendix A.11.

Materialisation cost. Table 4 quantifies the storage-side cost at the same stored-size budget. AtomShare needs the most multiply-accumulates to rebuild one expert, 9212, and holds the second-fewest peak resident words, 9716 against 9604 for structured sparsity. The shared store buys resident memory and pays for it in arithmetic when an expert is re-expanded through the atoms and the layer map: expanding the sparse code against the atoms costs $O(s_{l,e}pq)$ multiply-accumulates, and U_l, $V_l^\top$ and $G_l^{+1/2}$ are applied as three dense products, which gives the 9212 at the configuration read. A sparse or vector-quantised store restores an expert by scatter or lookup and spends none. Details in Appendix A.14.

Table 4: Storage cost: MACs per rebuilt expert (↓) and peak resident words (↓).

System	MACs	Resident
Uncompressed	0	75460
PE low-rank	196	10948
MP low-rank residual	2940	10948
Low-rank + sparse	196	11524
Magnitude / saliency	0	9988
Structured 2:4	0	9604
PM quantisation	196	10093
PE vector quant.	0	14980
AtomShare (ours)	9212	9716

Uniform code length. Table 6 examines the secondary variants. The uniform-code-length variant reads above the full method on the retained score at all five distinct operating points the two ablation tables cover, by 0.0091 to 0.0194, and below it on the routed-token deviation at two of them, 0.8232 against 0.8237 at $b = 0.55$ and 0.7963 against 0.8022 at $b = 0.80$. Pricing moves code length to the units whose routed tokens deviate most, trading retained score for routed deviation; the trade pays at $b = 0.35$, 0.8314 against 0.8520, and not at the loosest setting. Twelve atoms cap every code at six, half the atom count, and that variant returns 0.5016 at all three budgets. Details in Appendix A.12.

Table 6: Secondary variants on the primary stack at three budget settings b. Retained score (Ret., ↑), end-of-stack deviation Γ (↓) and routed-token deviation (Rout., ↓).

	$b = 0.35$			$b = 0.55$			$b = 0.80$		
Variant	Ret.	Γ	Rout.	Ret.	Γ	Rout.	Ret.	Γ	Rout.
Full method	0.5135	1.0649	0.8314	0.5235	1.0844	0.8237	0.5483	1.0755	0.8022
twelve shared atoms	0.5016	1.0730	0.8716	0.5016	1.0730	0.8716	0.5016	1.0730	0.8716
single pass	0.4862	1.0567	0.8658	0.5022	1.0611	0.8542	0.5268	1.0559	0.8310
uniform code length	0.5226	1.0696	0.8520	0.5429	1.1133	0.8232	0.5638	1.1189	0.7963
w/o recorded geometry	0.2158	1.0701	1.3244	0.1727	1.0616	1.4279	0.1265	1.0672	1.5488

Alternation schedule. Table 6 compares the alternation against a single pass on the residual it minimises. A single pass raises the final per-unit J from 0.7176 to 0.7636 at $b = 0.55$ and the routed-token deviation from 0.8237 to 0.8542. The sweeps thus reach a residual the single pass does not reach at the same stored size. Each sweep refits the atoms against codes and maps already fit to them, a revision one pass never makes. Details in Appendix A.13.

Dictionary amortization. Table 5 decomposes the highest compression ratio reached over the budget grid against the expert count. At four experts no store in the class fits the budget and none is reported; from eight the ratio rises from 6.149 to 8.385 at sixteen and then holds between 8.348 and 8.413 across twenty-four, thirty-two and forty-eight, where AtomShare holds the highest ratio of the nine, 8.348 against 8.264 for per-matrix quantisation. More experts sharing one atom set amortize the dictionary. Past sixteen, the saving buys further atoms. Details in Appendix A.15.

Table 5: Highest compression ratio (↑) per expert count.

Experts	AtomShare	Vec. quant.
4	–	5.091
8	6.149	6.222
16	8.385	7.000
24	8.370	7.304
32	8.413	7.467
48	8.348	5.130

6 Related Work

Per-matrix low-rank and sparse compression. The dominant line compresses each weight matrix in its own basis, by low-rank truncation and factorization (Ren & Zhu, 2024; Saxena et al., 2025; Wang et al., 2025c) or by one-shot sparsification and pruning (Frantar & Alistarh, 2023; Sun et al., 2024; Li et al., 2023; Meng et al., 2024). These methods fit an expert against how far its own matrix moves and pay their stored cost once per matrix. AtomShare fits one atom set shared across the whole stack, where a direction recurring across experts and depths is paid for once.

Mixture-of-experts compression and routing. Work specific to sparse MoE models prunes or skips experts (Lu et al., 2024; Lee et al., 2025), merges them or splits them into low-rank or shared-plus-delta parts (Li et al., 2024; Yang et al., 2024; Gu et al., 2025), quantizes them below one bit (Frantar & Alistarh, 2024), reschedules them for serving (Wang et al., 2025a), or studies their routing (Avinash, 2026). AtomShare keeps every expert and its router, changes only how expert weights are stored, and prices a layer by the rate at which perturbing it reroutes tokens below.

Shared and residual structure. Codebooks share a dictionary across weights (van der Ouderaa et al., 2026), cross-layer basis sharing reuses one low-rank basis across layers (Wang et al., 2025b), and quantization keeps a low-rank or sparse residual beside its base (Mozaffari et al., 2025; Dettmers et al., 2024; Jeong et al., 2024; Park et al., 2026). AtomShare carries one stack-wide atom set into each layer by a norm-preserving map and prices each unit's code by its routed-token output.

7 Conclusion

We presented AtomShare, a weight store in which the experts of every layer draw on one dictionary, rotated into each layer by Householder reflectors, and each expert's code length is set by what that expert changes on the tokens routed to it. Against the strongest per-expert compressor it retains 0.5483 against 0.5150 at a lower bit cost, and it reaches a ratio of 8.385 once sixteen experts share the dictionary. Sharing one dictionary across experts and depths turns the stored size into a single budget that the routed output, and not the weight error, divides among the units.

REFERENCES

M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006. doi: 10.1109/TSP.2006.881199.

Mynampati Sri Ranganadha Avinash. Task-conditioned routing signatures in sparse mixture-of-experts transformers. arXiv, 2026.

Ruisi Cai, Yeonju Ro, Geon-Woo Kim, Peihao Wang, Babak Ehteshami Bejnordi, Aditya Akella, and Zhangyang Wang. Read-ME: refactorizing LLMs as router-decoupled mixture of experts with system co-design. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2024. Curran Associates Inc. doi: 10.52202/079017-3687.

Damai Dai, Chengqi Deng, Chenggang Zhao, R.X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y.K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024. doi: 10.18653/v1/2024.acl-long.70.

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *ICLR*, 2024.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23(1), January 2022. ISSN 1532-4435.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *ICML*, 2023.

Elias Frantar and Dan Alistarh. QMoE: Sub-1-bit compression of trillion parameter models. In *Conference on Machine Learning and Systems*, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Hao Gu, Wei Li, Lujun Li, Qiyuan Zhu, Mark Lee, Shengjie Sun, Wei Xue, and Yike Guo. Delta decompression for MoE-based LLMs compression. In *ICML*, 2025.

Geonhwa Jeong, Po-An Tsai, Stephen W. Keckler, and Tushar Krishna. SDQ: Sparse decomposed quantization for LLM inference. arXiv, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. arXiv, 2024.

Herve Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011. doi: 10.1109/TPAMI.2010.57.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Jaeseong Lee, Seung won Hwang, Aurick Qiao, Daniel F Campos, Zhewei Yao, and Yuxiong He. STUN: Structured-then-unstructured pruning for scalable MoE pruning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025. doi: 10.18653/v1/2025.acl-long.671.

Pingzhi Li, Zhenyu Zhang, Prateek Yadav, Yi-Lin Sung, Yu Cheng, Mohit Bansal, and Tianlong Chen. Merge, then compress: Demystify efficient SMoE with hints from its routing policy. In *ICLR*, 2024.

Yun Li, Lin Niu, Xipeng Zhang, Kai Liu, Jianchen Zhu, and Zhanhui Kang. E-Sparse: Boosting the large language model inference through entropy-based N:M sparsity. arXiv, 2023.

Xudong Lu, Qi Liu, Yuhui Xu, Aojun Zhou, Siyuan Huang, Bo Zhang, Junchi Yan, and Hongsheng Li. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024. doi: 10.18653/v1/2024.acl-long.334.

Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 689–696, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553463. URL https://doi.org/10.1145/1553374.1553463.

Xiang Meng, Kayhan Behdin, Haoyue Wang, and Rahul Mazumder. ALPS: Improved optimization for highly sparse one-shot pruning for large language models. In *Advances in Neural Information Processing Systems 37*, 2024. doi: 10.52202/079017-1187.

Zakaria Mhammedi, Andrew Hellicar, Ashfaqur Rahman, and James Bailey. Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 2401–2409. JMLR.org, 2017.

Mohammad Mozaffari, Amir Yazdanbakhsh, and Maryam Mehri Dehnavi. SLiM: One-shot quantization and sparsity with low-rank approximation for LLM weight compression. In *ICML*, 2025.

Yeonsik Park, Hyeonseong Kim, and Seungkyu Choi. SERQ: Saliency-aware low-rank error reconstruction for LLM quantization. arXiv, 2026.

Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, pp. 40–44 vol.1, 1993. doi: 10.1109/ACSSC.1993.342465.

Siyu Ren and Kenny Q. Zhu. Low-rank prune-and-factorize for language model compression. In *Proceedings of the Language Resources and Evaluation Conference*, 2024. doi: 10.63317/2pb377kvqads.

Utkarsh Saxena, Sayeh Sharify, Kaushik Roy, and Xin Wang. ResQ: Mixed-precision quantization of large language models with low-rank residuals. In *ICML*, 2025.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Sheng Shen, Zhewei Yao, Chunyuan Li, Trevor Darrell, Kurt Keutzer, and Yuxiong He. Scaling vision-language models with sparse mixture of experts. In *Findings of the Association for Computational Linguistics: EMNLP*, 2023. doi: 10.18653/v1/2023.findings-emnlp.758.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *ICLR*, 2024.

Tycho F. A. van der Ouderaa, Mart van Baalen, Paul Whatmough, and Markus Nagel. Leech lattice vector quantization for efficient LLM compression. arXiv, 2026.

Haodong Wang, Qihua Zhou, Zicong Hong, and Song Guo. D2MoE: Dual routing and dynamic scheduling for efficient on-device MoE-based LLM serving. In *Proceedings of the 31st Annual International Conference on Mobile Computing and Networking*, 2025a. doi: 10.1145/3680207.3723493.

Jingcun Wang, Yu-Guang Chen, Ing-Chao Lin, Bing Li, and Grace Li Zhang. Basis sharing: Cross-layer parameter sharing for large language model compression. In *ICLR*, 2025b.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *ICLR*, 2025c.

Cheng Yang, Yang Sui, Jinqi Xiao, Lingyi Huang, Yu Gong, Yuanlin Duan, Wenqi Jia, Miao Yin, Yu Cheng, and Bo Yuan. MoE-I2: Compressing mixture of experts models through inter-expert pruning and intra-expert low-rank decomposition. In *Findings of the Association for Computational Linguistics: EMNLP*, 2024. doi: 10.18653/v1/2024.findings-emnlp.612.

AI USE STATEMENT

We used generative AI tools to translate the text (Claude Opus 5.0), to check grammar and spelling and polish the wording (GPT-5.6), to guide the drawing of the figures (Gemini), and to check the content (Grok). We did not use generative AI tools to modify code. We have reviewed all AI-assisted work: we checked every reference for hallucinated citations. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The details needed to reproduce the results of this paper, including the weight stacks, the baseline systems, the budget scale, the evaluation metrics, the hyperparameters and the complete algorithm, are disclosed in this appendix.

APPENDIX

Table 7 lists the contents of each part of the appendix.

Table 7: What each part of the appendix holds.

Section	Contents	Section	Contents
Appendix A.0	Limitations	Appendix A.8	Hyperparameters and sensitivity
Appendix A.1	Extended related work	Appendix A.9	Extended budget grid
Appendix A.2	Motivation measurements	Appendix A.10	Transfer settings
Appendix A.3	Algorithms	Appendix A.11	Component ablation
Appendix A.4	Weight stacks	Appendix A.12	Secondary variants
Appendix A.5	Baseline systems	Appendix A.13	Alternation schedule
Appendix A.6	Budget scale	Appendix A.14	Complexity and stored-size account
Appendix A.7	Evaluation metrics	Appendix A.15	Dictionary amortization

APPENDIX A.0: LIMITATIONS

Shared-energy sweep. How much a shared atom set gains depends on how much structure the experts have in common, and a reader deciding whether to apply AtomShare to a new model needs to know that dependence. The paper lacks a sweep over the share of energy the experts have in common at a fixed stored-size target, giving the compression ratio of the shared and the per-unit store and the retained margin between them at each level. This sweep can be provided on request.

Routing concentration. Pricing a code by its routed-token deviation rests on the routed ordering of the units differing from their ordering by weight error, and a reader needs to see how the two orderings relate once the routing is changed. The paper lacks the rank correlation between the two orderings at six budgets, under normal routing and under forced-uniform routing. This comparison can be provided on request.

Boundary conditions. A reader applying AtomShare needs to know how it behaves on stacks unlike the primary one. The paper lacks a refit of AtomShare, one ablated variant and two per-unit baselines on a second stack whose experts share less energy, at one budget, and the stored size under stack-coupled and per-unit allocation on a single-layer stack over the expert count. Both can be provided on request.

APPENDIX A.1: EXTENDED RELATED WORK

Per-matrix low-rank and sparse compression. The dominant line compresses each weight matrix in its own basis: low-rank truncation and factorization (Ren & Zhu, 2024; Saxena et al., 2025; Wang et al., 2025c), one-shot sparsification and pruning (Frantar & Alistarh, 2023; Sun et al., 2024; Li et al., 2023; Meng et al., 2024), second-order post-training quantization (Frantar et al., 2022), and quantization with low-rank or sparse residuals (Dettmers et al., 2024; Jeong et al., 2024; Park et al., 2026). These methods fit an expert against how far its own matrix moves and pay their stored cost once per matrix. AtomShare departs on both counts: it fits one atom set shared across the whole stack, and hence a direction recurring across experts and depths is paid for once, and it prices each unit by the deviation it delivers at the layer output rather than by its matrix distance.

Mixture-of-experts compression and routing. Sparse MoE layers send each token to a few experts chosen by a learned gate (Shazeer et al., 2017; Fedus et al., 2022). Work specific to sparse MoE models prunes or skips whole experts (Lu et al., 2024; Lee et al., 2025), merges experts or decomposes them into low-rank parts or a shared base plus expert-specific deltas (Li et al., 2024; Yang et al., 2024; Gu et al., 2025), quantizes them below one bit per weight (Frantar & Alistarh, 2024), reschedules them for serving (Wang et al., 2025a), or studies the routing that selects them (Avinash, 2026). This work is orthogonal: we keep every expert and the router that selects among them, and change only how an expert's weights are stored; the assignment itself is recomputed on the compressed stack. The budget allocation prices a layer by the rate at which perturbing it turns the routing over below, connecting the store to the selection it feeds.

Shared structure and vector quantization. Codebook and vector-quantization methods share a dictionary across vectors or weights (Jégou et al., 2011; Egiazarian et al., 2024; van der Ouderaa et al., 2026), cross-layer parameter sharing ties the weights of whole layers together (Lan et al., 2020), cross-layer basis sharing reuses one low-rank basis across the layers of a model (Wang et al., 2025b), and adapter-style methods add a shared low-rank term to a sparse base (Mozaffari et al., 2025). AtomShare differs in what is shared and how it is carried: a stack-wide atom set is transported into each layer by a norm-preserving reflector map and the input geometry, which is how atoms fit at one depth are carried to another, and the code of each unit is priced against the routed-token output rather than reconstruction error alone.

Sparse coding and orthogonal parametrization. Sparse coding represents a signal as a sparse combination of atoms from a dictionary. Orthogonal matching pursuit grows the support one atom at a time and refits the coefficients on the whole support after each addition (Pati et al., 1993); K-SVD alternates sparse coding with updates of the dictionary atoms (Aharon et al., 2006); and online dictionary learning scales the atom update to large sample sets (Mairal et al., 2009). A product of Householder reflections gives an orthogonal matrix that is stored as its reflection vectors (Mhammedi et al., 2017). AtomShare applies both to a weight stack: one atom set is shared across the experts and depths of the stack, each layer reaches it through a map built from Householder reflectors, and the code of each unit is priced by the deviation it delivers on its routed tokens.

APPENDIX A.2: MOTIVATION MEASUREMENTS

The body figure carries the three readings side by side. This section prints each panel on its own, states what it was measured on, and gives the full grids behind it. All three sets are read on stacks instantiated for this section alone; none of them reuses a result-table run.

Appendix A.2.1: The left panel, cross-expert redundancy. Each layer's experts are fitted twice at equal stored size: once by truncating every expert matrix on its own, and once by fitting one set of directions jointly across the experts of the layer and storing only a coefficient vector per expert. Equal stored size caps the joint directions at 1, 2, 3, 4, 6, 7, 9 and 11 for 4 to 48 experts, since the shared directions are stored once for the group while each expert stores only its coefficients. Fig. 4 plots the ratio of the two captured energies, one line per depth. With eight experts the joint fit is worse at every depth, ratio 0.768 to 0.949: two shared directions do not pay for the dictionary they require. The ratio straddles one at twelve experts (0.890 to 1.026) and again at sixteen (0.999 to 1.074), and is above one at every depth from twenty-four, reaching 1.302 to 1.363 at forty-eight. Across depth the ratio has no trend and the per-depth differences change sign; the grid therefore carries no depth ordering. Table 9 prints all sixty-four rows, with the two captured energies themselves: per-matrix stays inside 40.29 to 43.74 across the grid while jointly captured climbs from 29.74 to 56.99; the movement is therefore in the expert count and not in the per-matrix energy.

How much the experts share is a property of the stack rather than of the fit, and we therefore sweep it. A stack is instantiated at each of seven shared-energy fractions and the smallest ratio over the eight depths is read at sixteen and at forty-eight experts. Fig. 5 plots the sweep and Table 8 prints it. The curve rises with the fraction at both counts and crosses one between sampled levels, at an interpolated 0.572 at sixteen experts and 0.249 at forty-eight; no sampled fraction lands on either crossing.

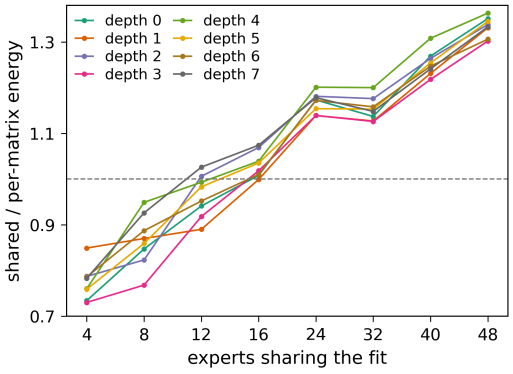

Figure 4: Cross-expert redundancy: energy captured by a jointly fitted set of directions over the energy captured by a per-matrix truncation at equal stored size, one line per layer depth. The dashed line marks ratio one, above which the shared fit wins.

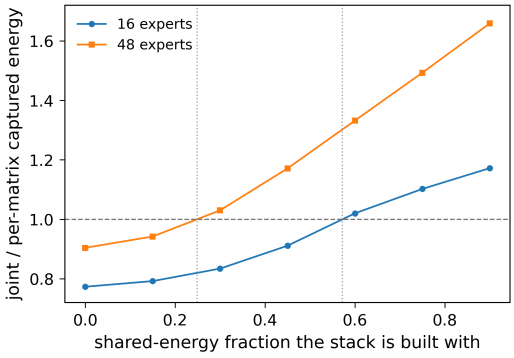

Figure 5: The shared-energy fraction at which one atom set starts to pay: smallest ratio over the eight depths against the shared-energy fraction the stack is instantiated with. The dotted verticals mark the interpolated crossings, 0.572 at sixteen experts and 0.249 at forty-eight.

Table 8: All readings of the share sweep: the smallest ratio over the eight depths of jointly captured energy to per-matrix captured energy, at two expert counts and seven shared-energy fractions, fourteen values. The ratio rises with the fraction at both counts and crosses one between sampled levels, at an interpolated 0.572 at sixteen experts and 0.249 at forty-eight.

Experts	0.00	0.15	0.30	0.45	0.60	0.75	0.90
16	0.773	0.792	0.834	0.911	1.020	1.102	1.172
48	0.904	0.942	1.030	1.171	1.332	1.493	1.659

Table 9: The redundancy grid in full: eight layer depths × eight expert counts, all sixty-four rows. k is the number of jointly fitted directions the per-matrix budget pays for at that expert count. Per-matrix captured energy and jointly captured energy are percentages at equal stored size; the ratio is the second over the first (↑). The ratio is below one at every depth with eight experts and is above one at every depth with forty-eight, reaching 1.302–1.363 there. The rise is not monotone: at every depth the ratio falls from twenty-four experts to thirty-two before rising again.

Depth	Experts	k	Per-matrix	Jointly	Ratio	Depth	Experts	k	Per-matrix	Jointly	Ratio
0	4	1	40.51	29.74	0.734	4	4	1	40.50	30.79	0.760
0	8	2	41.16	34.85	0.847	4	8	2	40.29	38.25	0.949
0	12	3	42.32	39.82	0.941	4	12	3	40.39	40.12	0.993
0	16	4	42.35	42.68	1.008	4	16	4	40.41	41.97	1.039
0	24	6	41.97	49.25	1.173	4	24	6	41.33	49.64	1.201
0	32	7	42.43	48.23	1.137	4	32	7	41.43	49.71	1.200
0	40	9	42.09	53.42	1.269	4	40	9	41.14	53.80	1.308
0	48	11	42.18	56.99	1.351	4	48	11	41.27	56.25	1.363
1	4	1	43.50	36.92	0.849	5	4	1	42.38	32.16	0.759
1	8	2	43.61	37.96	0.870	5	8	2	42.49	36.51	0.859
1	12	3	43.74	38.92	0.890	5	12	3	41.57	40.87	0.983
1	16	4	42.92	42.89	0.999	5	16	4	42.45	43.92	1.035
1	24	6	43.11	49.10	1.139	5	24	6	41.79	48.23	1.154
1	32	7	42.89	48.34	1.127	5	32	7	41.44	47.77	1.153
1	40	9	42.63	52.47	1.231	5	40	9	41.73	52.34	1.254
1	48	11	42.44	56.48	1.331	5	48	11	41.72	56.09	1.345
2	4	1	42.33	33.33	0.787	6	4	1	43.38	34.08	0.786
2	8	2	42.27	34.78	0.823	6	8	2	42.96	38.10	0.887
2	12	3	41.50	41.76	1.006	6	12	3	43.06	40.97	0.952
2	16	4	41.34	44.20	1.069	6	16	4	42.44	42.86	1.010
2	24	6	41.36	48.86	1.181	6	24	6	41.46	48.60	1.172
2	32	7	41.64	48.97	1.176	6	32	7	41.94	48.58	1.158
2	40	9	41.72	52.74	1.264	6	40	9	42.27	52.72	1.247
2	48	11	41.84	55.98	1.338	6	48	11	42.42	55.40	1.306
3	4	1	42.21	30.83	0.730	7	4	1	41.80	32.73	0.783
3	8	2	43.30	33.25	0.768	7	8	2	42.02	38.89	0.926
3	12	3	42.26	38.77	0.918	7	12	3	41.66	42.72	1.026
3	16	4	41.86	42.62	1.018	7	16	4	42.48	45.64	1.074
3	24	6	42.08	47.93	1.139	7	24	6	42.14	49.63	1.178
3	32	7	42.55	47.93	1.126	7	32	7	42.42	48.71	1.148
3	40	9	42.59	51.87	1.218	7	40	9	42.54	52.78	1.241
3	48	11	42.80	55.74	1.302	7	48	11	42.44	56.58	1.333

Appendix A.2.2: The middle panel, routed deviation against matrix error. Every expert of a layer is perturbed by an isotropic Gaussian draw with no compression family behind it, and two quantities are read per expert: the Frobenius norm of the change to its weight matrix, and the deviation that change induces at the layer output on the tokens the router sends that expert. Fig. 6 plots the Spearman rank correlation between the two, eight depths by six draws. Across the forty-eight readings the mean is 0.149, none exceeds 0.407, six fall below zero and the lowest is -0.202; Table 10 prints all forty-eight. The two read the same error differently: one weighs it on the tokens the expert receives, the other over the whole matrix. The reading belongs to this routing geometry and this perturbation family.

Appendix A.2.3: The right panel, routing flips below a perturbed layer. Only the layer-l weights are perturbed, at a stated relative Frobenius size, and the stack is run again from that layer; the reading is the fraction of tokens whose top-1 expert at layer $l{+}1$ changes. Fig. 7 plots seven depths by six perturbation sizes and Table 11 prints all forty-two. Depth 7 has no next layer and is not read. The rate runs 0.008 to 0.023 at a relative size of 0.05 and 0.102 to 0.198 at 0.40, and rises with the perturbation at every depth. The flip rates quoted for the scale of the flip term, 0.019 to 0.035, are the 0.10 column of this grid.

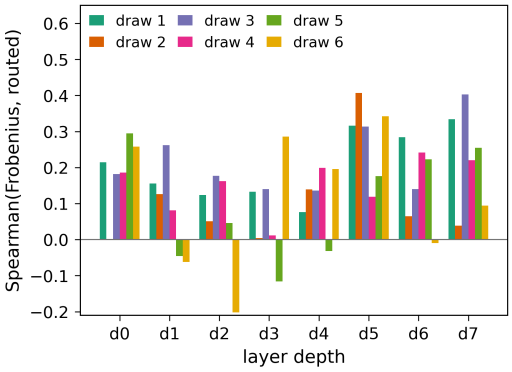

Figure 6: Routed deviation against matrix error: Spearman rank correlation between each expert's Frobenius reconstruction error and the deviation it delivers at the layer output on its routed tokens, six independent draws per depth. The axis is capped at 0.65, above every reading.

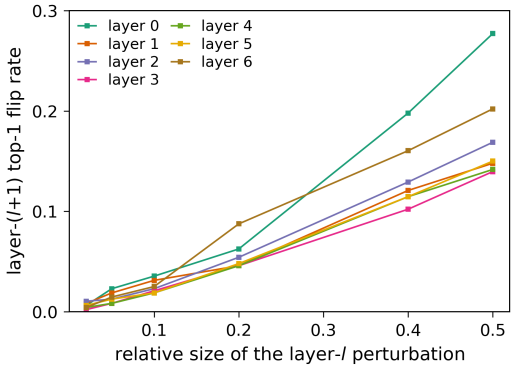

Figure 7: Routing flips below a perturbed layer: fraction of tokens whose top-1 expert at layer $l{+}1$ changes when only the layer-l weights are perturbed and the stack is run again from that layer, one line per depth.

Table 10: All readings of the rank-correlation sweep: Spearman rank correlation between per-expert Frobenius reconstruction error and the deviation delivered at the layer output on routed tokens, eight layer depths $\times$ six independent isotropic Gaussian draws of the perturbation, forty-eight values. No draw is a compression family. None exceeds 0.407, six fall below zero and the lowest is -0.202.

Depth	Draw 1	Draw 2	Draw 3	Draw 4	Draw 5	Draw 6
0	0.215	0.002	0.182	0.186	0.295	0.258
1	0.156	0.126	0.262	0.081	-0.046	-0.062
2	0.124	0.051	0.177	0.162	0.046	-0.202
3	0.133	0.004	0.140	0.012	-0.116	0.286
4	0.076	0.139	0.136	0.199	-0.032	0.196
5	0.316	0.407	0.314	0.119	0.176	0.342
6	0.284	0.065	0.140	0.242	0.223	-0.010
7	0.334	0.039	0.403	0.220	0.255	0.094

Table 11: All readings of the flip-rate sweep: fraction of tokens whose top-1 expert at layer $l{+}1$ changes when only the layer-l weights are perturbed at the stated relative Frobenius size and the stack is run again from that layer, seven layer depths $\times$ six perturbation sizes, forty-two values. Depth 7 has no next layer and is not read. The rate rises with the perturbation at every depth.

Depth	0.02	0.05	0.10	0.20	0.40	0.50
0	0.0063	0.0229	0.0354	0.0625	0.1979	0.2771
1	0.0083	0.0187	0.0312	0.0458	0.1208	0.1479
2	0.0104	0.0125	0.0229	0.0542	0.1292	0.1688
3	0.0021	0.0083	0.0208	0.0458	0.1021	0.1396
4	0.0042	0.0083	0.0187	0.0458	0.1146	0.1417
5	0.0063	0.0125	0.0187	0.0479	0.1146	0.1500
6	0.0042	0.0146	0.0250	0.0875	0.1604	0.2021

Appendix A.3: Algorithms

Algorithm 1 summarizes the complete AtomShare fit in three stages. Stage 1 records each layer's input geometry from one uncompressed pass, places the layer's experts in the metric it induces, starts every reflector map at the identity, and reads once for the stack the amplification ω_l each layer carries to the stack end and the rate ϕ_l at which perturbing it turns over the routing below. It then

initializes the shared atoms and sets a starting code length per unit from the words the budget leaves after the shared terms. Stage 2 runs $\tau_{\max}$ sweeps of one alternation: each sweep regrows every unit's support atom by atom with an exact coefficient refit after each addition, records the residual $J^{(\tau)}$, refits the shared atoms in closed form against every unit of the stack at once, and then sets and refines the reflectors of each layer map. Stage 3 holds the atoms and maps fixed, prices each unit at every code length up to $s^{\max}$ by the linear routed-deviation proxy of Eq. (10), gives every unit one atom and hands out the rest in decreasing return up to the count the budget admits, and checks the stored size against B.

Appendix A.4: Weight Stacks

Every reading in this paper is taken on a composed sparse MoE stack. Layer l receives the token matrix layer $l-1$ produced, tokens are routed to experts by nearest-centre scores, each token is mapped by the expert it reaches, and the result is added back to the residual stream. The stack therefore has a forward path: a layer-l weight perturbation reaches the layer-$(l{+}1)$ router and the end-of-stack response to a layer-l injection is measurable. Each layer reads the stream through its own anisotropic mixing; consequently, the token second moment arriving at a layer is that layer's own geometry G_l. A stack-wide set of twenty-four latent directions is shared across every depth and carried into each layer by an orthogonal frame; each expert is a short combination of them plus a per-expert residual, and the fraction of expert energy the combination carries is the shared-energy fraction, 0.548 for the primary, second and third stacks. The reference set holds $\max(240, 10E)$ tokens, that is, 480 at forty-eight experts. The primary stack is eight layers of forty-eight 14×14 experts, the second thirty-two experts at the same depth, and the third ten layers at forty-eight; the three are drawn at separate fixed seeds.

Appendix A.5: Baseline Systems

Eight per-expert compressors are read against AtomShare, each fitted to the same stored-size target and each storing one expert independently of the rest. Per-expert low-rank truncates the singular value decomposition of every expert, with one total rank budget spread over the matrices by the gain the next singular value would buy. Mixed-precision low-rank residual performs the same truncation in the layer's recorded input geometry rather than in weight coordinates. Low-rank plus sparse adapter stores a low-rank term together with a sparse residual on the entries the truncation leaves worst. Magnitude sparsification keeps the entries of largest absolute value and drops the rest; saliency-weighted sparsification keeps the entries of largest saliency instead; the two thus retain the same count under different orderings. Structured 2:4 sparsification keeps two entries of every four along a fixed axis, the pattern current sparse kernels execute. Per-matrix quantisation solves for the uniform bit width that meets the budget exactly on each matrix. Per-expert vector quantisation stores a codebook per expert and an index per block. The uncompressed stack is reported alongside them as the reference.

Appendix A.6: Budget Scale

A budget setting b names the stored-size target $B = LEpq(0.10 + 0.07b)$ words; it is thus a position on that scale rather than a retained fraction. The methods are not held to one stored size: each meets the setting in the form its own store takes, and the comparison is therefore read across the sizes each family lands on. A budget is met by an integer count of stored entries; hence two adjacent settings can land on one stored size and return one reading twice.

Appendix A.7: Evaluation Metrics

Five quantities are reported. The end-of-stack deviation runs the stack twice on the same reference tokens, once with its own weights and once with the store, and reads the relative displacement of the tokens leaving the last layer,

$$\Gamma(\Theta) = \frac{\left\| H_L(\Theta) - H_L \right\|_F}{\left\| H_L \right\|_F}, \tag{16}$$

where H_L stacks the tokens leaving layer L under the original weights and $H_L(\Theta)$ under the store. It is measured on the two runs, not propagated from per-unit errors, and it uses no recorded geometry.

Algorithm 1 AtomShare: fit one shared representation and divide one stored-size budget.

Require: stack $\{W_{l,e}\}$, reference set of T tokens, budget B covering the shared terms and one atom per unit, atom count $K \geq 2$, reflector count r, ridge λ, sweeps $\tau_{\max}$, word width w, code cap $s^{\max} = \max(2, \lfloor K/2 \rfloor)$
Ensure: Θ of Eq. (2) with $\mathcal{M}(\Theta) \leq B$

Stage 1: Common metric and once-per-stack reads
run the uncompressed stack on the reference set; record X_l, the routed rows $X_{l,e}$ and h_t
for each layer l **do**
  $G_l \leftarrow \frac{1}{T} X_l^\top X_l + \lambda I$ ▷ Eq. (3)
  factor $G_l^{1/2}$ and $G_l^{+1/2}$ on eigenvalues floored at 10^{-9} ▷ Eq. (4)
  $A_{l,e} \leftarrow W_{l,e} G_l^{1/2}$ for every expert e ▷ Eq. (3)
  $U_l, V_l \leftarrow I$ before the r reflectors of each map are set ▷ Eq. (5)
  $\Delta h_t^{(l)} \leftarrow \widehat{h}_t - h_t$ with only layer l in a rank-three probe store, for $l < L$ ▷ Eq. (11)
  $\omega_l \leftarrow \|\Delta h_{\text{end}}^{(l)}\| / \varepsilon_l^{\text{inj}}$ for $l < L$; $\omega_L \leftarrow 1$ ▷ Eq. (13)
  $\phi_l \leftarrow$ top-1 turnover at layer $l{+}1$ under a layer-l perturbation of stated relative size, for $l < L$ ▷ Eq. (12)
end for
$\phi_L \leftarrow 0$; initialize the shared atoms $\{D_k\}_{k=1}^K$ across all depths
$s_{l,e} \leftarrow$ words B leaves after the atom, map and geometry terms, divided by $1 + \lceil \log_2 K \rceil / w$, split by ω_l across layers and equally within a layer, rounded into $[1, s^{\max}]$ ▷ Eq. (14)

Stage 2: One alternation over codes, atoms and maps
for $\tau = 1$ to $\tau_{\max}$ **do**
  for each unit (l, e) **do** ▷ support regrown every sweep
    $S_{l,e} \leftarrow \emptyset$; $M_{l,e} \leftarrow 0$; $R_{l,e} \leftarrow U_l^\top A_{l,e} V_l$
    while $|S_{l,e}| < s_{l,e}$ **do**
      $k^\star \leftarrow \arg\max_{k \notin S_{l,e}} g_{l,e}(k)$; $S_{l,e} \leftarrow S_{l,e} \cup \{k^\star\}$ ▷ Eq. (8)
      refit $c_{l,e,\cdot}$ exactly on all of $S_{l,e}$ ▷ Eq. (9)
      $M_{l,e} \leftarrow \sum_{k \in S_{l,e}} c_{l,e,k} D_k$; $R_{l,e} \leftarrow U_l^\top A_{l,e} V_l - M_{l,e}$
    end while
  end for
  record $J^{(\tau)}$ after the code fit ▷ Eq. (15)
  refit $\{D_k\}$ in closed form jointly over every unit; refresh every $M_{l,e}$ ▷ Eq. (7)
  for each layer l: one setting pass, then four refinement passes **do**
    form $\Sigma_l = \sum_e A_{l,e} V_l M_{l,e}^\top$ and $\sum_e A_{l,e}^\top U_l M_{l,e}$ at the current maps
    set the reflectors $u_{l,i}$, $i = 1, \ldots, r$, of each map against its target ▷ Eq. (6)
    replace U_l and V_l together
  end for
end for

Stage 3: Routed-deviation pricing and one budget across the stack
for each unit (l, e), atoms and maps fixed **do**
  regrow $S_{l,e}$ to $s^{\max}$ as in Stage 2; record $\widehat{W}_{l,e}(s)$ at each length s ▷ Eq. (4)
  $\delta_{l,e}(s) \leftarrow$ Eq. (10) at $\widehat{W}_{l,e}(s)$, on the four rows nearest the centre if fewer than four are routed
end for
$m_l \leftarrow$ median of $\delta_{l,e}(1)$ over the experts of layer l
return of the next atom of (l, e) at length s: $\omega_l \max\{\delta_{l,e}(s) - \delta_{l,e}(s{+}1), 0\}$ ▷ Eq. (13)
multiply that return by $(1 + \phi_l)$ when $\delta_{l,e}(1) > m_l$ ▷ Eq. (12)
give every unit one atom, then each further atom to the largest return ▷ greedy on $\widetilde{\Gamma}$
stop at the atom count B admits at a uniform per-unit length, capping each unit at $s^{\max}$
$s_{l,e} \leftarrow$ atoms given; install the recorded code of each unit at $s_{l,e}$
check $\mathcal{M}(\Theta) \leq B$ ▷ Eq. (14)
return Θ ▷ Eq. (2)

The retained score reads the aligned reconstruction residual in each layer's own input geometry,

$$\rho(\Theta) = \frac{\sum_{l,e} \left\| (W_{l,e} - \widehat{W}_{l,e}) G_l^{1/2} \right\|_F^2}{\sum_{l,e} \left\| W_{l,e} G_l^{1/2} \right\|_F^2}, \qquad \text{Ret}(\Theta) = \exp(-0.9\, \rho(\Theta)), \tag{17}$$

which is 1.000 at zero residual and falls as the residual grows. Because it is read in the metric one component records, it is not a neutral rubric for the variant that removes that component, and the

Table 12: Stored size in 16-bit words that each compressed system lands on at the three body budget settings, with the budget itself in the last row. The families are not held to one stored size: each meets the setting in its own form, and the main comparison is accordingly read across the sizes recorded here. AtomShare holds the account of Eq. (14) inside the budget at every setting; five of the nine systems land above it at one or more settings, vector quantisation by the widest margin. Three pairs of rows coincide by construction rather than by coincidence: the two low-rank systems spend the same rank budget, the two sparsification systems keep the same number of entries, and per-matrix quantisation solves for the bit width that meets B exactly.

System	$b = 0.35$	$b = 0.55$	$b = 0.80$
Per-expert low-rank	10752	10752	11732
Mixed-precision low-rank residual	10752	10752	11732
Low-rank plus sparse adapter	11328	11328	11904
Magnitude sparsification	9216	10368	11520
Saliency-weighted sparsification	9216	10368	11520
Structured 2:4 sparsification	9408	10584	12096
Per-matrix quantisation	9370	10424	11741
Per-expert vector quantisation	14672	14784	14896
AtomShare (ours)	9016	10024	11536
Budget B	9370	10424	11741

variants are separated on the routed-token relative deviation instead,

$$\mathrm{RRD}(\Theta) = \frac{\mathrm{mean}_{l,e}\big\|X_{l,e}(W_{l,e} - \widehat{W}_{l,e})^\top\big\|_F \big/ \sqrt{n_{l,e}}}{\mathrm{mean}_{l,e}\big\|X_{l,e}W_{l,e}^\top\big\|_F \big/ \sqrt{n_{l,e}}}, \tag{18}$$

where $X_{l,e}$ holds the $n_{l,e}$ layer-l rows the router sends to expert e. It uses no recorded geometry and is read only on the tokens an expert actually receives. The last two are size readings,

$$\mathrm{CR}(\Theta) = \frac{LEpq}{\mathcal{M}(\Theta)}, \qquad \mathrm{bpp}(\Theta) = \frac{w}{\mathrm{CR}(\Theta)}, \tag{19}$$

with $\mathcal{M}(\Theta)$ the account of Eq. (14) in words and $w = 16$ the word width; the effective bits per parameter is thus what the store spends on each weight it replaces.

Appendix A.8: Hyperparameters and Sensitivity

The configuration read is twenty-eight shared atoms (except where a sweep or a variant states another count), three Householder reflectors per layer, a metric ridge $\lambda = 0.05$, eigenvalues of G_l floored at 10^{-9} before the inverse square root, $\tau_{\max} = 6$ alternation sweeps, four refinement passes per layer map once its reflectors are set, sixteen-bit stored words, and a rank-three probe store for ω_l. A unit routed fewer than four rows is priced on the four rows nearest its centre, and thus an expert with no traffic still carries a price. The two counts the store is most sensitive to are swept below. The budget of Eq. (1) is enforced on the account of Eq. (14): the code length is the largest that keeps $\mathcal{M}(\Theta) \le B$, and the atom count is the largest count from two up to the stated count that still leaves every unit a one-atom code inside B. Where even that smallest form does not fit, no store in the class is reported at that setting and the cell is left empty. Table 13 sweeps the shared-atom count and the per-layer reflector count over six instances. Averaged here over the six instance readings the table prints, the retained score runs from 0.5027 at twelve atoms to 0.5313 at twenty, and from 0.5234 to 0.5335 across the five reflector counts. Both sweeps are read at a single budget of 0.45 on six stack instances of their own, none of them the primary stack; consequently, the levels here are not the primary stack's readings and are only comparable within this table. The atom sweep holds three reflectors and the reflector sweep holds twenty-two atoms. Table 14 resizes the reference set the fit reads over five sizes on another six instances; averaged here over the six instance readings the table prints, the retained score stays between 0.5159 at 192 and 0.5202 at 768, and the fit is therefore not read as sensitive to the reference-set size over that range.

Table 13: Sensitivity of the retained score (↑) to the shared-atom count and the per-layer reflector count, over six instances. The atom sweep has an interior maximum at twenty atoms; the reflector sweep peaks at three. Both are read at a single budget of 0.45, the atom sweep at three reflectors and the reflector sweep at twenty-two atoms. This paper runs up to twenty-eight atoms, which is not the maximum of its sweep, and three reflectors, which is.

Setting	Instance 0	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5
12 atoms	0.5033	0.5029	0.5025	0.5004	0.5047	0.5022
16 atoms	0.5272	0.5251	0.5249	0.5227	0.5261	0.5241
20 atoms	0.5310	0.5321	0.5322	0.5303	0.5314	0.5311
24 atoms	0.5288	0.5290	0.5252	0.5251	0.5281	0.5274
28 atoms	0.5165	0.5209	0.5116	0.5165	0.5156	0.5162
36 atoms	0.5047	0.5066	0.5036	0.5027	0.5047	0.5046
1 reflector	0.5292	0.5296	0.5281	0.5269	0.5307	0.5301
2 reflectors	0.5309	0.5310	0.5289	0.5289	0.5292	0.5301
3 reflectors	0.5344	0.5337	0.5335	0.5324	0.5337	0.5335
4 reflectors	0.5248	0.5240	0.5219	0.5229	0.5238	0.5229
5 reflectors	0.5282	0.5256	0.5241	0.5242	0.5270	0.5248

Table 14: Retained score (↑) against the size of the reference set the fit reads, over six instances.

Reference-set size	Instance 0	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5
48	0.5206	0.5151	0.5187	0.5165	0.5209	0.5214
96	0.5207	0.5182	0.5175	0.5160	0.5148	0.5217
192	0.5171	0.5152	0.5169	0.5145	0.5175	0.5139
384	0.5184	0.5188	0.5159	0.5174	0.5150	0.5193
768	0.5196	0.5186	0.5232	0.5189	0.5205	0.5203

Appendix A.9: Extended Budget Grid

AtomShare reaches a retained score of 0.5135 at the tightest setting of the main benchmark and 0.5483 at the loosest, and the margin over the strongest per-expert baseline reads 0.0081 at the one setting and 0.0333 at the other.

Table 15 extends the main table to eight stored-size budgets on the primary stack, the five-budget appendix grid and the three-budget body grid merged; AtomShare's retained score rises at every step that moves the stored size, from 0.5135 at $b = 0.30$ to 0.5587 at $b = 0.90$, and holds still across the one pair that lands on one stored size, $b = 0.30$ with $b = 0.35$ at 0.5135. The body table's three budgets sit on it. Table 16 gives the effective bits per parameter at the same eight settings; AtomShare's runs from 1.92 at $b = 0.30$ to 2.56 at $b = 0.90$.

Appendix A.10: Transfer Settings

The transfer panel applies the settings chosen on the primary stack unchanged elsewhere. The three settings are separate weight stacks, each retained score read against its own residual; the levels are therefore not comparable across the columns. What the columns carry is the ordering inside each of them: AtomShare leads the panel on the primary and the larger setting and is behind the mixed-precision low-rank residual on the second, at a higher compression ratio, 7.66 against 7.00.

Appendix A.11: Component Ablation

The mechanism ablation removes one component at a time at three budget settings and reports three quantities. Γ is what Eq. (1) asks a store to minimise, and it is measured here exactly as written; on this stack it is saturated, every variant reading between 1.04 and 1.12. Among saturated values the ordering is not a quality ordering, and at $b = 0.45$ the variant with the worst store reads a lower Γ than the full method, 1.0475 against 1.0786, at a retained score of 0.1862 against 0.5158. The

Table 15: Extended budget grid: retained score (↑) at eight stored-size budget settings, the appendix grid and the body grid merged. AtomShare lands on seven distinct stored sizes over the eight settings, $b = 0.30$ and $b = 0.35$ falling on one, and its retained score rises at every step that moves the stored size. The sparsification rows do not rise with the budget, and hence the grid is not monotone as a whole. The body table's three settings sit on it.

	Budget setting b							
System	0.30	0.35	0.45	0.55	0.60	0.75	0.80	0.90
Uncompressed reference	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Per-expert low-rank	0.4708	0.4708	0.4708	0.4708	0.4708	0.4762	0.4779	0.4814
Mixed-precision low-rank residual	0.5054	0.5054	0.5054	0.5054	0.5054	0.5127	0.5150	0.5199
Low-rank plus sparse adapter	0.3444	0.3444	0.3444	0.3444	0.3444	0.3444	0.2810	0.2440
Magnitude sparsification	0.0210	0.0210	0.0210	0.0207	0.0205	0.0207	0.0207	0.0211
Saliency-weighted sparsification	0.0311	0.0311	0.0311	0.0308	0.0300	0.0294	0.0294	0.0298
Structured 2:4 sparsification	0.0069	0.0069	0.0069	0.0061	0.0061	0.0057	0.0057	0.0057
Per-matrix quantisation	0.0006	0.0011	0.0049	0.0122	0.0189	0.0514	0.0666	0.1020
Per-expert vector quantisation	0.4484	0.4484	0.4514	0.4514	0.4514	0.4514	0.4558	0.4558
AtomShare (ours)	0.5135	0.5135	0.5158	0.5235	0.5335	0.5397	0.5483	0.5587

Table 16: Extended budget grid: effective bits per parameter (↓) at the same eight stored-size budget settings.

	Budget setting b							
System	0.30	0.35	0.45	0.55	0.60	0.75	0.80	0.90
Uncompressed reference	16.00	16.00	16.00	16.00	16.00	16.00	16.00	16.00
Per-expert low-rank	2.29	2.29	2.29	2.29	2.29	2.44	2.49	2.61
Mixed-precision low-rank residual	2.29	2.29	2.29	2.29	2.29	2.44	2.49	2.61
Low-rank plus sparse adapter	2.41	2.41	2.41	2.41	2.41	2.41	2.53	2.65
Magnitude sparsification	1.96	1.96	2.08	2.20	2.33	2.45	2.45	2.57
Saliency-weighted sparsification	1.96	1.96	2.08	2.20	2.33	2.45	2.45	2.57
Structured 2:4 sparsification	2.00	2.00	2.00	2.25	2.25	2.57	2.57	2.57
Per-matrix quantisation	1.94	1.99	2.10	2.22	2.27	2.44	2.50	2.61
Per-expert vector quantisation	3.12	3.12	3.14	3.14	3.14	3.14	3.17	3.17
AtomShare (ours)	1.92	1.92	2.02	2.13	2.24	2.35	2.45	2.56

retained score is read in each layer's own input geometry, which is what one component records; it is therefore not a neutral rubric for the variant that removes that component: it puts that variant at a more than threefold drop, 0.5335 to 0.1502 at $b = 0.65$, where the routed reading puts the same removal at a rise of 83 per cent. The variants are separated on the routed-token deviation for that reason, and the quantity the allocation is computed against is neither of the two, being the weighted sum $\widetilde{\Gamma}$ of Eq. (13). Fig. 8 plots the variants across the three budgets.

Appendix A.12: Secondary Variants

The uniform-code-length variant reads above the full method on the retained score at all five distinct operating points this paper reads it at, which six budget labels cover, by 0.0091 to 0.0194 among the secondary variants and by 0.0155 to 0.0173 in the mechanism ablation. On the routed-token deviation, the column the variants are separated on, it reads below the full method at 0.8232 against 0.8237 at $b = 0.55$ and at 0.7963 against 0.8022 at $b = 0.80$ and $b = 0.85$, which land on one stored size and return the same reading twice. At the other three operating points the full method reads below it on that column, 0.8314 against 0.8520 at $b = 0.35$, 0.8282 against 0.8375 at $b = 0.45$ and 0.8099 against 0.8140 at $b = 0.65$. The routed-deviation-pricing variant likewise reads 0.0004 and 0.0001 below the full method at the two tighter budgets and 0.0004 above it at the loosest.

The flip term of Eq. (12) raises the price of the units whose routed deviation sits above their layer's median, by that layer's measured flip rate, and on this stack it changes no allocation: the dropped-flip-term variant returns the same retained score and the same Γ as the full method at all three settings. At the perturbation size the term is read at, the motivation stack returns flip rates of 0.019 to 0.035.

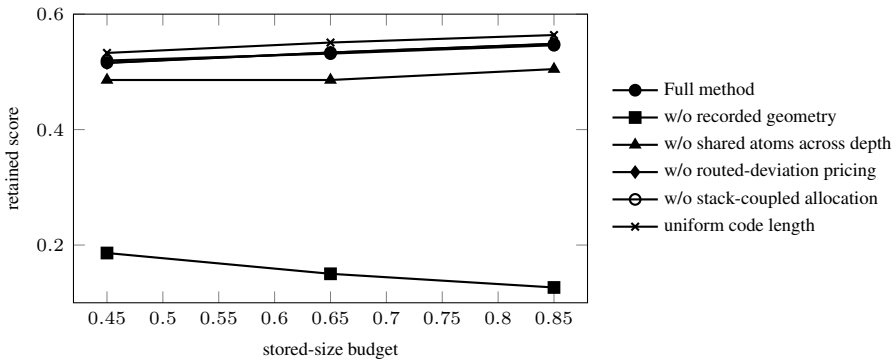

Figure 8: Each ablation variant across the three budgets. The no-geometry variant sits far below the rest at all three; the routed-pricing and stack-coupled-allocation variants differ from the full method on the retained score by at most 0.0038, which is the stack-coupled variant's 0.5196 against 0.5158 at $b = 0.45$; those three traces are thus not separable by eye.

The secondary-variant table reports four variants beside the full method. The twelve-atom variant holds twelve shared atoms in place of twenty-eight and returns 0.5016 at all three budgets, its code length already at the cap that atom count sets. Read on the routed-token deviation the twelve-atom variant reads 0.8716 throughout, the single-pass variant 0.8658 to 0.8310 and the no-geometry variant 1.3244 to 1.5488, all above the full method; the uniform-code-length variant is above it at the tightest budget, 0.8520, and below it at the other two, 0.8232 and 0.7963. Read on the retained score the first three are below the full method and the uniform-code-length variant is above it at every setting.

Appendix A.13: Alternation Schedule

The representation is fit as one alternation over the three blocks rather than matrix by matrix, and the single-pass row of the secondary-variant table reads that schedule against one pass on the residual J the sweeps minimise. A single pass raises the final per-unit J at every setting, from 0.7391 to 0.7996 at $b = 0.35$, from 0.7176 to 0.7636 at $b = 0.55$ and from 0.6662 to 0.7107 at $b = 0.80$, and the routed-token deviation from 0.8237 to 0.8542 at $b = 0.55$, and hence the sweeps buy a residual the single pass does not reach at the same stored size.

Appendix A.14: Complexity and Stored-Size Account

The cost reading counts a materialisation cost and a peak resident cost at the same stored-size budget. AtomShare's 9212 multiply-accumulates are the largest count of the ten systems, and its 9716 peak resident words are the lowest but one: structured sparsity holds 9604, the remaining seven compressed systems run from 9988 to 14980, and the uncompressed stack holds 75460. Magnitude and saliency-weighted sparsification keep the same number of entries and restore them by the same scatter; both counts therefore coincide for the two. Both are counted off the reconstruction each method performs; neither is a timing taken on serving hardware, and no throughput or latency is reported. Rebuilding one $p \times q$ expert from an AtomShare store costs $O(s_{l,e}pq)$ multiply-accumulates to expand the sparse code against the atoms and three dense products to apply U_l, $V_l^\top$ and $G_l^{+1/2}$, which is 9212 operations at the configuration read; a sparse or vector-quantised store restores an expert by scatter or lookup and spends none. The resident cost is the stored size $\mathcal{M}(\Theta)$ plus one dense pq buffer for the expert being materialised.

Table 18 breaks the stored size into its five components at six operating points. The atom store, the layer maps and the geometry factors are 5488, 672 and 840 words at every point; consequently, the whole of the budget movement is in the code: the coefficients run 768 to 3840 words and the atom indices that go with them 240 to 1200. The five sum to the stored size the compression ratio is read against, printed as the last row: 9520 words at $b = 0.45$, which is 196 words below the peak resident count, the difference being the one dense buffer that count adds. Table 19 reports the retained-score

Table 17: Where each cost of the fit is charged. The frequency column states how often each cost is paid; none of the first three is charged against the per-unit code budget that $\mathcal{M}(\Theta) \leq B$ constrains.

Operation	Charged	Frequency
Layer geometry G_l	fit lifetime	once per layer
Reflector eigenproblems, $2Lr$ per pass	fit lifetime	five passes each sweep
Atom and code refit	fit lifetime	once per sweep
Stored codes $s_{l,e}$	budget B	held at $\mathcal{M}(\Theta) \leq B$

Table 18: Stored-size account by component, in stored words, at six operating points. The five components are the whole of the stored size; the last row is their sum, the $\mathcal{M}$ the compression ratio is read against at every point, and it stays inside the budget at each.

	Operating point					
Component	0.10	0.25	0.45	0.65	0.80	0.95
Atoms	5488	5488	5488	5488	5488	5488
Layer maps	672	672	672	672	672	672
Geometry factors	840	840	840	840	840	840
Coefficients	768	1152	1920	2688	3456	3840
Support indices	240	360	600	840	1080	1200
Total	8008	8512	9520	10528	11536	12040

drift under a change of six configuration fields. Under the atom count and the reflector count the AtomShare row varies a real quantity of the method, the atom count to 24 and the reflector count to 2, while the nine baseline rows have no counterpart to vary, and their zero entries in those two columns therefore record a field that does not reach them. The vector-quantisation baseline is the exception: its own draw differs, which is why its two cells are nonzero. The other four columns are read for every row.

Appendix A.15: Dictionary Amortization

Table 20 prints the highest ratio reached over the budget grid against experts per layer for all nine compressed systems; the body keeps AtomShare and vector quantisation, the one per-expert compressor whose ratio moves with the expert count. No store in AtomShare's class fits the budget at four experts; the ratio it reaches rises from 6.149 at eight to 8.385 at sixteen before flattening between 8.348 and 8.413, while every per-expert compressor except vector quantisation holds one ratio across the range.

Table 20: Highest compression ratio reached over the budget grid (↑) against experts per layer for all nine compressed systems: per-expert low-rank (PE-LR), mixed-precision low-rank residual (MP-LR), low-rank plus sparse adapter (LR+S), magnitude (Mag) and saliency-weighted (Sal) sparsification, structured 2:4 sparsity, per-matrix quantisation (PMQ) and per-expert vector quantisation (VQ). A dash marks a setting at which no store in AtomShare's class fits the budget.

Experts	PE-LR	MP-LR	LR+S	Mag	Sal	2:4	PMQ	VQ	Ours
4	7.000	7.000	6.644	8.167	8.167	8.000	8.264	5.091	–
8	7.000	7.000	6.644	8.167	8.167	8.000	8.264	6.222	6.149
16	7.000	7.000	6.644	8.167	8.167	8.000	8.264	7.000	8.385
24	7.000	7.000	6.644	8.167	8.167	8.000	8.264	7.304	8.370
32	7.000	7.000	6.644	8.167	8.167	8.000	8.264	7.467	8.413
48	7.000	7.000	6.644	8.167	8.167	8.000	8.264	5.130	8.348

Table 19: Retained-score drift (↓) per configuration field, six fields. The atom-count and reflector-count columns carry the same caveat: the AtomShare row moves a stated quantity of the method while the nine baseline rows have no counterpart to move, the vector-quantisation baseline excepted.

System	Second stack draw	Reference batch	Atom count	Reflector count	Budget	Stack instance set
Uncompressed reference	0.000	0.000	0.000	0.000	0.000	0.000
Per-expert low-rank	0.000	0.002	0.000	0.000	0.000	0.001
Mixed-precision low-rank residual	0.001	0.004	0.000	0.000	0.000	0.000
Low-rank plus sparse adapter	0.034	0.005	0.000	0.000	0.000	0.008
Magnitude sparsification	0.005	0.002	0.000	0.000	0.000	0.003
Saliency-weighted sparsification	0.006	0.002	0.000	0.000	0.000	0.002
Structured 2:4 sparsification	0.002	0.001	0.000	0.000	0.001	0.000
Per-matrix quantisation	0.002	0.001	0.000	0.000	0.006	0.003
Per-expert vector quantisation	0.002	0.002	0.001	0.001	0.003	0.000
AtomShare (ours)	0.004	0.003	0.013	0.006	0.008	0.002