TRUESLICE: ZERO INFERENCE OVERHEAD PRUNING VIA GLOBAL RESIDUAL STREAM ROTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

To alleviate the significant memory and computational costs of large language models, numerous structural pruning methods have emerged, including residual stream width reduction. However, some residual width pruning methods introduce architectural changes and additional inference overhead. We propose TrueSlice, a calibration-based method that reduces residual width using a single global rotation derived from trace-normalized activation second moments. By retaining the original model architecture, TrueSlice can be applied to both standard transformer and modern hybrid networks, such as those in the Qwen3.5 family. Trace normalization balances contributions across the network, while the shared rotation enables slicing without additional residual rotation parameters or operations at inference. At matched total parameter reduction targets of 10–30%, TrueSlice achieves lower WikiText-2 perplexity than SliceGPT in 17 of 24 comparisons across eight Llama 3 and Qwen3 models without recovery training. On Llama-3.1-8B at 30% target reduction, TrueSlice achieves higher average task accuracy than SliceGPT after 3B recovery tokens. TrueSlice also complements attention head and MLP feature pruning: at a 50% target parameter reduction on Llama-3.1-8B, the best tested combination with Týr-the-Pruner raises average accuracy across five reasoning benchmarks from 44.86% to 48.01%, a gain of 3.15 percentage points over Týr alone.

1 INTRODUCTION

The substantial memory and computational costs of large language models (LLMs) motivate compression methods producing smaller models suitable for standard and efficient inference implementations. Residual stream width pruning is one such pathway to this goal. It removes dimensions from a model's residual stream and corresponding rows and columns from its dense weight matrices. However, reducing parameter count alone does not determine deployment efficiency because modifications to the model's computational structure can introduce additional inference operations. Residual width pruning therefore comprises three objectives: preserving model quality, reducing memory and computational requirements, and retaining the original architectural structure.

We introduce TrueSlice, a calibration-based residual width pruning method built on the rotation and slicing framework of SliceGPT (Ashkboos et al., 2024). Its central contribution is trace normalization on calibration statistics for selecting a shared residual subspace across the network, thereby balancing contributions from measurement locations with different activation magnitudes. Using this subspace throughout the model preserves ordinary residual connections and dense projections at reduced width. The resulting model incurs *zero inference overhead*: it requires no additional rotation operations or residual adapters. TrueSlice requires calibration data, but gradient-based recovery training is optional as a post-pruning step. This structure also supports application beyond standard attention models, as demonstrated on Qwen3.5-4B, a hybrid model incorporating Gated DeltaNet layers (QwenTeam, 2026; Yang et al., 2025b).

Our experiments evaluate compression quality, inference performance, and compatibility with complementary pruning methods. Trace normalization lowers perplexity in 17 of 18 ablation comparisons. Across eight Llama 3 and Qwen3 models (Grattafiori et al., 2024; Yang et al., 2025a), TrueSlice achieves lower perplexity than SliceGPT in 17 of 24 comparisons at matched total parameter reductions, including 15 of 16 at 10-20%. However, SliceGPT typically outperforms TrueSlice at

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.

30%. Across the three models benchmarked at a 30% target reduction, TrueSlice achieves higher throughput and lower per-token latency and time to first token than SliceGPT. Finally, TrueSlice is demonstrated to complement attention head and MLP feature pruning. At 50% total reduction on Llama-3.1-8B, the best tested combination with Týr-the-Pruner raises average accuracy across five benchmarks from 44.86% to 48.01%. These results establish trace-normalized global subspace selection as an effective approach to residual width pruning and a useful component of broader structural compression pipelines.

2 BACKGROUND

We introduce the transformer architecture and notation used throughout the paper. We then review related rotation-based compression methods, with particular emphasis on SliceGPT. We finish by discussing structural pruning and recovery frameworks that target different dimensions of transformer networks.

2.1 TRANSFORMER ARCHITECTURE AND NOTATION

To introduce the notation, we consider decoder-only transformer networks derived from the architecture of Vaswani et al. (2017), using the Llama 3 (Grattafiori et al., 2024) family as a representative example. These models consist of token embeddings, L pre-norm transformer layers, a pre-head normalization, and a language modeling (LM) head. We denote $\mathbf{X} \in \mathbb{R}^{N \times D_{\text{model}}}$ as the residual stream activations, where N is the sequence length, and D_{model} is the residual stream width.

Residual stream and normalization. As depicted in Figure 1, in sequential pre-norm transformers, each transformer layer ℓ is composed of a self-attention branch followed by an MLP branch:

$$\begin{aligned} \mathbf{X}'_\ell &= \mathbf{X}_{\ell-1} + \text{Attn}_\ell\big(\text{Norm}_{\ell,a}(\mathbf{X}_{\ell-1})\big), \\ \mathbf{X}_\ell &= \mathbf{X}'_\ell + \text{MLP}_\ell\big(\text{Norm}_{\ell,m}(\mathbf{X}'_\ell)\big). \end{aligned} \tag{1}$$

The placement of normalization within the branches, rather than following the residual addition, distinguishes pre-norm from post-norm architectures (Xiong et al., 2020). In parallel pre-norm transformers, both branches receive normalized versions of the same layer input, and their outputs are added to that input. Llama 3 uses RMSNorm (Zhang & Sennrich, 2019) for all normalization sublayers. For each token representation $\mathbf{x}$, it computes

$$\text{RMSNorm}(\mathbf{x}) = \frac{\mathbf{x}}{\sqrt{(\|\mathbf{x}\|_2^2 / D_{\text{model}}) + \epsilon}} \, \text{diag}(\boldsymbol{\gamma}), \tag{2}$$

where $\boldsymbol{\gamma}$ is a learned parameter vector and ϵ is a numerical stabilizer. We distinguish this learned normalization from the parameterless operation obtained during preprocessing in Section 3.

Attention and MLP projections. Llama 3 uses causal grouped-query attention (Ainslie et al., 2023) with query, key, and value projections $\mathbf{W}_\text{q}$, $\mathbf{W}_\text{k}$, and $\mathbf{W}_\text{v}$, and an output projection $\mathbf{W}_\text{o}$. Its SwiGLU (Shazeer, 2020) MLP uses gate and up projections, $\mathbf{W}_\text{gate}$ and $\mathbf{W}_\text{up}$, followed by a down projection $\mathbf{W}_\text{down}$. We denote projections reading from and writing to the residual stream by $\mathbf{W}_\text{in}^m$ and $\mathbf{W}_\text{out}^m$, with m indexing attention and MLP sublayers. The input notation covers all three query, key, and value projections or both gate and up projections. The residual width, attention head dimensions, and MLP intermediate dimension correspond to different projection axes and therefore distinct structural pruning targets.

Embeddings and language-model head. The embedding matrix $\mathbf{E}_0 \in \mathbb{R}^{V \times D_{\text{model}}}$, where V is the vocabulary size, maps token indices to residual stream representations. After the final layer, Llama 3 applies RMSNorm before the head $\mathbf{H}_0 \in \mathbb{R}^{D_{\text{model}} \times V}$ produces the vocabulary logits:

$$\mathbf{Y} = \text{RMSNorm}_f(\mathbf{X}_L)\mathbf{H}_0. \tag{3}$$

Some models use tied embeddings to reduce their parameter count, sharing the embedding and head weights and imposing $\mathbf{H}_0 = \mathbf{E}_0^\top$ (Press & Wolf, 2017). We distinguish these original matrices from the preprocessed matrices $\mathbf{E}$ and $\mathbf{H}$ used in Section 3.

2.2 Related Work

Rotation-based and subspace pruning. SliceGPT (Ashkboos et al., 2024) performs structural compression by reducing the residual stream width of transformer models. It uses PCA to compute orthonormal bases from calibration activations and removes coordinates associated with smaller eigenvalues, thereby constructing smaller, rotated and sliced dense projections. Although its established computational invariance permits a common orthogonal rotation, SliceGPT selects a different basis for each sublayer to accommodate differences in their activations' principal directions. These different bases require transformations along the residual connections to map between the different coordinate systems, introducing additional parameters and computational overhead. SliceGPT provides the theoretical and implementation foundation for TrueSlice. We formalize TrueSlice in Section 3.

Other works also investigate shared or learned rotations, with La RoSA (Liu et al., 2025) applying layer-wise PCA rotations for dynamic activation sparsification. In an ablation, it evaluates a shared global rotation obtained by averaging input covariance matrices across layers. This eliminates residual adapters but worsens perplexity relative to its baseline in the reported experiment, leading the authors to favor layer-wise rotations. This ablation explores shared bases derived from pooled activation statistics without describing layer-wise trace normalization or permanent residual stream width reduction. RotPruner (Chen & Wang, 2025) instead learns orthogonal rotations through autoregressive and distillation objectives with pruning masks applied during optimization. It similarly studies rotation sharing as a tradeoff between basis flexibility and residual transformation overhead.

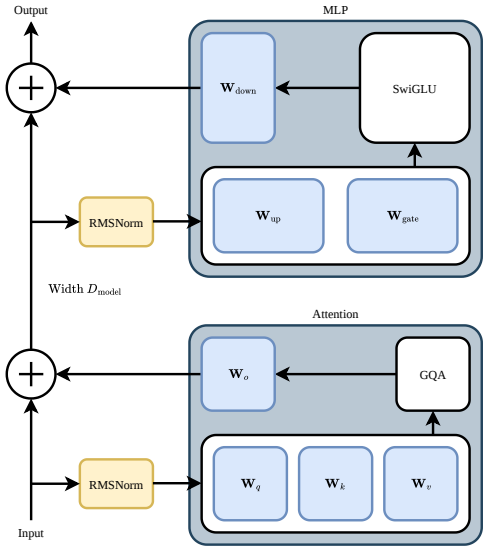

Figure 1: A Llama-style transformer layer, adapted from Ashkboos et al. (2024, Figure 2), showing the attention and MLP sublayers connected to the residual stream of width D_{model}. Positional encoding and internal attention operations are omitted for clarity.

Alternative structural pruning methods address residual stream transformation constraints. One such method, IntraSlice (Li et al., 2026), performs approximate PCA-based compression but does so within the attention and MLP modules, combining attention-head selection and compression with a separate procedure for nonlinear MLP compression. Its transformations are incorporated into module weights without introducing residual stream adapters. DISP-LLM (Gao et al., 2024) instead enables sublayers to use different subsets of existing residual coordinates. An index selection operation gathers each sublayer's chosen input features, while an index addition operation adds its outputs back at their designated residual coordinates, leaving the remaining coordinates unchanged. These operations avoid dense changes of basis, although indexing overhead remains during inference. DISP-LLM also analyzes the parameter overhead of SliceGPT's residual projections.

Complementary structural pruning. Other structural pruning methods reduce dimensions within attention and MLP modules while leaving the residual stream width unchanged. Týr-the-Pruner (Li et al., 2025) targets these dimensions by combining local pruning and compensatory weight updates with an evolutionary search over layer-wise sparsity distributions. Its iterative procedure alternates between pruning and searching to allocate compression under a global sparsity constraint. Because these targets are distinct from residual stream width, such methods offer a complementary approach to compression, motivating our evaluation of combining Týr-the-Pruner with TrueSlice.

3 METHOD

TrueSlice directly builds upon the computational invariance and matrix slicing framework of Ashkboos et al. (2024), with an implementation derived from revision `6b12cde` of the SliceGPT codebase. Whereas SliceGPT computes different PCA bases across sublayers, TrueSlice computes a single global residual stream subspace from a trace-normalized sum of calibration activation second moments, thereby eliminating SliceGPT's additional residual stream transformations. We support standard transformers, linear attention networks (Katharopoulos et al., 2020), and hybrid networks like Qwen3.5 (QwenTeam, 2026). We additionally support sequential and parallel pre-norm transformers, which differ in normalization fusion and calibration measurement locations but share the same global subspace selection and slicing procedure.

3.1 NORMALIZATION FUSION

We consider pre-norm transformers with L layers and hidden dimension D_{model}. Before calibration, we follow SliceGPT's preprocessing and absorb the learned parameters of residual stream normalization layers into subsequent adjacent linear projections. Given a normalization parameter $\mathbf{G} = \operatorname{diag}(\boldsymbol{\gamma})$ and a projection $\mathbf{W}$, this step replaces $\mathbf{W}$ with $\mathbf{GW}$, and if normalization biases are present, they are absorbed into the projection biases. Under SliceGPT's normalization assumptions, this reparameterization preserves the full-width network function in exact arithmetic. Let $\mathcal{N}$ denote the resulting parameterless RMS normalization. For an orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{D_{\text{model}} \times D_{\text{model}}}$, the normalization identity established by Ashkboos et al. (2024, Eq. (2)) gives

$$\mathcal{N}(\mathbf{XQ}) = \mathcal{N}(\mathbf{X})\mathbf{Q}. \tag{4}$$

For sequential layers, the self-attention and MLP normalization parameters are fused into their respective input projections. For parallel layers with a shared pre-normalization, its parameters are fused into both the self-attention and MLP input projections.

Pre-head normalization. The fusion of the pre-head normalization generally requires untying the LM head from the input embeddings. Let $\mathbf{E}_0 \in \mathbb{R}^{V \times D_{\text{model}}}$ be an originally tied embedding matrix and $\mathbf{G}_f = \operatorname{diag}(\boldsymbol{\gamma}_f)$ the pre-head normalization scale matrix. Initially, for a model using tied embeddings, the head is $\mathbf{H}_0 = \mathbf{E}_0^\top$, but after the normalization fusion, the equivalent head is

$$\mathbf{H} = \mathbf{G}_f \mathbf{E}_0^\top, \qquad \mathbf{H}^\top = \mathbf{E}_0 \mathbf{G}_f. \tag{5}$$

These fused weights generally differ from the input embedding values. Updating the shared LM head would therefore also change the embeddings. SliceGPT's fusion procedure, inherited by TrueSlice, creates an independent copy for the head weights before the fusion and subsequent rotation and slicing. This untying must be included in the total parameter reduction calculations. For TrueSlice, an originally shared matrix with VD_{model} parameters becomes two matrices containing $2VD_{\text{new}}$ parameters in total. In our SliceGPT configurations, the head retains its original input width, giving $V(D_{\text{new}} + D_{\text{model}})$ parameters.

3.2 COMPUTATIONAL INVARIANCE

The computational invariance established by Ashkboos et al. (2024, Theorem 1) provides the theoretical foundation for the rotation and slicing framework that TrueSlice inherits from SliceGPT. It states that an orthogonal rotation of the residual stream coordinates, accompanied by the corresponding weight transformations, exactly preserves the original network function. We restate the result using activation matrices $\mathbf{X} \in \mathbb{R}^{N \times D_{\text{model}}}$, with one token per row, and column-vector biases.

Theorem 1 (Restatement of Ashkboos et al. (2024), Theorem 1)**.** *Consider an RMSNorm-connected transformer satisfying the normalization assumptions described above where the residual stream normalization parameters have already been fused. Let $\mathbf{W}_{\text{in}}^m$ and $\mathbf{W}_{\text{out}}^m$ denote the input and output projections of self-attention or MLP sublayer m, with corresponding biases $\boldsymbol{b}_{\text{in}}^m$ and $\boldsymbol{b}_{\text{out}}^m$, when present. Let $\mathbf{E}$ and $\mathbf{H}$ be the embedding and head matrices. For any orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{D_{\text{model}} \times D_{\text{model}}}$, the transformations*

$$\begin{aligned} \widetilde{\mathbf{E}} &= \mathbf{EQ}, & \widetilde{\mathbf{H}} &= \mathbf{Q}^\top \mathbf{H}, \\ \widetilde{\mathbf{W}}_{\text{in}}^m &= \mathbf{Q}^\top \mathbf{W}_{\text{in}}^m, & \widetilde{\mathbf{W}}_{\text{out}}^m &= \mathbf{W}_{\text{out}}^m \mathbf{Q}, \\ \tilde{\boldsymbol{b}}_{\text{out}}^m &= \mathbf{Q}^\top \boldsymbol{b}_{\text{out}}^m \end{aligned} \tag{6}$$

preserve the network function in exact arithmetic. Input-projection and head biases remain unchanged: $\tilde{\boldsymbol{b}}^m_{\text{in}} = \boldsymbol{b}^m_{\text{in}}$ *and* $\tilde{\boldsymbol{b}}_{\text{head}} = \boldsymbol{b}_{\text{head}}$.

Together with Equation 4, these transformations preserve the network function without requiring explicit rotations between the linear projections and parameterless normalization layers.

3.3 Trace-Normalized Second Moment Accumulation

After normalization parameter fusion, we pass a calibration dataset through the fused, full-width model and measure the activations. For sequential architectures, we measure the input to the first transformer layer and the residual stream states immediately after each attention and MLP residual addition. For parallel architectures with a shared pre-normalization, we measure the input to the first transformer layer and each layer's output after both branch outputs have been added to the residual stream. Thus, the collected measurements represent the accumulated residual stream states before the next normalization.

Denoting $\mathcal{S}$ as the set of measurement locations, we therefore have $|\mathcal{S}| = 2L + 1$ for sequential architectures and $|\mathcal{S}| = L + 1$ for parallel architectures. Let $\mathbf{X}_{s,i} \in \mathbb{R}^{N_i \times D_{\text{model}}}$ be the activations at measurement location s for calibration sequence i, excluding padding positions when masking is enabled. We then accumulate

$$\mathbf{C}_s = \sum_i \mathbf{X}^\top_{s,i} \mathbf{X}_{s,i}. \tag{7}$$

Because the activations are not mean-centered, $\mathbf{C}_s$ is an uncentered scatter matrix whose trace equals the location's total squared activation magnitude: $\operatorname{tr}(\mathbf{C}_s) = \sum_i \|\mathbf{X}_{s,i}\|^2_F$. Direct summation therefore gives greater weight to locations with larger activation magnitudes. TrueSlice removes this difference in scale by dividing each $\mathbf{C}_s$ by its trace during accumulation. Assuming $\operatorname{tr}(C_s) > 0$ at every location, each contribution has unit trace and therefore equal total weight:

$$\mathbf{C}_{\text{glbl}} = \sum_{s \in \mathcal{S}} \frac{\mathbf{C}_s}{\operatorname{tr}(\mathbf{C}_s)}. \tag{8}$$

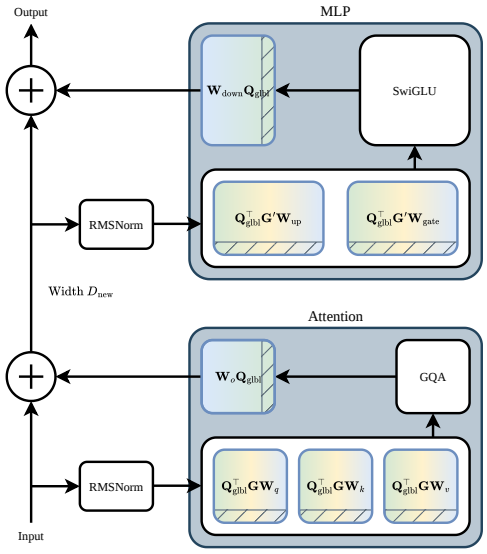

Figure 2: A Llama-style transformer layer after TrueSlice processing (compare Figure 1), adapted from Ashkboos et al. (2024, Figure 4). Normalization scales G and G' are fused into the input projections, and a shared rotation Q_{glbl} is absorbed into the projection weights. Shaded rows and columns are discarded to reduce the residual width to D_{new}. Matrix products are computed during pruning. Thus, the resulting layer requires no additional rotation operations or residual adapters at inference.

We compute the eigendecomposition of $\mathbf{C}_{\text{glbl}}$ and form the global basis $\mathbf{Q}_{\text{glbl}}$ by arranging the resulting eigenvectors as columns in decreasing order of their corresponding eigenvalues.

3.4 Global Subspace Slicing

Following the matrix slicing procedure of Ashkboos et al. (2024, Section 3.4), let $\mathbf{D} \in \mathbb{R}^{D_{\text{model}} \times D_{\text{new}}}$ contain the first D_{new} columns of $\mathbf{I}_{D_{\text{model}}}$. We define the retained global basis as

$$\mathbf{U} = \mathbf{Q}_{\text{glbl}}\mathbf{D}, \qquad \mathbf{U}^\top\mathbf{U} = \mathbf{I}_{D_{\text{new}}}. \tag{9}$$

The columns of $\mathbf{U}$ are the leading D_{new} eigenvectors of $\mathbf{C}_{\text{glbl}}$. For the stacked calibration activations $\mathbf{X}_s$ at measurement location s, the reduced coordinates and their orthogonal reconstruction are

$$\mathbf{Z}_s = \mathbf{X}_s\mathbf{U}, \qquad \widetilde{\mathbf{X}}_s = \mathbf{Z}_s\mathbf{U}^\top = \mathbf{X}_s\mathbf{U}\mathbf{U}^\top. \tag{10}$$

Using the computational invariance stated previously, we absorb the full rotation into the model's projections and retain only the selected coordinates. These operations combine into

$$\begin{aligned} \mathbf{E}^{\text{new}} &= \mathbf{E}\mathbf{U}, & \mathbf{H}^{\text{new}} &= \mathbf{U}^\top \mathbf{H}, \\ \mathbf{W}_{\text{in}}^{m,\text{new}} &= \mathbf{U}^\top \mathbf{W}_{\text{in}}^{m}, & \mathbf{W}_{\text{out}}^{m,\text{new}} &= \mathbf{W}_{\text{out}}^{m}\mathbf{U}, \\ \boldsymbol{b}_{\text{out}}^{m,\text{new}} &= \mathbf{U}^\top \boldsymbol{b}_{\text{out}}^{m}. \end{aligned} \tag{11}$$

Here $\mathbf{E}$ and $\mathbf{H}$ denote the preprocessed embedding and head matrices, and m indexes the attention and MLP sublayers, as in the preceding theorem. The input projection and head biases remain unchanged. The full rotation preserves the network function in exact arithmetic under the stated invariance assumptions, while truncation introduces an approximation. After slicing, we restore native RMSNorm layers at the reduced width, adjusting their effective scales and stabilizers to preserve the normalization computed using the original width.

Figure 2 shows how the compressed model operates directly in the reduced coordinates, so $\widetilde{\mathbf{X}}_s$ describes the orthogonal reconstruction rather than an explicit operation performed during inference. This reconstruction analysis concerns fixed calibration activations and does not guarantee optimal language model loss after pruning. Because every sublayer uses the same retained basis, residual connections remain ordinary additions, requiring no additional residual stream rotation parameters or operations at inference.

4 Experimental Results

Setup. Pruning targets denote reductions in total parameter count relative to the dense model, rather than reductions in the residual stream width. We select the retained width separately for each method to best match these targets. Across the evaluated combinations, the achieved parameter reductions meet the targets or fall short by less than 0.1 percentage points. In hardware benchmarks, where widths are rounded to multiples of eight, achieved reductions are within 0.4 percentage points of the target. We additionally guarantee that SliceGPT always retains at least as many parameters as TrueSlice for any target-model pair. Appendix A.1 details the input sparsity values and achieved total-parameter reductions for each method, model, and target. For all experiments, we use TrueSlice revision `55b5acc`, which includes minimal changes to SliceGPT to support newer families of models and versions of Hugging Face Transformers (Wolf et al., 2020). Unless otherwise stated, both methods use the dense model's native data type, double-precision second-moment calculations, dataset sampling seed 42, and 128 WikiText-2 (Merity et al., 2016) calibration samples with a maximum sequence length of 2,048 tokens. Calibration and evaluation use disjoint training and test splits of the same dataset.

4.1 Role of Trace-Normalization

We isolate the effect of trace normalization (Section 3) by comparing two accumulation rules within TrueSlice, while holding all other settings fixed. The unnormalized variant directly sums calibration activation scatter matrices across measurement locations, whereas the normalized variant divides each matrix by its trace before summation. Figure 3 shows that these traces span several orders of magnitude, with broadly similar profiles for WikiText-2 and Alpaca (Taori et al., 2023) within each model. Later locations generally have larger traces and thus contribute more to the unnormalized sum. Trace normalization gives each location a unit-trace contribution, removing this scale imbalance while preserving each matrix's principal directions and eigenvalue ratios.

Table 1 compares perplexity across three models and three target parameter reductions, using WikiText-2 (W2) or Alpaca (Al) for calibration and evaluation on disjoint splits. Trace normalization lowers perplexity in 17 of 18 comparisons, with the sole exception being Qwen3-14B on Alpaca at 10% reduction, where perplexity increases by 0.01. Benefits are generally larger at higher compression, although they do not increase monotonically in every model-dataset pair. At 30% reduction on WikiText-2, normalization lowers perplexity from 47.00 to 34.69 for Qwen3.5-4B, from 27.69 to 18.82 for Llama-3.1-8B, and from 22.56 to 15.67 for Qwen3-14B. The Qwen3.5-4B results also support its effectiveness in a hybrid attention model incorporating Gated DeltaNet layers.

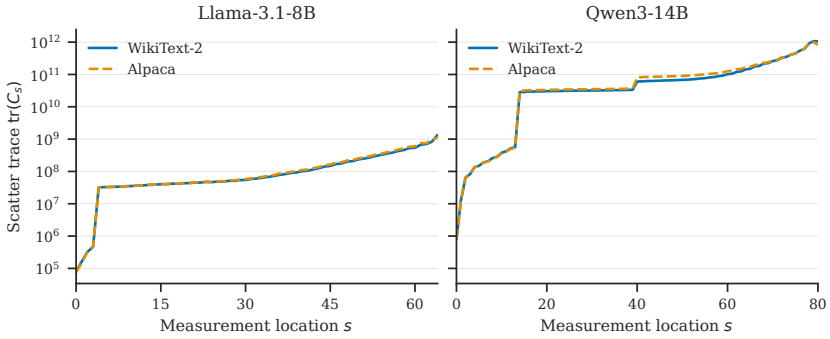

Figure 3: Calibration activation scatter traces across measurement locations in Llama-3.1-8B (left) and Qwen3-14B (right). Measurement locations are defined in Section 3.

Table 1: Effect of trace normalization on perplexity after TrueSlice. W2 and Al denote WikiText-2 and Alpaca, respectively, used for calibration and evaluation on disjoint splits. Lower is better.

Method		Qwen3.5-4B		Llama-3.1-8B		Qwen3-14B	
		W2	Al	W2	Al	W2	Al
Dense		9.58	4.60	6.29	3.25	8.65	3.73
Unnormalized	10%	15.51	9.86	7.87	3.47	9.73	3.86
	20%	25.44	14.51	12.33	4.41	11.94	4.21
	30%	47.00	19.49	27.69	7.15	22.56	5.40
Normalized	10%	14.27	7.44	7.56	3.38	9.65	3.87
	20%	21.18	10.72	10.65	3.97	11.41	4.18
	30%	34.69	16.64	18.82	5.47	15.67	5.21

4.2 Perplexity Across Models

Table 2 compares TrueSlice and SliceGPT with no subsequent recovery training across eight Llama 3 and Qwen3 models, using WikiText-2 training and test splits for calibration and perplexity evaluation, respectively. At matched target parameter reductions, TrueSlice achieves lower perplexity in 17 of the 24 comparisons: all eight at 10%, seven at 20%, and two at 30%, which we investigate in Appendix A.2. Cross-family comparisons warrant caution because we evaluate Llama 3 base models and Qwen3 instruct variants.

Both methods substantially increase perplexity for the models with tied embeddings: Llama-3.2-1B, Llama-3.2-3B, and Qwen3-4B. During normalization fusion, both methods untie the embedding and LM head weights, adding approximately 263M, 394M, and 389M parameters to the respective models before slicing. Matching a target reduction relative to the original dense model therefore requires removing these additional parameters, necessitating greater residual width reduction as shown in Appendix A.1. Although this cost is measurable, these experiments do not isolate the causes of perplexity degradation or differences between methods. Attributing them to tied embeddings, Q/K normalization, or model redundancy requires controlled experiments.

4.3 Hardware Performance

We measure the pruning compute cost and inference performance at a 30% target parameter reduction on a single NVIDIA H100 SXM 80 GB GPU hosted on RunPod using BF16 precision. Both methods use 128 calibration samples with a maximum sequence length of 2,048 tokens. Inference benchmarking consists of 128 sequences with 128-token prompts and 128 decode steps at batch size one, with five repetitions and five untimed warmup steps before each sequence in each repetition. To reflect a realistic deployment, compressed hidden dimensions are rounded up to multiples of eight, but remain unaligned to multiples of 128. This could potentially favor the dense baseline through

Table 2: WikiText-2 test perplexity for dense models and models pruned without recovery training. Lower is better.

Method		Llama-3.2		Llama-3.1		Qwen3			
		1B	3B	8B	70B	4B	8B	14B	32B
Dense		9.75	7.81	6.29	2.81	13.66	9.72	8.65	7.61
SliceGPT	10%	30.71	19.31	11.29	7.02	18.38	11.78	10.40	8.68
	20%	44.40	26.99	18.24	8.80	23.07	14.70	11.70	9.75
	30%	69.39	38.76	33.43	11.92	34.30	19.93	14.61	11.77
TrueSlice	10%	24.89	13.00	7.56	4.82	17.37	10.57	9.65	8.22
	20%	42.21	21.47	10.65	6.82	25.31	12.96	11.41	9.49
	30%	74.70	38.88	18.82	10.74	51.14	20.14	15.67	12.06

Table 3: Pruning cost and inference performance at a 30% target parameter reduction on an NVIDIA H100 SXM 80 GB GPU in BF16. Latency is per cached token-forward step. TTFT is time to first token. Higher throughput and lower times are better.

Method	Model	Pruning (s)	Throughput (t/s)	Latency (ms)	TTFT (ms)
Dense	Llama-3.2-1B	0	161.035	6.210	6.582
	Llama-3.1-8B	0	83.882	11.922	13.019
	Qwen3-32B	0	28.702	34.841	43.215
SliceGPT	Llama-3.2-1B	171	160.895	6.215	6.628
	Llama-3.1-8B	498	84.132	11.886	12.422
	Qwen3-32B	1689	32.219	31.047	36.453
TrueSlice	Llama-3.2-1B	197	164.352	6.085	6.500
	Llama-3.1-8B	614	85.296	11.724	12.114
	Qwen3-32B	1511	33.317	30.015	36.063

hardware or kernel optimizations. We retain a nominal 30% reduction target and match the methods to within 16M parameters for each model, prioritizing comparable parameter budgets over isolating the effects of dimension alignment.

Table 3 details pruning time, including normalization fusion, rotation, slicing, and any restoration steps, alongside inference performance. Under this batch-size-one configuration, TrueSlice improves cached decoding throughput over SliceGPT by 1.4–3.4%, with lower per-token latency and TTFT across all three models. Pruning takes longer for TrueSlice on the two Llama models but less time on Qwen3-32B. These benchmarks evaluate inference at a fixed precision and batch size, but they do not measure peak memory consumption, maximum batched throughput, or performance relative to quantized models.

4.4 Zero-Shot Accuracy

We use the LM Evaluation Harness (Gao et al., 2021) included in the SliceGPT codebase for all zero-shot task evaluations. Following Ashkboos et al. (2024), we benchmark the methods on PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2020), HellaSwag (Zellers et al., 2019), and ARC-e and ARC-c (Clark et al., 2018).

Figure 4 compares SliceGPT and TrueSlice on Llama-3.1-8B at a matched 30% target parameter reduction using only WikiText-2. Using SliceGPT's LoRA-based fine-tuning procedure (Hu et al., 2022), we test models before and after 3B tokens of recovery training. Hardware constraints limit recovery training to this budget with performance at larger budgets remaining untested. Both methods use the same perplexity evaluation procedure, which differs from earlier experiments, preventing direct comparison with previously reported perplexities. Recovery increases average accuracy from 50.9% to 58.0% for TrueSlice and from 48.5% to 54.4% for SliceGPT. After recovery, TrueSlice outperforms SliceGPT on all five tasks and achieves lower perplexity (14.63 versus 17.04).

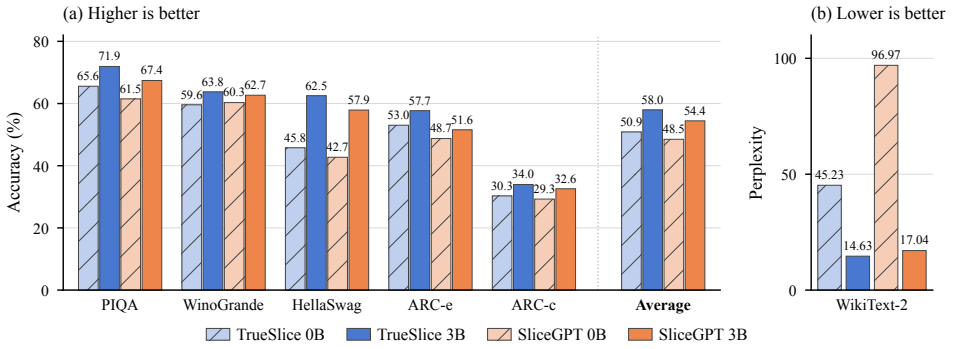

Figure 4: Zero-shot accuracy (%) on Llama-3.1-8B at a 30% target parameter reduction and perplexity before (0B) and after (3B) recovery. Average is the unweighted mean across the five benchmarks. Higher accuracy and lower perplexity are better.

Table 4: Zero-shot accuracy (%) on Llama-3.1-8B after combined pruning. Labels $a\%$ / $b\%$ allocate percentage points of total-parameter reduction to Týr-the-Pruner / TrueSlice with $b = 0$ denoting Týr alone. Avg. is the unweighted mean across the five benchmarks. Higher is better.

Benchmark	**25% Total Cut**		**50% Total Cut**			
	25% / 0%	15% / 10%	50% / 0%	40% / 10%	30% / 20%	20% / 30%
PIQA	76.06	74.65	63.71	64.80	64.64	60.45
WinoGrande	68.67	69.61	52.64	55.09	54.93	53.59
HellaSwag	68.60	66.60	41.12	42.31	41.12	36.06
ARC-e	73.86	74.45	42.59	43.35	51.64	45.45
ARC-c	44.97	44.45	24.23	26.96	27.73	26.88
Avg.	66.43	65.95	44.86	46.50	48.01	44.49

4.5 Combining TrueSlice with Týr-the-Pruner

We evaluate TrueSlice followed by Týr-the-Pruner on Llama-3.1-8B without recovery training, combining residual width reduction with attention-head and MLP-feature pruning. We match a given target parameter reduction r by adjusting Týr's input sparsity to $1-(1-r)P_0/P_{\mathrm{TS}}$ for dense parameter count P_0 and post-TrueSlice count P_{TS}. Table 4 compares allocations of the total parameter reduction between the two methods, relative to the original dense parameter count. At the 25% target, the combined 15% / 10% allocation achieves an average accuracy of 65.95%, compared with 66.43% for Týr-the-Pruner alone. At the 50% target, the 40% / 10% allocation improves all five benchmarks, raising average accuracy from 44.86% to 46.50%. The 30% / 20% allocation achieves the highest average among the tested configurations, at 48.01%, improving four benchmarks and matching Týr-the-Pruner alone on HellaSwag. Increasing TrueSlice's contribution to 30 percentage points reduces average accuracy to 44.49%. Selected combinations therefore outperform Týr-the-Pruner alone at the larger reduction, although the benefit depends on the allocation.

5 Conclusion

We presented TrueSlice, which uses trace-normalized calibration second moments to select a shared residual subspace, reducing width while preserving the dense architectural structure. Experiments on standard and hybrid attention models demonstrate the benefits of trace normalization. TrueSlice achieves lower perplexity than SliceGPT in most matched parameter reduction comparisons, improves inference performance, and yields better task performance after recovery training at a matched 3B-token budget. Selected combinations with Týr-the-Pruner also demonstrate the benefits of combining residual width reduction with attention-head and MLP-feature pruning. Future work could explore other complementary pruning methods and compression allocations.

AI USE STATEMENT

We used generative AI to assist in formalizing mathematical claims, writing proofs, and providing feedback on benchmarking methodology and experimental setup. We also used it to debug code, adapt SliceGPT benchmarking functions, condense and refine author-written text, produce plots from experiment results, identify and summarize missed related literature, and format references. We did not use generative AI to generate synthetic data, develop theoretical models or conceptual frameworks, provide critical arguments for proofs, propose or refine hypotheses, translate text, clean datasets, or support qualitative or thematic analysis. We used generative AI to provide feedback on our interpretation and presentation of experimental results, but all interpretations and conclusions reported in the paper are our own.

We reviewed all AI-assisted work. More precisely, we manually revised AI-assisted text and mathematical content for clarity and correctness, confirmed all code changes before integration, checked produced figures against the underlying data, and reviewed all identified literature. We take responsibility for the final content of this work, including all AI-assisted text, mathematical claims, proofs, code, and figures.

REPRODUCIBILITY STATEMENT

Although we did not retain the pruned model checkpoints, we have submitted the complete code for TrueSlice and our updated implementation of SliceGPT. We have also included much of the code used to run the experiments reported in this paper.

ACKNOWLEDGMENTS

We are deeply grateful to Anonymous for the substantial time, effort, and technical expertise he devoted to running and accelerating many of the large-scale experiments reported in this paper. His generous support and tireless work were essential to completing this research.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, 2023. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298/.

Saleh Ashkboos, Maximilian Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations*, volume 2024, pp. 11682–11701, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/316648eb8b4ffb6010f531b07848c300-Abstract-Conference.html.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34(05), pp. 7432–7439, 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

Haoxian Chen and Limin Wang. RotPruner: Large language model pruning in rotated space. Submission to ICLR 2025, 2025. URL https://openreview.net/forum?id=wV9iMiyQcc. Submission number 7292.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. doi: 10.48550/arXiv.1803.05457. URL https://arxiv.org/abs/1803.05457.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffery Hsu, Kyle McDonell, Niklas Muennighoff, et al. A framework for few-shot language model evaluation, 2021. URL https://github.com/eleutherai/lm-evaluation-harness. Version v0.0.1. Sept, 2021.

Shangqian Gao, Chi-Heng Lin, Ting Hua, Tang Zheng, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. DISP-LLM: Dimension-independent structural pruning for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 72219–72244, 2024. doi: 10.52202/079017-2305. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/84a7fc24ed52e8eff514c33e8ac76ea3-Abstract-Conference.html.

Aaron Grattafiori et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. doi: 10.48550/arXiv.2407.21783. URL https://arxiv.org/abs/2407.21783.

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

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 5156–5165, 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

Guanchen Li, Yixing Xu, Zeping Li, Ji Liu, Xuanwu Yin, Dong Li, and Emad Barsoum. Týr-the-Pruner: Structural pruning LLMs via global sparsity distribution optimization. In *Advances in Neural Information Processing Systems*, volume 38, pp. 149163–149192, 2025. doi: 10.52202/085713-4989. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/db2cbf43a349bc866111e791b58c7bf4-Abstract-Conference.html.

Meng Li, Peisong Wang, Yuantian Shao, Qinghao Hu, Hongjian Fang, Yifan Zhang, Zhihui Wei, and Jian Cheng. IntraSlice: Towards high-performance structural pruning with block-intra PCA for LLMs. *arXiv preprint arXiv:2602.01975*, 2026. doi: 10.48550/arXiv.2602.01975. URL https://arxiv.org/abs/2602.01975.

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 *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 39968–39986. PMLR, 2025. URL https://proceedings.mlr.press/v267/liu25cj.html.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016. doi: 10.48550/arXiv.1609.07843. URL https://arxiv.org/abs/1609.07843.

Ofir Press and Lior Wolf. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 157–163. Association for Computational Linguistics, 2017. URL https://aclanthology.org/E17-2025/.

QwenTeam. Qwen3.5-4B. https://huggingface.co/Qwen/Qwen3.5-4B, 2026.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34(05), pp. 8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399. URL https://ojs.aaai.org/index.php/AAAI/article/view/6399.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. doi: 10.48550/arXiv.2002.05202. URL https://arxiv.org/abs/2002.05202.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpaca: A strong, replicable instruction-following model. https://crfm.stanford.edu/2023/03/13/alpaca.html, 2023. Stanford Center for Research on Foundation Models blog post.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6/.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10524–10533. PMLR, 2020. URL https://proceedings.mlr.press/v119/xiong20b.html.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a. doi: 10.48550/arXiv.2505.09388. URL https://arxiv.org/abs/2505.09388.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, volume 2025, pp. 29687–29707, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/4904fad153f6434a7bcf04465d4be2cc-Abstract-Conference.html.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800. Association for Computational Linguistics, 2019. URL https://aclanthology.org/P19-1472/.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

A Appendix

A.1 Exact Sparsity Values

Table 5 reports the input sparsity values and achieved total parameter reductions for each model and method. Input sparsity denotes the residual width reduction fraction supplied to the method, whereas total parameter reductions are measured relative to the original dense parameter counts shown in parentheses. All achieved reductions are within 0.1 percentage points of their targets, with SliceGPT retaining at least as many parameters as TrueSlice in every comparison.

A.2 Model Diagnostics

To investigate the changing relative performance between TrueSlice and SliceGPT, we compare the relative squared reconstruction error $\|\mathbf{X}_s - \mathbf{X}_s\mathbf{U}_s\mathbf{U}_s^\top\|_F^2/\|\mathbf{X}_s\|_F^2$ at each residual-stream location s for both methods. Here $\mathbf{X}_s$ represents the dense model's calibration activations, and $\mathbf{U}_s$ is the retained basis selected by the corresponding method. Both methods are evaluated against the dense activations, using their respective retained widths at the matched parameter reductions. Figure 5

Table 5: Input sparsity values and achieved total parameter reductions for TrueSlice and SliceGPT.

Model	Target	TrueSlice		SliceGPT	
	Cut	Input	Actual	Input	Actual
Llama-3.2-1B	10%	0.257325	9.9471%	0.361817	9.9376%
(1,235,814,400)	20%	0.339844	19.9530%	0.451661	19.9516%
	30%	0.422364	29.9588%	0.542969	29.9548%
Llama-3.2-3B	10%	0.198243	9.9917%	0.304688	9.9552%
(3,212,749,824)	20%	0.287110	19.9683%	0.387370	19.9297%
	30%	0.376303	29.9814%	0.472331	29.9489%
Llama-3.1-8B	10%	0.099854	9.9854%	0.200196	9.9793%
(8,030,261,248)	20%	0.199952	19.9951%	0.288819	19.9778%
	30%	0.299805	29.9805%	0.379151	29.9563%
Llama-3.1-70B	10%	0.099976	9.9976%	0.200928	9.9954%
(70,553,706,496)	20%	0.199952	19.9951%	0.283326	19.9816%
	30%	0.299927	29.9927%	0.367676	29.9917%
Qwen3-4B	10%	0.179297	9.9938%	0.262110	9.9638%
(4,022,468,096)	20%	0.270313	19.9754%	0.348438	19.9444%
	30%	0.361719	29.9999%	0.436719	29.9724%
Qwen3-8B	10%	0.099854	9.9853%	0.209473	9.9795%
(8,190,735,360)	20%	0.199952	19.9951%	0.297364	19.9730%
	30%	0.299805	29.9804%	0.387208	29.9563%
Qwen3-14B	10%	0.100000	10.0000%	0.202344	9.9849%
(14,768,307,200)	20%	0.200000	20.0000%	0.289063	19.9974%
	30%	0.300001	30.0000%	0.377540	29.9954%
Qwen3-32B	10%	0.100000	10.0000%	0.174610	9.9892%
(32,762,123,264)	20%	0.200000	20.0000%	0.262891	19.9809%
	30%	0.300001	30.0000%	0.352735	29.9869%
Qwen3.5-4B	10%	0.217969	9.9724%	N/A	N/A
(4,205,751,296)	20%	0.305079	19.9983%	N/A	N/A
	30%	0.391797	29.9793%	N/A	N/A

shows that, for Qwen3-8B, TrueSlice has slightly lower mean error at 10% reduction, but higher error at 20% and 30%. Thus, at higher compression levels, SliceGPT's separate subspaces reconstruct these dense activations more accurately despite retaining fewer residual dimensions at all locations except the final output. This indicates an increasing cost of TrueSlice's global subspace constraint.

Figure 6 shows how these differences vary across measurement locations. For Qwen3-8B, TrueSlice generally incurs larger reconstruction errors at the embedding output and early sublayers, while the later sublayers are broadly comparable to SliceGPT. Llama-3.1-8B exhibits a more pronounced contrast: TrueSlice has considerably larger errors at the embeddings and earliest sublayers, but smaller errors in later layers. These opposing differences partially cancel in the mean. SliceGPT's final reconstruction error is zero because it retains the full residual width at the final output, whereas TrueSlice also slices this location. This endpoint therefore reflects the compression configuration rather than recovery of the dense model's predictions.

The results support a tradeoff between retained width and subspace flexibility, but do not establish a complete explanation for the perplexity reversal observed in Table 2. In particular, SliceGPT has lower mean reconstruction error on Qwen3-8B at 20%, where TrueSlice has lower perplexity. Similarly, SliceGPT has slightly lower mean error on Llama-3.1-8B at 30%, despite TrueSlice's substantial perplexity advantage. Reconstruction error measures the squared magnitude of discarded activation components rather than their importance for prediction. The growing reconstruction disadvantage on Qwen3-8B is therefore consistent with TrueSlice's diminishing and ultimately reversed perplexity advantage. That being said, the discrepancies demonstrate the limits of this diagnostic.

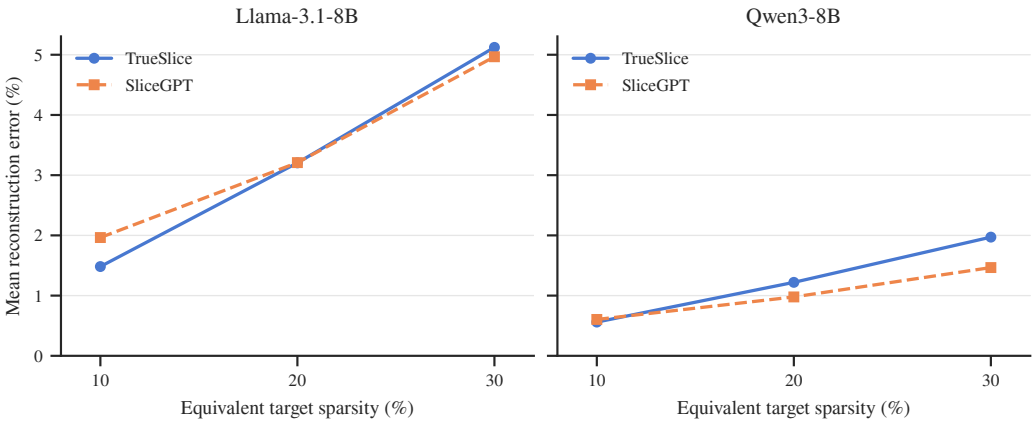

Figure 5: Mean relative squared reconstruction error weighting all measurement locations equally in Llama-3.1-8B (left) and Qwen3-8B (right).

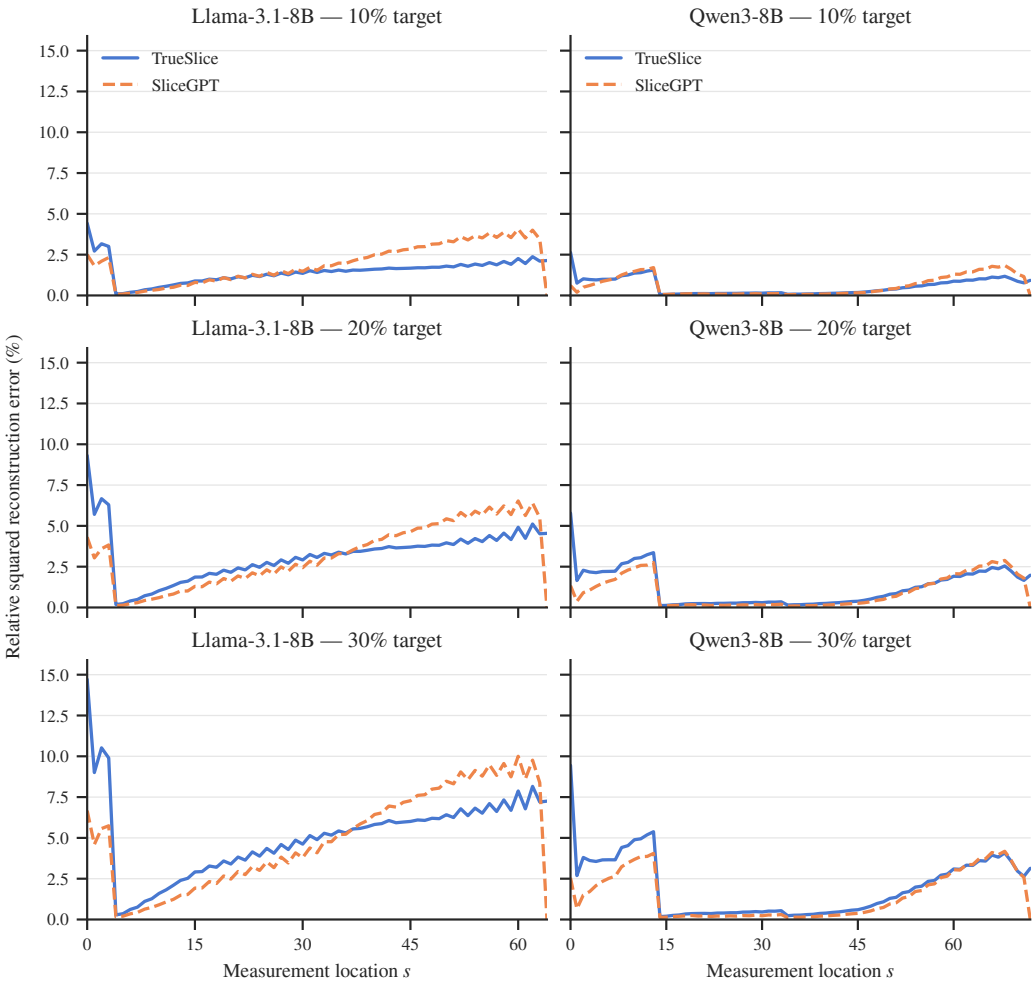

Figure 6: Relative squared reconstruction error across measurement locations in Llama-3.1-8B (left) and Qwen3-8B (right). Measurement locations are defined in Section 3.