WUSPRUNER: SECOND-MOMENT-OPTIMAL TRANSFORMS FOR STRUCTURED PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Structured pruning is typically formulated as identifying which model dimensions to remove, often within the model's original representation. However, the effect of coordinate deletion fundamentally depends on the basis in which pruning is performed. This raises a more fundamental question: *how should the basis be chosen specifically for coordinate deletion?* We introduce *WUSPruner*, a transform-based framework for structured pruning of large language models. We formulate basis selection as a local linear reconstruction problem that directly minimizes the reconstruction loss induced by pruning. Inspired by the second-moment transformation in WUSH (Chen et al., 2026), we consider a WUS transform constructed jointly from activation and weight statistics. We show that under positive-definite second moments, WUS transform minimizes the local reconstruction loss over all invertible changes of basis. In the resulting basis, the activation and weight second moments are simultaneously diagonalized and balanced, causing the deletion error to decompose into independent spectral contributions and yielding a principled criterion for coordinate selection. Guided by this result, we develop a scalable pruning method based on blockwise WUS transformations and compact input projections. Experiments on Llama and Qwen models from 8B to 72B parameters show consistent improvements over structured pruning baselines across language modeling, knowledge, commonsense reasoning, mathematics, and code generation benchmarks, while also delivering practical reductions in inference latency and memory usage.

1 INTRODUCTION

Large language models achieve strong performance at substantial computational and memory cost, motivating compression techniques that reduce the cost of inference. Among them, structured pruning (Sun et al., 2024; Ashkboos et al., 2024; Tang et al., 2026; 2025) is particularly attractive because it removes entire dimensions or channels, yielding smaller dense operators that can translate into practical reductions in latency and memory usage regardless of the hardware specification. However, structured pruning is also inherently destructive: removing a representation dimension eliminates its contribution to the computation rather than merely zeroing individual parameters without constraint. As a result, even moderate pruning can substantially distort the outputs of the original model. The central challenge is therefore to remove model dimensions while preserving the performance of the original dense model as much as possible. In this work, we study this problem through the local reconstruction of linear mappings, where the intermediate dimension is reduced while minimizing the resulting output distortion.

A key observation is that structured pruning is inherently basis-dependent. Many existing pruning heuristics operate directly in the model's native representation, assigning importance scores to weights or dimensions and removing those deemed least important. Methods such as Wanda (Sun et al., 2024) perform pruning in the model's original representation, whereas transform-based approaches such as SliceGPT (Ashkboos et al., 2024) modify the basis before deletion. This highlights that pruning quality depends not only on coordinate importance, but also on the choice of representation basis. However, the PCA-based transformation of SliceGPT is chosen according to activation statistics rather than the reconstruction error induced by pruning itself. This leaves a fundamental question open: *how should the basis be chosen specifically for coordinate deletion?* Our goal is

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.

therefore to identify a transformation that minimizes the reconstruction loss induced by coordinate deletion under a fixed pruning budget.

In this work, we formalize this objective as a local linear reconstruction problem. Given calibration activations and the corresponding weight matrix, we optimize over invertible changes of basis to minimize the reconstruction loss after deleting a fixed number of transformed coordinates. We show that this objective admits a theoretically optimal solution given by a WUS transformation inspired by the second-moment construction of WUSH (Chen et al., 2026). The transform is constructed jointly from activation and weight statistics and is obtained by removing the Hadamard component used for quantization. Under positive-definite activation and weight second moments, WUS attains the minimum reconstruction loss over all invertible transformations. In the WUS basis, the transformed activation and weight second moments are simultaneously diagonalized and balanced, eliminating cross-coordinate interactions in the deletion error. As a result, the reconstruction loss decomposes into independent coordinate-wise contributions determined by a singular-value spectrum, providing a principled criterion for coordinate selection. This characterization further connects transformed coordinate deletion to optimal rank-constrained approximation.

The theoretically optimal construction, however, is not directly suitable for large-scale Transformer pruning. Applying a full transformation to high-dimensional representations incurs substantial construction and storage costs, and practical deployment requires the transformed representation to integrate efficiently with existing Transformer modules. To bridge this gap, we develop *WUSPruner*, a scalable approximation to the unrestricted WUS construction. We partition each input space into blocks and compute the WUS transformation locally within each block, substantially reducing the cost of constructing the pruning basis. The retained transformed coordinates are then implemented through compact input projections followed by smaller dense linear maps. Together, these designs make the theoretically motivated basis construction practical for large-scale Transformer pruning while keeping the additional memory and computational overhead manageable.

Empirically, *WUSPruner* consistently improves structured pruning quality across model families and scales. We evaluate models ranging from 8B to 72B parameters on language modeling, commonsense reasoning, knowledge, mathematics, and code generation benchmarks. Across the evaluated models, *WUSPruner* achieves higher average downstream accuracy than the structured pruning baselines and consistently improves perplexity on PTB and C4. The gains persist when scaling from 8B models to Qwen3.8-27B and Qwen2.5-72B, indicating that the proposed basis construction remains effective at larger model scales. Beyond pruning quality, *WUSPruner* also translates structural compression into practical efficiency gains, including both inference latency and memory consumption reduction. In summary, our main contributions are:

- We formulate structured pruning as a *basis-selection problem*: instead of pruning coordinates only in the model's original representation, we optimize the change of basis with respect to the reconstruction loss induced by coordinate deletion.
- We provide a theoretical characterization of the optimal basis for this objective. Under positive-definite activation and weight second moments, we show that the WUS transformation minimizes the local reconstruction loss over all invertible transformations. In this basis, the deletion error decomposes into independent spectral contributions, connecting coordinate pruning to optimal rank-constrained approximation.
- Building on this result, we develop *WUSPruner*, a scalable structured pruning method based on blockwise WUS transformations and compact shared input projections. Experiments on Llama and Qwen models from 8B to 72B parameters show consistent improvements over structured pruning baselines across language modeling and downstream tasks, together with practical reductions in inference latency and memory usage.

2 Related Work

Structured and transform-based pruning. Post-training pruning methods typically use calibration data or task-derived signals to estimate parameter importance and remove redundant model components. SparseGPT (Frantar & Alistarh, 2023) enables one-shot unstructured and semi-structured sparsification through approximate second-order reconstruction, while Wanda (Sun et al., 2024) combines weight magnitudes with input activation norms to construct data-aware pruning scores.

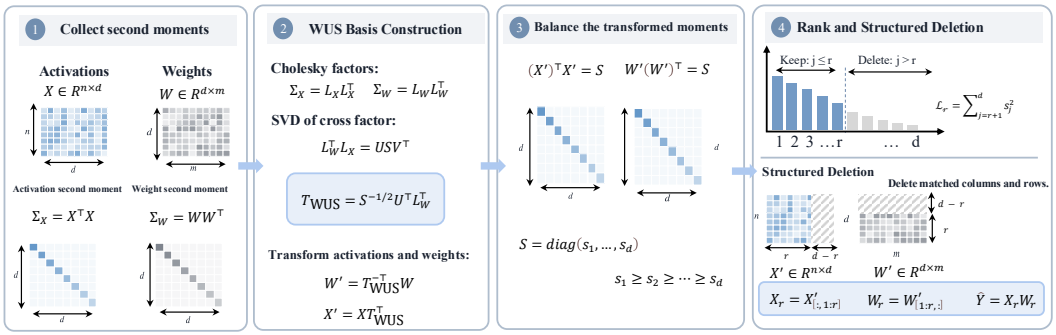


Figure 1: Overview of *WUSPruner*. WUS constructs a basis from activation and weight second moments such that the transformed moments are balanced and diagonalized with a shared spectrum S. This makes the pruning loss decompose into independent coordinate-wise contributions, allowing transformed coordinates to be ranked by s_i and the least important matched dimensions to be structurally removed.

Structured pruning instead removes entire coupled components to obtain hardware-friendly compression. LLM-Pruner (Ma et al., 2023) identifies dependent structures using gradient-based importance and restores performance through low-rank adaptation, ZipLM (Kurtić et al., 2023) explicitly optimizes the trade-off between predictive loss and runtime on a target inference platform, and FLAP (An et al., 2024) combines activation fluctuation statistics with adaptive structure selection and bias compensation. A complementary line of work studies whether pruning becomes easier after changing the representation basis. SliceGPT (Ashkboos et al., 2024), for instance, exploits orthogonal invariances in Transformers to rotate hidden representations using activation-derived PCA bases and subsequently remove low-importance dimensions. Our proposed *WUSPruner* derives the basis from a different objective: *minimizing the local output reconstruction error induced by coordinate deletion while jointly accounting for the second moments of both activations and weights.* To translate this principle into a practical structured compressor, our *WUSPruner* employs local blockwise projections and reduced linear maps while retaining the original residual-stream dimensionality, and explicitly includes the projection operators in the compressed-model cost.

Activation-aware low-rank compression. Low-rank compression (Hu et al., 2021; Dettmers et al., 2023; Chavan et al., 2023; Zhao et al., 2024; Hayou et al., 2024; Tian et al., 2024) approximates a dense linear operator by the product of smaller matrices, reducing both parameter count and computation. Classical SVD provides the optimal approximation under an unweighted Frobenius objective, whereas recent LLM compression methods incorporate activation statistics to better reflect the actual output distortion induced by compression. ASVD (Yuan et al., 2023) rescales weight matrices according to activation distributions before decomposition, mitigating the effect of activation outliers and adapting compression to layer sensitivity. SVD-LLM (Wang et al., 2025), for instance, introduces truncation-aware data whitening to connect singular-value truncation with calibration-set reconstruction loss and further performs sequential updates of compressed weights. This viewpoint is closely related to our formulation. After an appropriate change of basis, retaining only r coordinates produces an operator of rank at most r. Under the unrestricted local formulation considered in this work, the optimal coordinate-deletion solution achieves the same minimum output reconstruction error as the best activation-weighted rank-r approximation. This yields a spectral characterization that connects structured coordinate pruning with activation-aware low-rank compression. Our *WUSPruner* translates the unrestricted theoretical construction into a practical blockwise representation with shared input projections and spectral coordinate selection.

3 METHOD

We introduce *WUSPruner*, a transform-based framework for structured pruning. We first formulate pruning as coordinate deletion under an invertible change of basis and characterize the reconstruction error induced by deletion. We then show that the WUS transformation attains the minimum local reconstruction error over unrestricted invertible transformations under positive-definite second

moments, yielding a spectral criterion for coordinate selection. Finally, we develop a scalable block-wise approximation for Transformer LLMs. The overview of *WUSPruner* is shown in Figure 1.

3.1 Transformed Coordinate Deletion

Consider a linear map $Y = XW$, where $X \in \mathbb{R}^{n\times d}$ contains calibration activations and $W \in \mathbb{R}^{d\times m}$ is the weight matrix. Our objective is to reduce the intermediate dimension from d to $r = d - k$ while preserving the output of the original map. For any invertible transformation $T \in \mathrm{GL}(d)$, the activations and weights can be reparameterized without changing the original computation: $XW = (XT^\top)(T^{-\top}W)$. Let $P \in \mathbb{R}^{d\times d}$ be a diagonal projector onto the k coordinates to be deleted. After coordinate deletion, the resulting approximation is $\widehat{Y}(T) = XT^\top(I - P)T^{-\top}W$.

Although the transformation leaves the computation unchanged before pruning, the approximation after deletion depends on the chosen basis. We therefore consider the local reconstruction objective

$$\mathcal{L}(T) = \|Y - \widehat{Y}(T)\|_F^2 = \|XT^\top PT^{-\top}W\|_F^2. \tag{1}$$

Without loss of generality, we take P to select the last k coordinates, since any other selection can be represented by a permutation of the transformed basis.

Define the unnormalized second moments $\Sigma_X = X^\top X$ and $\Sigma_W = WW^\top$. By cyclic invariance of the trace, equation 1 becomes

$$\mathcal{L}(T) = \mathrm{Tr}(P\, M_X(T)\, P\, M_W(T))\,, \tag{2}$$

where $M_X(T) = T\Sigma_X T^\top$ and $M_W(T) = T^{-\top}\Sigma_W T^{-1}$. Thus, under this local objective, the effect of coordinate deletion depends jointly on the transformed activation and weight second moments. We do not center the activations, since the objective concerns reconstruction of the original linear outputs; a shared bias, if present, cancels from the reconstruction error.

To make the role of the basis explicit, let $\widetilde{X} = XT^\top$ and $\widetilde{W} = T^{-\top}W$. For a deleted coordinate set $\mathcal{D}$, the removed contribution is $Y - \widehat{Y} = \sum_{i\in\mathcal{D}} \widetilde{X}_{:,i}\widetilde{W}_{i,:}$. Its squared reconstruction error can be written as

$$\mathcal{L}(T;\mathcal{D}) = \sum_{i\in\mathcal{D}} [M_X(T)]_{ii}[M_W(T)]_{ii} + 2\sum_{\substack{i,j\in\mathcal{D}\\ i<j}} [M_X(T)]_{ij}[M_W(T)]_{ij}. \tag{3}$$

The first term represents individual coordinate contributions, while the second captures interactions between deleted coordinates. A basis that removes these interactions turns coordinate selection into independent reconstruction costs.

3.2 Optimal Local Deletion with WUS

We next characterize a transformation that minimizes equation 1. Assume $\Sigma_X \succ 0$ and $\Sigma_W \succ 0$, with Cholesky factorizations $\Sigma_X = L_X L_X^\top$ and $\Sigma_W = L_W L_W^\top$. Consider the singular value decomposition $L_W^\top L_X = USV^\top$, where $S = \mathrm{diag}(s_1, \ldots, s_d)$ and $s_1 \geq \cdots \geq s_d > 0$.

Inspired by the second-moment transformation in WUSH (Chen et al., 2026), we define the WUS transformation:

$$T_{\mathrm{WUS}} = S^{-1/2}U^\top L_W^\top. \tag{4}$$

Its key property is that it simultaneously diagonalizes and balances the two second moments:

$$M_X(T_{\mathrm{WUS}}) = M_W(T_{\mathrm{WUS}}) = S. \tag{5}$$

Indeed, $T_{\mathrm{WUS}}^{-\top}W = S^{1/2}U^\top L_W^{-1}W$, and $L_W^{-1}\Sigma_W L_W^{-\top} = I$ gives the transformed weight second moment S. On the activation side, $L_W^\top \Sigma_X L_W = US^2U^\top$, which yields the same transformed second moment S.

Therefore, all off-diagonal interaction terms in equation 3 vanish. For any deleted coordinate set $\mathcal{D}$,

$$\mathcal{L}(T_{\mathrm{WUS}};\mathcal{D}) = \sum_{i\in\mathcal{D}} s_i^2. \tag{6}$$

Each transformed coordinate consequently has an explicit deletion cost, and retaining coordinates associated with the largest singular values minimizes the error within the WUS basis.

Theorem 1 (Optimal Local Coordinate Deletion). *Suppose $\Sigma_X \succ 0$ and $\Sigma_W \succ 0$. Let P delete k coordinates and construct T_{WUS} as in equation 4. Then*

$$\min_{T \in \mathrm{GL}(d)} \mathcal{L}(T) = \mathcal{L}(T_{\mathrm{WUS}}) = \sum_{i=d-k+1}^{d} s_i^2. \tag{7}$$

Proof. For any admissible T, the approximation $\widehat{Y}(T)$ has rank at most $r = d - k$. By the Eckart–Young–Mirsky theorem, $\mathcal{L}(T) \geq \sum_{i=r+1}^{d} \sigma_i(XW)^2$. Let $Z = XL_X^{-\top}$, so that $Z^\top Z = I$ and $XW = ZL_X^\top W$. Hence, XW and $L_X^\top W$ have the same nonzero singular values. Moreover, $(L_X^\top W)(L_X^\top W)^\top = (L_W^\top L_X)^\top (L_W^\top L_X)$, and therefore $\sigma_i(XW) = s_i$. equation 5 shows that WUS attains this lower bound. □

The theorem establishes optimality for a fixed linear map and fixed calibration activations over unrestricted invertible transformations. It does not imply optimality for the full nonlinear network. WUS is also not the unique minimizer: diagonal rescaling of its transformed coordinates leaves the retained and deleted contributions unchanged. The result instead characterizes an optimal deletion construction and establishes a direct connection to rank-constrained approximation: WUS realizes the best rank-r local output reconstruction error through a change of basis followed by coordinate deletion.

3.3 Practical Design for Transformer LLMs

The unrestricted construction above provides a theoretical solution to the local deletion problem, but applying a full transformation to high-dimensional Transformer representations is expensive. We therefore develop a scalable approximation based on local input projections, shared transformations, and blockwise WUS construction.

Integration into attention and MLP modules. We replace each selected linear map with an input projection followed by a smaller dense linear map while preserving its output width. The projection is applied immediately before the linear map, allowing the compressed module to retain the original interfaces with normalization, nonlinearities, and residual connections. In particular, the residual-stream width and input embeddings remain unchanged. Each transformation is constructed from the activations entering its associated linear map and the corresponding weights. Maps operating on different intermediate representations use separate transformations, whereas maps sharing the same input can share both the basis and projected activations. The same construction can also be applied to the language-model output head while preserving the vocabulary dimension.

Sharing projections across linear maps. For linear maps $W^{(1)}, \ldots, W^{(J)}$ sharing the same input, we construct a common basis from the concatenated weight matrix $W_{\mathrm{group}} = [W^{(1)} \ \cdots \ W^{(J)}]$. Its weight second moment is $\Sigma_{W,\mathrm{group}} = \sum_{j=1}^{J} W^{(j)} W^{(j)\top}$. The resulting reconstruction objective is the sum of the individual map errors, allowing the same input projection to be shared across the group. We use this construction for the gate and up projections. Each map retains its own compressed weight, while the shared projection is stored and evaluated only once. This sharing amortizes the additional parameter cost introduced by the transformation across multiple linear maps, reducing the storage overhead of the compressed representation.

Scalable basis construction. To avoid constructing a full $d \times d$ transformation, we partition the input coordinates into disjoint blocks $\{\mathcal{B}_b\}_{b=1}^{B}$. For each block, let $X_b = X_{:,\mathcal{B}_b}$ and $W_b = W_{\mathcal{B}_b,:}$, with second moments $\Sigma_{X,b} = X_b^\top X_b$ and $\Sigma_{W,b} = W_b W_b^\top$. We then compute an independent WUS transformation T_b within each block using equation 4. For block width h, the activation moments require $O(dh)$ storage instead of $O(d^2)$, while the Cholesky and SVD operations act on matrices of size at most $h \times h$, resulting in an aggregate factorization cost of $O(dh^2)$ for equal-sized blocks. If a moment matrix is not positive definite, we add diagonal damping until its Cholesky factorization succeeds. Larger blocks more closely approximate the unrestricted construction but require greater preprocessing and storage cost. The blockwise formulation therefore provides a practical trade-off

between approximation quality and construction overhead. Let $s_{b,1} \geq \cdots \geq s_{b,|\mathcal{B}_b|}$ denote the WUS spectrum within block b. From equation 6, coordinate i has local deletion cost $s_{b,i}^2$. Given a total retention budget r, we allocate r_b retained coordinates to each block approximately in proportion to its size, with $\sum_b r_b = r$, and retain the r_b coordinates with the largest spectral contributions. Thus, coordinate selection requires only sorting within each block and does not involve gradient-based optimization.

Compact execution and storage. Let $R_b \in \mathbb{R}^{r_b \times |\mathcal{B}_b|}$ select the retained coordinates in block b. The compressed computation is

$$\widehat{Y} = \sum_{b=1}^{B} \underbrace{X_b T_b^\top R_b^\top}_{\text{projected activations}} \underbrace{R_b T_b^{-\top} W_b}_{\text{compressed weights}} . \tag{8}$$

Defining the compact input projection $C_b = R_b T_b$ and compressed weight $\overline{W}_b = R_b T_b^{-\top} W_b$, each block contribution becomes $(X_b C_b^\top)\overline{W}_b$. The projected activations from all blocks are concatenated and consumed by a smaller dense linear map, reducing its input dimension while preserving its output dimension. If all coordinates are retained, the original computation is recovered exactly because $T_b^\top T_b^{-\top} = I$.

Ignoring biases, a compressed map stores $rm + \sum_b r_b|\mathcal{B}_b|$ scalar values compared with dm in the original map. For equal block width h, this becomes $rm + rh$. The projection storage is therefore included in the reported compression budget. For a group of maps sharing the same projection, this cost is counted only once.

4 Experiments

4.1 Experimental Setup

Models and Datasets. We use the WikiText (Merity et al., 2016) training set as the calibration data for most experiments, sampling 128 sequences with a sequence length of 2048. For the mathematics and code generation experiments, we instead use a mixed calibration set of 256 samples, consisting of 128 samples from `theblackcat102/evol-codealpaca-v1` (Luo et al., 2023) and 128 samples from `allenai/tulu-3-sft-personas-math` (Lambert et al., 2024). All baseline methods use the same calibration data and calibration configuration as *WUSPruner* to ensure a fair comparison. We evaluate our method across multiple model families, including Llama3-8B (Grattafiori et al., 2024), Qwen3-8B (Yang et al., 2025), and Qwen3.5-9B (Team, 2026). To further evaluate scalability, we extend the experiments to Qwen3.8-27B and Qwen2.5-72B.

Baselines. We compare *WUSPruner* with three representative compression baselines: Wanda-sp (Sun et al., 2024), SliceGPT (Ashkboos et al., 2024), and SVD-LLM (Wang et al., 2025). Wanda-sp is a structured variant of Wanda that ranks removable dimensions using weight magnitude and activation statistics. SliceGPT applies activation-based PCA rotations before dimension deletion. SVD-LLM performs activation-aware low-rank compression using truncation-aware whitening and singular value decomposition. For all comparisons, baseline methods use the same calibration data and calibration configuration as *WUSPruner*.

Evaluation. We evaluate the pruned models from three complementary perspectives. For language modeling, we report perplexity on WikiText-2 (Merity et al., 2016), PTB (Marcus et al., 1993), and C4 (Raffel et al., 2020). For commonsense reasoning and knowledge, we report accuracy on PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), ARC-Challenge (Clark et al., 2018), MMLU (Hendrycks et al., 2021), OpenBookQA (Mihaylov et al., 2018), and CMMLU (Li et al., 2024). We further evaluate mathematical reasoning and code generation on GSM-8K (Cobbe et al., 2021), GSM-Platinum, HumanEval (Chen et al., 2021), HumanEval-Plus, MBPP (Austin et al., 2021), and MBPP-Plus. Unless otherwise specified, all downstream evaluations are conducted using the *lm-evaluation-harness* package (Gao et al., 2024), and HumanEval and MBPP are evaluated using the EvalPlus framework (Liu et al., 2024).

Implementation Details. We implement *WUSPruner* in PyTorch with Hugging Face Transformers and compress decoder layers sequentially, using activations propagated through previously pruned

Table 1: Main results on commonsense reasoning, knowledge, and language modeling benchmarks across three LLMs with 25% compression ratio. We report zero-shot accuracy (%) and perplexity (PPL), where higher accuracy and lower perplexity indicate better performance. Bold values denote the best results among pruning methods, excluding the dense model.

Model	Method	Zero-shot Acc. ↑							Perplexity ↓			
		PIQA	HellaSwag	ARC-C	MMLU	OBQA	CMMLU	Avg.	Wiki2	PTB	C4	Avg.
Llama3-8B	Dense	80.3	79.2	53.6	61.9	45.0	48.5	61.4	6.1	9.9	9.3	8.4
	Wanda-sp	65.2	34.5	28.8	26.5	29.8	25.1	34.9	253.6	336.7	313.7	301.3
	SliceGPT	62.8	49.6	31.4	25.0	**34.0**	25.3	38.0	**12.1**	116.1	121.7	83.3
	WUSPruner (Ours)	**69.2**	**53.8**	**32.5**	**37.2**	33.0	**29.4**	**42.5**	14.8	**24.8**	**31.3**	**23.6**
Qwen3-8B	Dense	77.8	74.7	56.5	72.9	42.6	76.9	66.9	9.7	15.4	15.2	13.4
	Wanda-sp	65.8	46.4	28.6	38.1	29.8	29.9	39.7	46.0	39.2	43.3	42.8
	SliceGPT	67.6	**55.5**	35.3	31.6	33.4	28.6	42.0	**13.1**	67.8	43.8	41.5
	WUSPruner (Ours)	**67.8**	53.7	**39.2**	**54.3**	**34.6**	**47.5**	**49.5**	15.0	**28.6**	**33.7**	**25.7**
Qwen3.5-9B	Dense	80.3	78.0	55.8	78.6	42.8	80.3	69.3	8.6	12.9	11.9	11.1
	Wanda-sp	67.9	33.6	**48.3**	36.0	31.0	33.3	41.6	53.5	62.6	44.3	53.4
	SliceGPT	66.2	55.3	37.0	28.5	34.2	25.3	41.0	**11.2**	36.7	33.7	27.2
	WUSPruner (Ours)	**69.0**	**56.0**	39.0	**54.5**	**34.4**	**53.8**	**51.1**	12.4	**20.7**	**21.8**	**18.3**

Table 2: Performance on mathematical reasoning and code generation benchmarks using Qwen3.5-9B with 25% compression ratio. We report accuracy (%) across six benchmarks and their average.

Method	Zero-shot Acc. ↑						
	HumanEval	HumanEval-Plus	MBPP	MBPP-Plus	GSM-8K	GSM-Platinum	Avg.
Dense	84.8	79.9	78.6	63.8	87.4	70.6	77.5
Wanda-sp	4.9	4.3	11.1	9.5	4.2	3.1	6.2
SliceGPT	62.8	55.5	54.2	44.1	56.4	19.3	48.7
WUSPruner (Ours)	**64.0**	**57.3**	**57.1**	**49.5**	**58.9**	**37.3**	**54.0**

layers for calibration. WUS transformations are constructed from activation and weight second moments using Cholesky factorization followed by SVD; diagonal damping is applied when needed for numerical stability. For Llama, Qwen2/Qwen2.5, and Qwen3, we prune the attention query and output projections together with the MLP gate, up, and down projections, sharing the input transformation between the gate and up projections. For Qwen3.5, we use the same targets in full-attention layers and prune only the MLP projections in linear-attention layers. We additionally compress the language-modeling head along its input dimension while preserving the vocabulary output. Latency is measured on a single L40S GPU with batch size 1 and sequence length 2048. We use 3 warm-up runs and average over 10 measured runs with 64 decoding steps. All experiments are conducted on 10×L40S GPU server.

4.2 Main Results.

Results on Commonsense Benchmarks. We compare *WUSPruner* with Wanda-sp and SliceGPT across three language models, as summarized in Table 1. *WUSPruner* consistently achieves the highest average accuracy among the pruning methods, obtaining 42.5%, 49.5%, and 51.1% on Llama3-8B, Qwen3-8B, and Qwen3.5-9B, respectively. These results correspond to improvements of 4.5, 7.5, and 9.5 percentage points over the strongest baseline for each model. Moreover, *WUSPruner* achieves the highest accuracy on five out of six individual benchmarks for each model, demonstrating its effectiveness across different model architectures and evaluation tasks. Notably, *WUSPruner* exhibits substantial improvements on knowledge-intensive benchmarks. On MMLU, it achieves accuracy gains of 10.7, 16.2, and 18.5 percentage points over the strongest baseline for Llama3-8B, Qwen3-8B, and Qwen3.5-9B, respectively. Similar improvements are observed on CMMLU, where *WUSPruner* outperforms the strongest baseline by 4.1, 17.6, and 20.5 percentage points. These results indicate that *WUSPruner* is particularly effective at preserving knowledge-related capabilities under structured pruning.

In terms of language modeling performance, *WUSPruner* achieves the lowest average perplexity across all three models, reducing it from 83.3 to 23.6 on Llama3-8B, from 41.5 to 25.7 on Qwen3-8B, and from 27.2 to 18.3 on Qwen3.5-9B compared with the strongest baseline in each case. Interestingly, while SliceGPT achieves slightly lower perplexity on WikiText-2, its performance deteriorates substantially on PTB and C4, suggesting a potential sensitivity to the calibration data distribution. Since WikiText is used for calibration, this discrepancy may indicate that SliceGPT's activation-based transformation is more closely adapted to the calibration distribution, potentially limiting its generalization to other text distributions. In contrast, *WUSPruner* maintains consistently lower perplexity on PTB and C4 across all three models, suggesting improved robustness across

Table 3: Evaluation of *WUSPruner* on larger language models, including Qwen3.8-27B and Qwen2.5-72B with 25% compression ratio.

Model	Method	Zero-shot Acc. ↑							Perplexity ↓			
		PIQA	HellaSwag	ARC-C	MMLU	OBQA	CMMLU	Avg.	Wiki2	PTB	C4	Avg.
Qwen3.8-27B	Dense	81.6	82.9	58.5	83.4	46.2	71.7	70.7	7.0	8.8	11.0	8.9
	Wanda-sp	71.6	66.1	37.3	55.4	37.2	43.9	51.9	24.5	24.2	25.7	24.8
	SliceGPT	70.1	63.7	43.0	37.2	40.2	28.6	47.1	**8.7**	29.3	25.9	21.3
	WUSPruner (Ours)	**75.7**	**68.6**	**51.1**	**69.1**	**40.6**	**60.5**	**60.9**	9.1	**13.3**	**16.9**	**13.1**
Qwen2.5-72B	Dense	83.6	86.0	62.7	83.3	46.2	89.9	75.2	3.8	8.4	9.1	7.1
	Wanda-sp	70.1	65.4	47.1	59.3	**43.4**	53.2	56.4	12.9	24.5	**17.2**	18.2
	SliceGPT	73.2	69.2	43.3	60.3	40.6	49.6	56.0	**6.4**	48.0	26.1	26.8
	WUSPruner (Ours)	**76.2**	**71.2**	**53.0**	**72.0**	41.4	**69.8**	**63.9**	8.1	**13.9**	17.5	**13.1**

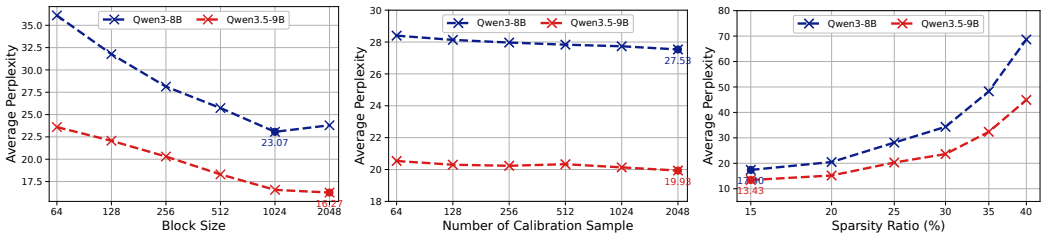
evaluation datasets. These findings highlight the importance of jointly considering activation and weight statistics when constructing pruning transformations to preserve language modeling performance beyond the calibration distribution. Together, these results demonstrate that *WUSPruner* provides a more favorable balance between downstream task accuracy and language modeling quality across the evaluated models.

Results on Math and Coding Benchmarks. Table 2 evaluates the mathematical reasoning and coding capabilities of the pruned Qwen3.5-9B model across six benchmarks. *WUSPruner* consistently outperforms both Wanda-sp and SliceGPT on all evaluated tasks, achieving an average accuracy of 54.0%, compared with 48.7% for SliceGPT and 6.2% for Wanda-sp. In particular, *WUSPruner* achieves substantial improvements on GSM-Platinum, increasing accuracy from 19.3% to 37.3%, an absolute gain of 18.0 percentage points over SliceGPT. It also achieves improvements of 5.4 and 2.5 percentage points on MBPP-Plus and GSM-8K, respectively. These results suggest that *WUSPruner* better preserves mathematical reasoning and coding capabilities under structured pruning, particularly on challenging tasks that require multi-step reasoning and precise code generation. The results on commonsense benchmarks can be found in the Appendix B.1.

Scaling Up to Larger Models. To evaluate the scalability of *WUSPruner*, we extend our experiments to Qwen3.8-27B and Qwen2.5-72B, as shown in Table 3. *WUSPruner* consistently outperforms Wanda-sp and SliceGPT on both models, achieving average accuracies of 60.9% and 63.9%, respectively. These results represent improvements of 9.0 and 7.5 percentage points over the strongest baseline for each model. Notably, *WUSPruner* achieves substantial gains on knowledge-intensive benchmarks, outperforming the strongest baseline by 13.7 and 11.7 percentage points on MMLU, and by 16.6 and 16.6 percentage points on CMMLU for the 27B and 72B models, respectively. In terms of language modeling performance, *WUSPruner* achieves the lowest average perplexity on both models, reducing it from 21.3 to 13.1 on Qwen3.8-27B and from 18.2 to 13.1 on Qwen2.5-72B compared with the strongest baseline. Consistent with the observations on smaller models, *WUSPruner* achieves lower perplexity on PTB and C4, while SliceGPT performs better on WikiText-2. These results suggest that *WUSPruner* maintains its effectiveness across different model scales, preserving both downstream task accuracy and language modeling quality in larger language models.

4.3 Analysis

Sensitivity to Block Size, Calibration Size, and Sparsity. We evaluate the sensitivity of *WUSPruner* to block size, calibration-set size, and pruning sparsity, as shown in Figure 2. We first observe that increasing the block size consistently improves performance on both Qwen3-8B and Qwen3.5-9B, as reflected by the decreasing average perplexity. This behavior is consistent with our formulation: a larger block relaxes the blockwise constraint and more closely approximates the original WUS transformation derived in Section 3.2. However, larger blocks also incur higher computational and memory costs when constructing the transformation, leading to a practical trade-off between pruning quality and preprocessing overhead. In contrast, *WUSPruner* is largely insensitive to the number of calibration samples. Increasing the calibration set from 64 to 2048 samples results in only minor variations in perplexity for both models, indicating that the proposed transformation can be estimated reliably from a relatively small calibration set. Finally, as expected, performance gradually degrades as the pruning ratio increases, with the degradation becoming more pronounced under aggressive sparsity. Nevertheless, both models exhibit consistent trends across sparsity levels, suggesting that *WUSPruner* behaves stably over a broad range of compression ratios. Overall, these results show that *WUSPruner* benefits from larger transformation blocks while remaining robust to



(a) Different Block Sizes (b) Different Calibration Sizes (c) Different Sparsity

Figure 2: Sensitivity analysis of WUSPruner with respect to block size, calibration set size, and pruning ratio on Qwen3-8B and Qwen3.5-9B. Larger block sizes generally yield lower perplexity, consistent with a closer approximation to the original WUS transformation, while increasing computational overhead. WUSPruner remains stable across a wide range of calibration set sizes, and its performance degrades smoothly as the pruning ratio increases.

Table 4: Perplexity evaluation of *WUSPruner* compared with SVD-LLM at a 25% sparsity ratio.

Method	Perplexity ↓			
	Wiki2	PTB	C4	Avg.
Dense	9.7	15.4	15.2	13.4
SVD-LLM	15.1	45.9	93.3	51.4
WUSPruner (Ours)	**15.0**	**28.6**	**33.7**	**25.7**

Table 5: Speedup and memory reduction. Speedup and peak GPU memory reduction are measured relative to the dense model under the same inference configuration.

Sparsity	Method	Inference Efficiency		GPU Memory	
		Latency (ms) ↓	Speedup ↑	Peak (GB) ↓	Reduction ↑
-	Dense	183.9	1.00×	16.2	-
25%	*WUSPruner*	160.8	1.14×	13.2	-18.5%
50%	*WUSPruner*	124.1	1.48×	9.4	-42.0%

calibration-set size and exhibiting predictable degradation as compression becomes more aggressive. Additional results across compression ratios are provided in Appendix B.3.

Comparison with SVD-based Compression. We compare *WUSPruner* with SVD-LLM on Qwen3-8B at a 25% sparsity ratio, using the same calibration and evaluation setup, as shown in Table 4. *WUSPruner* consistently outperforms SVD-LLM across all three datasets. While the two methods achieve comparable perplexity on WikiText-2 (15.0 vs. 15.1), *WUSPruner* substantially reduces perplexity on PTB from 45.9 to 28.6 and on C4 from 93.3 to 33.7, leading to a much lower average perplexity of 25.7 compared with 51.4 for SVD-LLM. These results suggest that *WUSPruner* preserves language modeling quality more effectively across different data distributions under the same compression ratio. Additional benchmark comparisons are provided in Appendix B.2.

Speedup and Memory Reduction. Table 5 reports the practical inference benefits of *WUSPruner* under different pruning ratios. Compared with the dense model, *WUSPruner* consistently reduces both inference latency and peak GPU memory usage, with larger gains obtained at higher sparsity. At 25% sparsity, the latency decreases from 183.9 ms to 160.8 ms, while peak GPU memory is reduced from 16.2 GB to 13.2 GB. When the sparsity is increased to 50%, the latency further decreases to 124.1 ms and the peak memory usage drops to 9.4 GB. This demonstrates that the structural compression introduced by *WUSPruner* translates into tangible runtime benefits, rather than merely reducing the nominal parameter count. Moreover, the improvements become more pronounced as the pruning ratio increases, highlighting the potential of *WUSPruner* for reducing both computational and memory costs in resource-constrained deployment scenarios. Additional runtime results are provided in Appendix B.4.

5 Conclusion

We presented *WUSPruner*, a transform-based framework for structured pruning of large language models. By formulating pruning as coordinate deletion under a change of basis, we show that the WUS transformation minimizes local reconstruction error over invertible transformations and yields a principled spectral criterion for coordinate selection. We further develop a practical block-wise WUS approximation with compact shared projections for Transformer LLMs. Experiments on models from 8B to 72B demonstrate consistent improvements over structured pruning baselines, together with practical reductions in inference latency and memory usage. These results highlight basis optimization as a principled and effective direction for structured LLM compression.

AI USE STATEMENT

In this work, we used generative AI tools for assist in the writing of proofs, design or provide feedback on research methodology or experiments, implement methods. We have not used generative AI tools for assist with translation, support qualitative and thematic data analysis, interpret results, propose or refine hypotheses, and Generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, clean and reformat dataset, are not applicable to this work. Additionally, we used generative AI tools for create or edit software code. We have reviewed all AI-assisted work. We checked the LLM-generated code carefully and tested all results multiple times. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICAL STATEMENT

This work aims to improve the efficiency of LLM deployment by reducing memory and inference costs. Positive impacts include improved accessibility of capable language models in resource-constrained environments and reduced computational cost. Potential negative impacts include making capable generative models easier to deploy for misuse, such as spam or misinformation generation. The method does not introduce new model capabilities, and standard safeguards for the underlying LLMs should still be applied after compression.

REPRODUCIBILITY STATEMENT

We describe the formulation and algorithm of *WUSPruner* in Section 3, and provide the experimental settings and implementation details in Section 4. These descriptions and analyses clarify the conditions under which our results are obtained and support their reproduction.

REFERENCES

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. URL https://arxiv.org/abs/2312.11983.

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.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Arnav Chavan, Zhuang Liu, Deepak Gupta, Eric Xing, and Zhiqiang Shen. One-for-all: Generalized lora for parameter-efficient fine-tuning. *arXiv preprint arXiv:2306.07967*, 2023.

Jiale Chen, Vage Egiazarian, Roberto L. Castro, Torsten Hoefler, and Dan Alistarh. WUSH: Near-optimal adaptive transforms for LLM quantization. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2512.00956v3.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex

Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

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:1803.05457v1*, 2018.

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

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, 2023. URL https://arxiv.org/abs/2301.00774.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*, 2024.

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

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. *arXiv preprint arXiv:2106.09685*, 2021.

Eldar Kurtić, Elias Frantar, and Dan Alistarh. Ziplm: Inference-aware structured pruning of language models. *Advances in Neural Information Processing Systems*, 36:65597–65617, 2023.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. Cmmlu: Measuring massive multitask language understanding in chinese. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11260–11285, 2024.

Jiawei Liu, Songrun Xie, Junhao Wang, Yuxiang Wei, Yifeng Ding, and Lingming Zhang. Evaluating language models for efficient code generation. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=IBCBMeAhmC.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct, 2023.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. LLM-Pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2305.11627.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://www.aclweb.org/anthology/J93-2004.

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

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024.

Shengkun Tang, Oliver Sieberling, Eldar Kurtic, Zhiqiang Shen, and Dan Alistarh. Darwinlm: Evolutionary structured pruning of large language models. *arXiv preprint arXiv:2502.07780*, 2025.

Shengkun Tang, Zekun Wang, Bo Zheng, Liangyu Wang, Rui Men, Siqi Zhang, Xiulong Yuan, Zihan Qiu, Zhiqiang Shen, and Dayiheng Liu. Slimqwen: Exploring the pruning and distillation in large moe model pre-training. *arXiv preprint arXiv:2605.08738*, 2026.

Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Chengzhong Xu. Hydralora: An asymmetric lora architecture for efficient fine-tuning. *Advances in Neural Information Processing Systems*, 37: 9565–9584, 2024.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2403.07378.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, Qiang Wu, Yan Yan, and Guangyu Sun. Asvd: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*, 2023.

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

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024.

A NOTATION SUMMARY

Table 6: Summary of the main notation used in the paper.

Symbol	Description	Shape / Domain
X	Calibration activations	$\mathbb{R}^{n\times d}$
W	Weight matrix of a linear layer	$\mathbb{R}^{d\times m}$
d	Original input dimension	–
r	Number of retained coordinates	–
T	Invertible change-of-basis transformation	$\mathrm{GL}(d)$
P	Diagonal projector selecting deleted coordinates	$\mathbb{R}^{d\times d}$
$\mathcal{L}(T)$	Local output reconstruction loss after coordinate deletion	$\mathbb{R}_{\geq 0}$
$\Sigma_X = X^\top X$	Activation second-moment matrix	$\mathbb{R}^{d\times d}$
$\Sigma_W = WW^\top$	Weight second-moment matrix	$\mathbb{R}^{d\times d}$
L_X, L_W	Cholesky factors of Σ_X and Σ_W	$\mathbb{R}^{d\times d}$
$S = \mathrm{diag}(s_1, \ldots, s_d)$	WUS spectrum obtained from $L_W^\top L_X = USV^\top$	$\mathbb{R}^{d\times d}$
T_{WUS}	WUS transformation, $S^{-1/2}U^\top L_W^\top$	$\mathbb{R}^{d\times d}$
s_i^2	Reconstruction cost associated with deleting coordinate i	$\mathbb{R}_{>0}$
B_b	Coordinate set of block b	$B_b \subseteq \{1, \ldots, d\}$
$X_b,\ W_b$	Activation and weight submatrices for block b	–
T_b	Blockwise WUS transformation	$\mathbb{R}^{\lvert B_b\rvert\times\lvert B_b\rvert}$
r_b	Number of retained coordinates in block b	–
R_b	Selector of retained coordinates in block b	$\mathbb{R}^{r_b\times\lvert B_b\rvert}$
$C_b = R_b T_b$	Compact input projection	$\mathbb{R}^{r_b\times\lvert B_b\rvert}$
$\overline{W}_b$	Compressed weight matrix	$\mathbb{R}^{r_b\times m}$

B MORE EXPERIMENTAL RESULTS

B.1 MORE BENCHMARK RESULTS OF MODEL CALIBRATED BY MATH AND CODING DATA.

Table 7: Additional evaluation of Qwen3.5-9B calibrated with the mixed mathematics and coding dataset at a 25% compression ratio.

Method	Zero-shot Acc. ↑							Perplexity ↓			
	PIQA	HellaSwag	ARC-C	MMLU	OBQA	CMMLU	Avg.	Wiki2	PTB	C4	Avg.
Dense	80.3	78.0	55.8	78.6	42.8	80.3	69.3	8.6	12.9	11.9	11.1
Wanda-sp	67.4	43.5	33.1	45.0	**31.0**	41.0	43.5	73.5	74.7	67.1	71.8
SliceGPT	58.3	37.8	27.2	32.5	27.4	29.0	35.4	53.1	58.6	57.6	56.4
WUSPruner (Ours)	**68.1**	**50.5**	**38.9**	**52.1**	30.6	**54.1**	**49.1**	**21.8**	**29.8**	**28.9**	**26.8**

To examine whether the choice of calibration domain affects general-purpose capabilities, we further evaluate models calibrated with the mixed mathematics and coding dataset on commonsense, knowledge, and language modeling benchmarks. *WUSPruner* remains consistently stronger than the pruning baselines, achieving the highest accuracy on five of the six downstream benchmarks and improving the average accuracy to 49.1%, compared with 43.5% for Wanda-sp and 35.4% for SliceGPT. The advantage is particularly pronounced in language modeling, where *WUSPruner* achieves an average perplexity of 26.8, substantially lower than 71.8 for Wanda-sp and 56.4 for SliceGPT. These results indicate that *WUSPruner* is relatively robust to the calibration domain and better preserves general model capabilities even when the calibration data are specialized toward mathematics and coding.

B.2 MORE BENCHMARK COMPARISON WITH SVD-LLM.

We further compare *WUSPruner* with SVD-LLM on Qwen3-8B at a 25% compression ratio. Overall, *WUSPruner* achieves stronger downstream performance, improving the average accuracy from 47.0% to 49.5%. The gains are particularly pronounced on knowledge-intensive benchmarks, with improvements of 8.8 and 9.8 percentage points on MMLU and CMMLU, respectively, while SVD-LLM performs better on HellaSwag and OpenBookQA. *WUSPruner* also substantially improves

Table 8: Additional comparison with SVD-LLM on Qwen3-8B at a 25% compression ratio.

Method	Performance ↑							Perplexity ↓			
	PIQA	HellaSwag	ARC-C	MMLU	OBQA	CMMLU	Avg.	Wiki2	PTB	C4	Avg.
Dense	77.8	74.7	56.5	72.9	42.6	76.9	66.9	9.7	15.4	15.2	13.4
SVD-LLM	67.6	**54.3**	38.9	45.5	**38.2**	37.7	47.0	15.1	45.9	93.3	51.4
WUSPruner (Ours)	**67.8**	53.7	**39.2**	**54.3**	34.6	**47.5**	**49.5**	**15.0**	**28.6**	**33.7**	**25.8**

language modeling quality, reducing the average perplexity from 51.4 to 25.8, with especially large reductions on PTB and C4. These results show that the proposed transform-based pruning approach provides a better overall trade-off between downstream task performance and language modeling quality than activation-aware low-rank compression under the same compression ratio.

B.3 Benchmark Performance with Different Compression Ratio

Table 9: Benchmark performance of *WUSPruner* under different compression ratios on Qwen3-8B and Qwen3.5-9B.

Model	Sparsity	Zero-shot Acc. ↑							Perplexity ↓			
		PIQA	HellaSwag	ARC-C	MMLU	OBQA	CMMLU	Avg.	Wiki2	PTB	C4	Avg.
Qwen3-8B	15%	74.1	64.7	50.6	64.8	39.6	64.1	59.5	12.0	19.3	20.9	17.4
	20%	71.2	59.4	43.5	60.2	37.6	56.3	54.7	13.5	22.8	25.2	20.5
	25%	67.8	53.7	39.2	54.3	34.6	47.5	49.5	15.0	28.6	33.7	25.8
	30%	63.0	47.8	33.1	47.7	30.6	36.8	43.1	17.5	41.2	44.3	34.3
	35%	60.7	42.3	28.2	37.7	30.6	29.9	38.2	21.4	59.1	64.3	48.2
	40%	59.1	37.8	24.4	30.2	28.8	25.7	34.3	27.2	87.4	91.4	68.6
Qwen3.5-9B	15%	75.9	68.8	51.5	70.6	40.0	68.9	62.6	9.8	15.3	15.2	13.4
	20%	73.8	63.3	46.3	65.0	37.8	61.1	57.8	10.8	17.2	17.6	15.2
	25%	69.0	56.0	39.0	54.5	34.4	53.8	51.1	12.4	20.7	21.8	18.3
	30%	64.4	48.1	31.7	47.5	32.4	44.6	44.7	14.9	27.2	28.8	23.6
	35%	61.3	42.3	27.6	33.9	29.0	35.7	38.3	18.4	39.0	39.7	32.3
	40%	58.7	36.8	23.7	26.9	26.0	28.2	33.3	23.5	55.1	56.3	44.9

Performance across Compression Ratios. We further evaluate *WUSPruner* on Qwen3.5-9B under compression ratios ranging from 15% to 40%. As the compression ratio increases, performance degrades smoothly and consistently across all evaluated benchmarks. The average downstream accuracy decreases from 62.6% at 15% compression to 51.1% at 25%, while the average perplexity increases moderately from 13.4 to 18.3. More aggressive compression leads to a more pronounced degradation: at 40%, the average accuracy drops to 33.3% and the average perplexity rises to 44.9. Similar monotonic trends are observed across individual commonsense, knowledge, and language modeling benchmarks. These results indicate that *WUSPruner* provides a predictable trade-off between compression ratio and model quality, with moderate compression preserving substantially more of the original model performance than aggressive compression.

B.4 Runtime of *WUSPruner*.

Table 10: Runtime comparison of different compression methods. We report the total wall-clock time required to compress each model using the same calibration setting.

Method	Qwen3-8B	Qwen3.5-9B	Qwen3.8-27B	Qwen2.5-72B
Wanda-sp	0.2	0.3	0.8	1.1
WUSPruner (Ours)	0.5	0.67	1.5	2.3

We further measure the wall-clock time required to compress models of different scales using the same calibration setting as in the main experiments. As expected, *WUSPruner* introduces additional preprocessing cost compared with the lightweight Wanda-sp baseline, since it requires collecting second-moment statistics and constructing the WUS transformations. Nevertheless, the overall compression cost remains modest: *WUSPruner* requires only 0.5 and 0.67 hours for Qwen3-8B and Qwen3.5-9B, respectively, and 1.5 and 2.3 hours for Qwen3.8-27B and Qwen2.5-72B. The 8B

and 9B models are processed on a single GPU, while the 27B and 72B models use 4 and 6 GPUs, respectively, primarily to accommodate the larger models in memory. Overall, the additional pre-processing introduced by WUS is small relative to model deployment and remains practical even at the 72B scale.