Reduced Matrix Multiplication: Input-Adaptive Matrix-Product Reduction for LLM Inference

Anonymous ACL submission

Abstract

Transformer-based language models achieve strong performance but incur substantial inference cost due to repeated high-dimensional matrix multiplications. We propose Reduced Matrix Multiplication (RMM), a training-free, input-adaptive inference method that reduces Transformer matrix products by selecting informative slices along their contraction dimensions, without modifying model weights. Under a simple retention-ratio control, RMM provides a smooth and predictable accuracy–efficiency trade-off. Across language models ranging from 1B to 70B parameters, we show that larger models tolerate more aggressive reduction, and that RMM remains stable across discriminative evaluation, autoregressive generation, and long-context reasoning. We further show that the same principle extends to multimodal vision–language inference. Mechanistic ablations reveal a structural asymmetry within Transformers: attention-side computations are substantially more reducible than MLP components. Finally, wall-clock benchmarks with custom kernels on an NVIDIA A100 show that these computational savings can translate into practical runtime gains, especially at longer sequence lengths. Together, these results position RMM as a scalable direction for input-adaptive inference-time optimization.

1 Introduction

Transformer models (Vaswani et al., 2017) continue to improve with scale (Kaplan et al., 2020), but their inference cost also grows rapidly. As model size increases, efficient inference becomes increasingly important in practice. A major source of this cost is the repeated high-dimensional matrix products in attention and feed-forward layers. This raises a basic question: during inference, do all indices along the shared multiplication axes of these matrix products need to be evaluated for every input, or can part of this computation be reduced adaptively while preserving model behavior?

Prior work has explored redundant computation in Transformer inference from several angles (Liu et al., 2021; Peng et al., 2023; Sajjad et al., 2023). One line of work simplifies model computation through structured pruning, low-rank approximation, or dimension reduction (Sun et al., 2024; Ma et al., 2023; Frantar and Alistarh, 2023; Ashkboos et al., 2024; Gao et al., 2024). Another line of work reduces context-level redundancy through token compression, KV-cache management, or decoding-time scheduling (Li et al., 2023b; Pan et al., 2024; Zhang et al., 2023; Xiao et al., 2024; Fu et al., 2025; Shah et al., 2024; Yuan et al., 2025). These methods reduce inference cost by modifying fixed model structures, shortening inputs or caches, or changing execution flow, but they do not directly ask whether the contracted computation inside each Transformer matrix product can be reduced adaptively for the current input. A closely related direction exploits activation sparsity by skipping small-magnitude activation entries during inference (Liu et al., 2025; Lee et al., 2024). Although such methods also use input-dependent activation information, their primary object is the sparsification of activation tensors or hidden states. Our focus is different: rather than sparsifying hidden states themselves, we reduce the shared multiplication axis of each matrix product. Depending on the operation, this axis may correspond to hidden channels in linear or MLP projections, attention-head feature dimensions in attention-score computation, or token positions in the attention-value product. This framing distinguishes RMM from fixed component pruning, activation-state sparsification, and token pruning: the goal is to reduce the contracted computation performed by each matrix product at inference time.

In this work, we propose *Reduced Matrix Multiplication* (RMM), a training-free, input-adaptive

method for Transformer inference. Rather than executing the full matrix products in attention and MLP layers, RMM dynamically selects and computes informative indices along the shared multiplication axis of each product, without modifying model weights. Beyond acceleration, RMM provides a controllable way to reduce computation through a simple retention ratio, enabling a systematic study of redundancy in Transformer inference. Across models ranging from 1B to 70B parameters and diverse downstream tasks, we find that larger models generally tolerate more aggressive reduction. We further observe a clear structural asymmetry within Transformers: attention-side computations are substantially more reducible, whereas MLP components are much more sensitive to reduction. We also show that RMM extends to multimodal vision–language inference, and implement Triton kernels to verify that the reduced matrix products can translate into practical wall-clock speedups.

2 Related Work

Prior work has shown that Transformer inference contains substantial redundancy across neurons, layers, attention heads, and larger structural units: models can often tolerate pruning or reduction while preserving performance (Peng et al., 2023; Sajjad et al., 2023). These findings suggest that not all inference-time computation is equally indispensable.

One major direction reduces computation by modifying or compressing model structure, including structured pruning, unstructured pruning, low-rank approximation, and dimension reduction (Sun et al., 2024; Ma et al., 2023; Frantar and Alistarh, 2023; Ashkboos et al., 2024; Gao et al., 2024). Such methods typically determine the reduction pattern before inference and operate on a fixed model structure, answering which parts of the model can be removed or compressed in advance rather than whether computation should vary with the current input.

Another direction reduces redundancy in the input context, cache, attention pattern, or execution process through token compression, KV-cache management, decoding-time skipping, sparse attention, efficient attention kernels, or scheduling optimization (Li et al., 2023b; Pan et al., 2024; Zhang et al., 2023; Xiao et al., 2024; Shah et al., 2024; Yuan et al., 2025). These methods can reduce latency or memory overhead, especially in long-context settings, but they mainly act on inputs, cached tokens, attention patterns, or execution flow rather than directly reducing the high-dimensional matrix products inside attention and feed-forward layers.

Closely related to our work are activation-sparsity methods, which skip small-magnitude activation entries during inference (Liu et al., 2025; Lee et al., 2024). These methods are also input-dependent and often training-free, but their sparsity masks are defined over activation entries or hidden states. In contrast, RMM defines the retained set over the contracted axis of each matrix product, thereby reducing the multiplication itself rather than sparsifying activations as model states. This distinction is important because the shared axis can have different meanings across operations: it may correspond to hidden channels in MLP or linear projections, attention-head feature dimensions in attention-score computation, or token positions in the attention-value product.

Our work is also related to classical randomized approximate matrix multiplication (Drineas et al., 2006), which approximates a matrix product by sampling column-row pairs according to importance distributions. Such estimators can in principle be used within a single Transformer forward pass. However, under a fixed reduction budget, stochastic sampling introduces per-instance approximation variance; reducing this variance generally requires more sampled pairs, which weakens the achievable computational savings. RMM adopts the same matrix-product perspective while using deterministic, activation-aware index selection along the shared multiplication axis, yielding predictable reduced products whose effects we evaluate across Transformer components. Overall, RMM takes a direct computational perspective: it reduces matrix products themselves.

3 Methodology

3.1 Preliminaries

Transformer inference is mainly composed of matrix multiplications in self-attention and feed-forward networks (MLPs). Let the input hidden states at layer l be $X^{(l)} \in \mathbb{R}^{L\times d}$, where L is the sequence length and d is the hidden dimension. In self-attention, the input is linearly projected to queries, keys, and values as $Q = X^{(l)}W_Q$, $K = X^{(l)}W_K$, and $V = X^{(l)}W_V$,

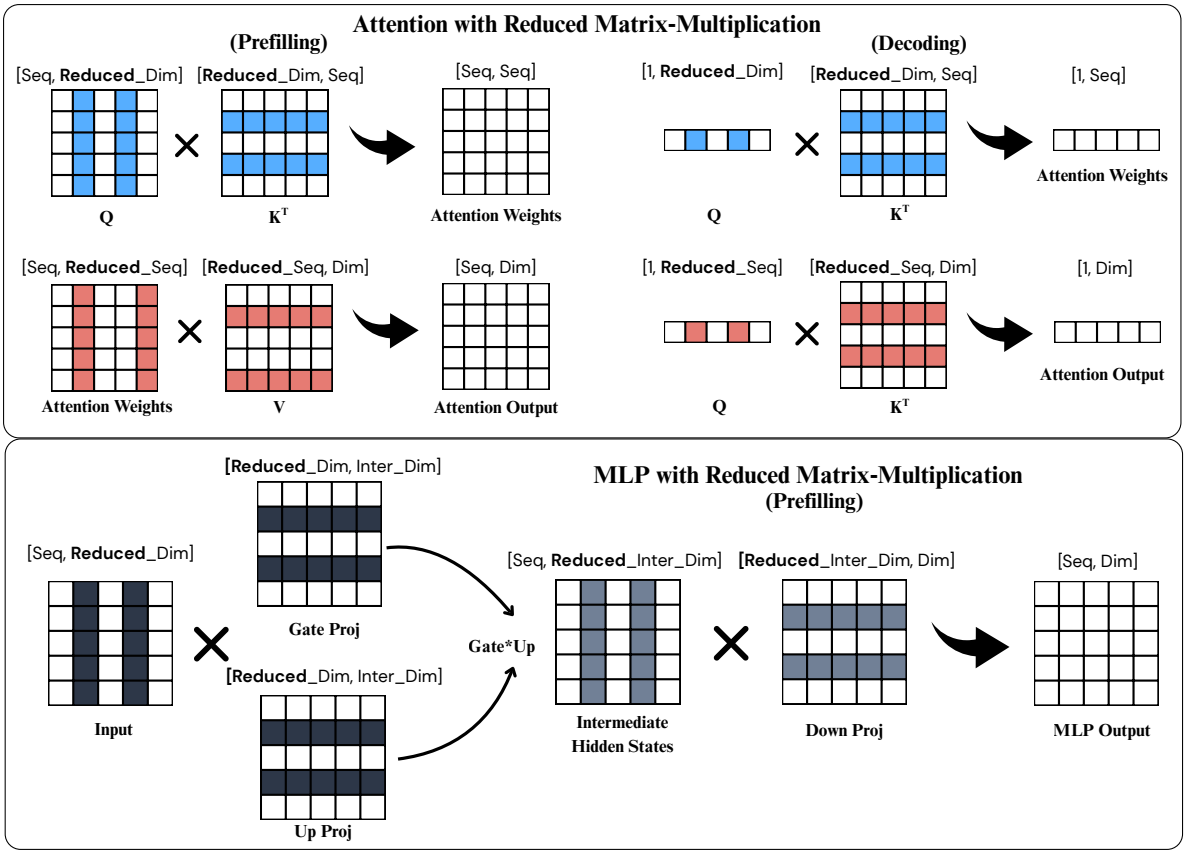

Figure 1: Application of RMM in major computations of Transformer language models.

where $W_Q, W_K, W_V \in \mathbb{R}^{d \times d_h}$, and d_h is the feature dimension of each attention head. Attention scores are computed by $QK^\top$, and the output is obtained by $\mathrm{softmax}(QK^\top)V$. Similarly, the MLP block consists of large linear transformations between activations and weight matrices. Although these computations arise in different modules, their core can be written in the unified form $Y = AB$, where $A \in \mathbb{R}^{n \times d}$ denotes the activation matrix determined by the current input, $B \in \mathbb{R}^{d \times m}$ denotes a weight matrix or intermediate representation, and $Y \in \mathbb{R}^{n \times m}$ is the output.

3.2 Reduced Matrix Multiplication

Classical approximate matrix multiplication establishes that the product AB can be approximated via Monte Carlo sampling: one repeatedly draws column-row pairs from the shared dimension according to a probability distribution derived from both matrices, rescales them, and averages over independent draws to obtain a low-error estimate in expectation (Drineas et al., 2006). However, Transformer inference performs only a single forward pass per input, leaving no opportunity to average over repeated samples, so such randomized estimators cannot be directly applied with reliable per-instance quality. This motivates a deterministic, input-adaptive approach. Based on the unified matrix multiplication form above, we define Reduced Matrix Multiplication(RMM) as follows. For a matrix product $Y = AB$, where $A \in \mathbb{R}^{n \times d}$ and $B \in \mathbb{R}^{d \times m}$, RMM selects an index set $\mathcal{I} \subseteq [d]$ with $|\mathcal{I}| = \lceil \rho d \rceil$, where $\rho \in (0, 1]$ is a user-controlled retention ratio, and computes

$$\mathrm{RMM}_\rho(A, B) \triangleq A_{:,\mathcal{I}} \, B_{\mathcal{I},:}.$$

Activation-aware dimension selection. To select $\mathcal{I}$, we assign each feature dimension $j \in [d]$ an importance score $s_j \triangleq \|A_{:,j}\|_2$, which measures the magnitude of feature j under the current input. Given a retention ratio ρ, we select $\mathcal{I} = \mathrm{TopK}(\{s_j\}_{j=1}^d, \lceil \rho d \rceil)$. This procedure is fully deterministic. Since the selection depends on the current activations, the resulting subspace may vary across inputs, layers, attention heads, and decoding steps. RMM adapts its computation to each input by selecting dimensions based on current activation magnitudes. This choice is theoretically grounded: we prove in Appendix E.1 that TopK selection by column norm is minimax optimal, min-

imizing the worst-case approximation error over all possible B at any given retention budget. Our ablation experiments (Section 6) further confirm that both dynamic selection and activation-aware scoring are essential to the effectiveness of RMM.

3.3 Applying RMM to Attention and MLPs

We apply RMM to Attention and MLP layers as follows (Figure 1).

Attention. Consider a single attention head with queries $Q \in \mathbb{R}^{L_q \times d_h}$, keys $K \in \mathbb{R}^{L_k \times d_h}$, and values $V \in \mathbb{R}^{L_k \times d_h}$. We compute feature scores $s_j = \|Q_{:,j}\|_2$ and select $\mathcal{I} = \mathrm{TopK}(\{s_j\}_{j=1}^{d_h}, \lceil \rho_d d_h \rceil)$, yielding reduced attention scores $\widetilde{S} = \frac{1}{\sqrt{d_h}} Q_{:,\mathcal{I}} K_{:,\mathcal{I}}^\top$. Attention weights are then obtained as $P = \mathrm{softmax}(\widetilde{S} + M)$, where M denotes the optional causal or attention mask. For grouped-query attention, dimension selection is performed per head on Q, with the corresponding dimensions gathered from the shared K and V tensors. We further optionally sparsify the attention–value multiplication PV over the token dimension by computing token scores $a_t = \|P_{:,t}\|_2$, selecting $\mathcal{T} = \mathrm{TopK}(\{a_t\}_{t=1}^{L_k}, \lceil \rho_t L_k \rceil)$, and evaluating $\widetilde{O} = P_{:,\mathcal{T}} V_{\mathcal{T},:}$.

MLP and linear projections. Given activations $X \in \mathbb{R}^{L \times d}$ and weights $W \in \mathbb{R}^{d \times d'}$, we compute feature scores $s_j = \|X_{:,j}\|_2$, select $\mathcal{I} = \mathrm{TopK}(\{s_j\}_{j=1}^{d}, \lceil \rho_d d \rceil)$, and evaluate $\widetilde{Y} = X_{:,\mathcal{I}} W_{\mathcal{I},:}$. The same rule applies to linear projections as well as to feed-forward layers.

3.4 Complexity

For a matrix multiplication $A \in \mathbb{R}^{n \times d}$ and $B \in \mathbb{R}^{d \times m}$, dense computation costs $O(ndm)$. With feature retention ratio ρ_d, RMM evaluates $A_{:,\mathcal{I}} B_{\mathcal{I},:}$ with $|\mathcal{I}| = \lceil \rho_d d \rceil$, reducing the cost to $O(n \rho_d d\, m)$. In attention, reducing $QK^\top$ over the head dimension lowers the cost from $O(L_q L_k d_h)$ to $O(L_q L_k \rho_d d_h)$. If token selection is also applied to the attention–value product, the cost of PV is reduced from $O(L_q L_k d_h)$ to $O(L_q \rho_t L_k d_h)$. RMM additionally requires computing feature scores and selecting top-k indices: computing $s_j = \|A_{:,j}\|_2$ costs $O(nd)$, while top-k selection over d (or L_k) is a lightweight vector-level operation. In practice, these overheads are small relative to the dense matrix multiplications that RMM replaces.

4 Experimental Setup

Overview. Our experiments are structured to validate both the effectiveness of dynamic, activation-aware pruning and the empirical insights it enables under controlled retention ratios. We first establish the necessity of dynamic selection by comparing RMM against representative weight-level static pruning methods under matched sparsity budgets on LLaMA 3.1 8B. We then sweep retention ratios across model scales from 1B to 70B to examine how performance degrades as computation is reduced and how redundancy varies with scale. Next, we test robustness under more realistic inference settings, including autoregressive generation and long-context reasoning. We further perform component-wise pruning analyses on attention and MLP blocks to identify which computations are more redundant and which are more critical. Finally, we evaluate generalization on a vision–language model and report wall-clock latency on an NVIDIA A100 GPU to verify that the computational savings translate into actual runtime improvements.

Models and tasks We evaluate RMM on a wide spectrum of pre-trained LLMs, including Llama 3.1 70B, Llama 3.1 8B, Llama 3.2 3B, Llama 3.2 1.5B (Grattafiori et al., 2024), Qwen3 32B and Qwen 3.1 7B (Yang et al., 2025), and Qwen2.5-VL-7B-Instruct (Bai et al., 2025). The benchmarks span multiple capabilities: (i) general QA and reasoning, including COPA (Gordon et al., 2012), PIQA (Bisk et al., 2020), COMMONSENSEQA (Talmor et al., 2019), ARC-EASY, ARC-CHALLENGE (Clark et al., 2018), and MMLU (Hendrycks et al., 2021); (ii) language modeling evaluation on WIKITEXT (Merity et al., 2016) and BOOKCORPUS (Zhu et al., 2015); (iii) mathematics and coding tasks, including GSM8K (Cobbe et al., 2021) and HUMANEVAL (Chen et al., 2021); (iv) long-context reasoning, including RULER-CWE and RULER-HOTPOT (Hsieh et al., 2024); (v) summarization on CNN/DAILYMAIL (Nallapati et al., 2016); and (vi) vision–language tasks, including POPE (Li et al., 2023a), Blink Art Style, Blink Forensic Detection, and Blink Counting (Fu et al., 2024). All tasks are evaluated in the zero-shot setting without task-specific fine-tuning. For a more controlled analysis, the main-paper results apply reduction to attention-side matrix multiplications. Full detailed result tables are provided in the appendix.

Baselines. We compare RMM against representative pruning and inference-time optimization baselines. **Static pruning methods** include SparseGPT (Frantar and Alistarh, 2023), Wanda (Sun et al., 2024), SliceGPT (Ashkboos et al., 2024), and magnitude pruning (Han et al., 2015). For **dynamic inference-time baselines**, we include H2O (Zhang et al., 2023), which dynamically manages the KV cache during decoding but does not modify the feature dimensions involved in matrix multiplications. As control baselines, we also include a **static variant of RMM**, which selects a fixed subset of feature dimensions during prefill based on activation statistics and reuses the same subset for all subsequent tokens. And **random pruning** retains feature dimensions uniformly at random at each decoding step under the same sparsity budget as RMM. Together, these baselines allow us to compare RMM against static pruning, non-adaptive activation-based selection, random selection, and dynamic methods operating at different levels of the inference stack.

5 Main Results

5.1 Controlled comparison of Pruning behavior

We first compare pruning strategies on a single model under a controlled setting to isolate the effect of different pruning strategies. All experiments in this stage are conducted on **LLaMA 3.1 8B** with a fixed retention ratio of $\mathrm{RR} = 0.5$. We consider two inference settings:

Discriminative question answering We first evaluate zero-shot QA performance on standard reasoning benchmarks (Table 1). Performance is measured by comparing the log-likelihoods of candidate answers. We compare RMM against representative static pruning baselines, including SparseGPT, Wanda, SliceGPT, and magnitude pruning. RMM achieves the best average accuracy among all pruning methods and shows more consistent degradation across tasks. In contrast, static baselines suffer substantially larger and less uniform drops across benchmarks. These results suggest that fixed, non-adaptive pruning decisions are insufficient to maintain stable performance in practical downstream tasks.

Abstractive summarization We further evaluate pruning strategies on the abstractive summarization task, a token-by-token generation setting (Table 2). In addition to the static baselines, we include two dynamic baselines in this setting: (i) random pruning, which selects retained dimensions uniformly at random at each decoding step, and (ii) H2O, which dynamically manages the KV cache at the token level. All methods are evaluated under identical retention ratios. At $\mathrm{RR} = 0.8$, RMM remains close to the full model while substantially reducing computation. When the retention ratio is reduced to $\mathrm{RR} = 0.5$, RMM continues to outperform all baselines by a clear margin. Static pruning methods degrade rapidly in generation quality, while random pruning, despite being dynamic, fails to maintain coherent and semantically consistent summaries. H2O performs better than static pruning in this setting, but remains consistently weaker than activation-aware matrix-level pruning.

Summary Under a fixed model and retention ratio, the relative behavior of pruning strategies differs markedly between discriminative and generative inference settings. Static and activation-agnostic methods exhibit less stable degradation, while RMM maintains a clear advantage across both settings. Additional detailed results are reported in Table 10.

5.2 Scaling across models and retention ratios

We next examine how pruning tolerance changes with model scale under different retention ratios. We evaluate RMM on a range of model sizes. For each model, we sweep the retention ratio over $\mathrm{RR} \in \{0.9, 0.8, 0.7, 0.6, 0.5\}$. We use the same zero-shot discriminative evaluation setup, covering commonsense and reasoning benchmarks as well as more structured tasks including GSM8K, MMLU, and HumanEval.

Table 3 summarizes the results across models and retention ratios. At matched retention levels, larger models consistently retain stronger task performance. For example, at $\mathrm{RR} = 0.8$, LLaMA 3.1 70B remains close to the full model on most benchmarks, whereas smaller models show more pronounced degradation, especially on challenging tasks such as GSM8K and HumanEval. As the retention ratio decreases further, all models degrade, but smaller models exhibit an earlier performance inflection—with noticeable drops already at $\mathrm{RR} = 0.7$—while larger models degrade more gradually and maintain higher absolute accuracy even under aggressive pruning ($\mathrm{RR} \leq 0.6$). Overall, these results reveal a clear scaling trend:

Method (RR = 0.5)	ARC-C	ARC-E	COPA	PIQA	CommQA	Avg
Full model (RR = 1.0)	49.50	76.32	77.20	79.92	66.01	69.79
RMM	36.80	63.00	70.60	76.60	51.90	59.78
SparseGPT	31.44	64.21	70.40	70.95	43.41	56.08
Wanda	28.09	60.70	67.20	68.88	38.41	52.66
SliceGPT	20.74	31.75	56.00	53.43	23.10	37.00
Magnitude	22.74	33.68	57.20	57.56	25.06	39.25

Table 1: Zero-shot QA performance on LLaMA 3.1 8B under a fixed retention ratio ($\mathrm{RR} = 0.5$). All pruning methods operate at the same retention ratio, while the full model ($\mathrm{RR} = 1.0$) is shown for reference.

Model	Method	RR	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-Lsum	BERTScore
LLaMA 3.1 8B	Full model (RR = 1.0)	–	37.44	15.56	24.31	31.29	86.76
	RMM	0.8	37.54	15.70	24.23	31.35	86.72
	RMM	0.5	**34.15**	**13.60**	**22.03**	**28.70**	**85.75**
	Static	0.8	37.36	15.63	24.24	31.24	86.67
	Static	0.5	28.01	9.85	19.28	24.34	84.02
	Random	0.8	6.90	0.60	6.20	6.69	77.97
	Random	0.5	5.67	0.20	5.23	5.54	81.42
	H2O	0.8	24.38	9.26	16.28	21.88	82.72
	H2O	0.5	24.38	9.26	16.28	21.88	82.72

Table 2: Abstractive summarization performance on CNN/DailyMail using LLaMA 3.1 8B under different pruning strategies. All pruning methods are evaluated under the same retention ratios. H2O maintains a fixed token budget and therefore yields identical results across retention ratios.

larger models tolerate more aggressive pruning at the same retention ratio. Additional results and alternative models are provided in Appendix B.

5.3 Stability and robustness under generation and long-context settings

While the previous sections focus on discriminative benchmarks, practical inference also requires stable long-form generation and reliable reasoning over extended contexts. We therefore further evaluate RMM under both generative and long-context settings to assess whether dynamic pruning remains stable beyond discriminative task evaluation.

Qualitative generation behavior We next examine how pruning affects autoregressive generation. Table 5 shows representative outputs from the full model and RMM under different retention ratios. At $\mathrm{RR} = 0.8$, the outputs of RMM remain largely consistent with those of the full model in both semantic content and overall structure. At $\mathrm{RR} = 0.6$, the generated text begins to show simplification and stylistic drift, but remains coherent and semantically aligned with the prompt. Notably, degradation remains gradual and predictable as pruning becomes more aggressive, rather than manifesting as abrupt or unstable failure.

Long-context reasoning We further evaluate RMM on long context benchmarks to test whether pruning remains reliable over long sequences. As shown in Table 4, RMM achieves performance comparable to the full model across all tested context lengths at both RR . We do not observe a systematic increase in degradation as context length grows, suggesting that dynamic pruning does not disproportionately impair long-range dependency modeling within this regime.

Summary Across generative and long-context settings, RMM remains robust across a wide range of retention ratios, with degradation that is smooth rather than abrupt under stronger pruning. These results show that the benefits of dynamic pruning are not limited to discriminative benchmarks, but also extend to more realistic inference scenarios.

5.4 Vision-language generalization

We further evaluate whether RMM generalizes beyond text-only models by testing it on the vision–language model. As shown in Table 6, the core trend observed in text-only models also holds in the multimodal setting: RMM remains close to the full model under mild pruning and continues to outperform static and random pruning under more aggressive reduction. At $\mathrm{RR} = 0.8$, RMM achieves performance nearly identical to the full model across all benchmarks. At $\mathrm{RR} = 0.5$, it still retains strong performance and substantially outperforms both static and random pruning under the same retention ratio. Figure 2 provides qualitative support by visualizing the attention maps for the first output token. Both the full model and RMM attend to semantically relevant visual regions, whereas static pruning under-attends to these regions and random pruning produces scattered, less focused patterns. These qualitative observations are consistent with the quantitative results in Table 6, showing that RMM better preserves the visual grounding needed

Model	Method	RR	Copa	ARC-C	ARC-E	PiQA	CommQA	GSM8K	MMLU	HumanEval
Qwen3.1 7B	Baseline	–	72.8	39.8	69.82	72.69	47.58	39.87	55.54	40.24
	RMM	0.9	66.0	33.78	64.56	69.37	46.65	24.93	52.73	39.64
	RMM	0.8	64.0	29.77	57.19	67.19	47.58	17.58	47.30	38.41
	RMM	0.7	60.8	28.09	46.84	63.49	39.64	5.83	33.55	31.09
	RMM	0.6	58.4	25.42	37.02	59.58	29.89	2.50	25.97	20.73
	RMM	0.5	49.4	20.74	32.11	53.16	24.49	1.70	23.78	9.75
Llama3.1 8B	Baseline	–	77.2	49.5	76.32	79.92	66.01	26.15	63.48	35.36
	RMM	0.9	76.6	48.16	75.26	79.16	65.44	24.64	62.20	35.36
	RMM	0.8	77.2	47.49	75.09	79.05	64.70	23.73	60.26	34.75
	RMM	0.7	77.0	46.82	72.81	77.48	62.65	23.19	55.24	32.31
	RMM	0.6	73.4	37.46	68.60	77.48	59.38	14.93	38.62	26.21
	RMM	0.5	70.6	36.79	62.98	76.65	51.92	5.91	24.78	23.17
Qwen3 32B	Baseline	–	81.4	57.86	78.25	80.89	61.59	62.62	80.81	37.80
	RMM	0.9	83.6	55.18	76.14	80.74	62.24	62.69	80.01	40.24
	RMM	0.8	83.2	51.84	72.63	80.20	61.02	57.99	78.64	42.07
	RMM	0.7	82.6	48.49	69.47	80.58	60.61	55.11	77.50	42.68
	RMM	0.6	82.6	50.84	70.53	80.36	58.72	50.87	73.17	45.12
	RMM	0.5	82.2	46.15	67.19	77.80	54.87	39.87	65.09	46.10
Llama3.1 70B	Baseline	–	84.4	56.19	78.25	83.24	57.99	53.67	75.29	51.21
	RMM	0.9	84.4	54.52	78.77	83.51	57.99	51.47	75.04	53.65
	RMM	0.8	84.6	56.86	76.84	82.59	59.71	48.14	72.58	46.95
	RMM	0.7	81.4	53.18	74.74	82.48	59.87	42.75	67.01	39.63
	RMM	0.6	76.6	50.50	74.91	77.69	60.44	34.49	53.81	34.75
	RMM	0.5	70.2	41.47	64.21	73.61	56.84	19.93	29.73	18.90

Table 3: Performance comparison across different models and retention ratios on various benchmarks.

Task	Cond.	5K	15K	30K
CWE	Base	98.17	94.03	29.59
	0.8	98.11	94.09	29.30
	0.5	97.97	94.01	28.93
Hotpot	Base	53.6	56.4	51.2
	0.8	53.6	55.8	50.8
	0.5	53.6	55.6	50.5

Table 4: Long-context results on Ruler.

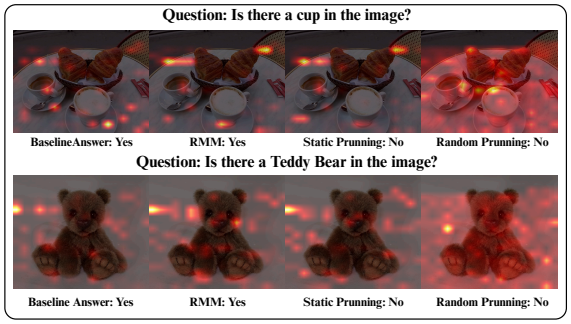

Figure 2: Attention map from the vision language model.

for multimodal inference.

6 Ablation and Mechanistic Analysis

6.1 Validating the key design choices of RMM

We validate the two key design choices of RMM: dynamic selection(Table 2) and activation-aware scoring(Table 6). Rather than relying on a single-task ablation, we draw evidence from comparisons across multiple tasks. The importance of dynamic selection is reflected in the consistent gap between RMM and its static variant, which fixes the retained feature subset across tokens. It widens substantially at more aggressive retention ratios, as seen in both summarization and vision–language tasks. This suggests that adapting the retained dimensions to input-dependent activation changes is critical for preserving performance under stronger pruning. The importance of activation-aware scoring is supported by comparisons against random pruning. Although random pruning also changes the retained subset dynamically, it performs much worse under the same retention ratios. This shows that the gains of RMM do not arise merely from changing the retained subset over time, but from selecting informative dimensions according to the current activation pattern. Taken together, these results show that both dynamic selection and activation-aware scoring are essential to the effectiveness of RMM.

6.2 Different components Mechanistic Analysis

We next examine where pruning can be applied most safely within the Transformer. We analyze the sensitivity of different modules to reduction on LLaMA 3.1 8B by separately pruning attention-side components (Q projection, QKV projection, and attention output), MLP-side components (up, gate, and down projections), and their combinations. Due to space limitations, we report the full pruning matrix in Appendix Table 11. Across all settings, attention-side computations are substantially more robust to pruning than MLP-side components. When pruning only attention-related operations, performance degrades gradually as the retention ratio decreases and remains close to the baseline even at moderate reduction levels. In contrast, pruning MLP components leads to much sharper

	Prompt	Base Model	RMM Inference	
			RR 0.8	RR 0.6
Ex.1	The future of artificial intelligence is	here, and it's already changing the way we live and work. From self-driving cars to virtual assistants.	here, and it's already changing the way we live and work. From self-driving cars to virtual assistants.	here. It's called ChatGPT. This AI chatbot can write essays, poems, and even code.
Ex.2	Tell me something about Boston.	Boston is a city in the state of Massachusetts, in the United States of America.	Boston is a city in the state of Massachusetts, in the United States of America.	I'm a native New Yorker, and I've been here for 10 years.

Table 5: Generation comparison between base model (Llama-3.1-8B) and RMM inference under different retention ratios.

Method	RR	Pope	Art.	Foren.	Count.
Baseline	–	83.67	100	100	100
RMM	0.8	**82**	**100**	**100**	**100**
Static	0.8	81	100	100	100
Random	0.8	10.3	43.59	58.33	54.17
RMM	0.5	**67.33**	**97.44**	**97.73**	**99.17**
Static	0.5	63	91.29	43.33	79.17
Random	0.5	1.3	41.88	53.03	43.33

Table 6: Performance on Qwen 2.5-VL-7B.

L	Op.	Dense	RMM	Speedup
1024	$QK^\top$	0.120	0.089	1.36×
1024	AV	0.065	0.039	1.67×
2048	$QK^\top$	0.433	0.336	1.29×
2048	AV	0.207	0.114	1.81×
4096	$QK^\top$	1.675	1.071	1.56×
4096	AV	0.753	0.399	1.89×

Table 7: GEMM kernel latency (ms).

Seq. Len.	Dense	RMM	Speedup
1024	109.39	103.91	1.05×
2048	264.67	208.93	1.27×
4096	661.36	473.21	1.40×

Table 8: End-to-end latency (ms).

performance dropping across tasks. In particular, pruning the entire MLP block leads to severe performance collapse even at relatively high retention ratios, showing the structural importance of MLP layers for preserving representation capacity. By contrast, different attention-side components exhibit greater functional redundancy: pruning their combinations leads to more moderate degradation. These results reveal a clear structural asymmetry within the Transformer: attention-side computations contain higher redundancy, whereas MLP components are more rigid and harder to prune. This suggests that practical deployments should prioritize pruning attention modules and apply more conservative reduction to the MLP.

6.3 Practical Viability: Wall-Clock Latency

To evaluate whether the computational savings of RMM translate into practical runtime gains, we measure both kernel-level and end-to-end wall-clock latency on LLaMA 3.1 8B using an NVIDIA A100 GPU with a batch size of 1 and a retention ratio of $\rho = 0.8$. All latency numbers are averaged over 10 runs. We follow the latency evaluation protocol of (Sun et al., 2024). For kernel-level benchmarks (Table 7), we measure the full cost of each RMM operation, including norm computation, top-k selection, and the reduced matrix multiplication, to verify that the selection overhead does not offset the computational savings. For end-to-end evaluation (Table 8), the dense baseline uses HuggingFace `generate` with SDPA as the attention backend, and the RMM variant replaces the attention kernels with custom Triton implementations. Overall, these results show that the computational savings of RMM can translate into tangible runtime benefits, especially when sequence lengths are sufficiently large.

7 Conclusion

We introduced *Reduced Matrix Multiplication* (RMM), a training-free and input-adaptive method that reduces Transformer inference computation by selecting informative indices along the shared multiplication axis of each matrix product. Across model scales and diverse tasks, RMM yields controllable accuracy–efficiency trade-offs, remains stable in generative and long-context settings, and extends to vision–language inference. Our analyses further reveal that attention-side computations are substantially more reducible than MLP components, and wall-clock benchmarks show that these reductions can translate into practical runtime gains. More broadly, these results highlight that redundancy in Transformer inference is not uniformly distributed across components, suggesting that efficient inference methods should account for such structural differences. These findings suggest that matrix-product-level adaptive reduction is a promising direction for efficient Transformer inference.

Limitations

This work focuses on a training-free formulation of input-adaptive matrix-level reduction for Transformer inference. Our emphasis is on establishing the core algorithmic idea and evaluating it in a controlled setting. We have not explored extensions beyond the training-free setting considered here. Another promising direction concerns multimodal models. While our results suggest that the framework is applicable beyond pure language modeling, we have not conducted a detailed study of how dynamic reduction behaves across the different components of vision–language models. It would be valuable to test whether their components exhibit different redundancy patterns under dynamic pruning.

References

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicegpt: Compress large language models by deleting rows and columns. *Preprint*, arXiv:2401.15024.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

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, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *Preprint*, arXiv:1803.05457.

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

Petros Drineas, Ravi Kannan, and Michael W Mahoney. 2006. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM Journal on computing*, 36(1):158–183.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot. *Preprint*, arXiv:2301.00774.

Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A. Smith, Wei-Chiu Ma, and Ranjay Krishna. 2024. Blink: Multimodal large language models can see but not perceive. *Preprint*, arXiv:2404.12390.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. 2025. Deep think with confidence. *Preprint*, arXiv:2508.15260.

Shangqian Gao, Chi-Heng Lin, Ting Hua, Tang Zheng, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. 2024. Disp-llm: Dimension-independent structural pruning for large language models. *Preprint*, arXiv:2410.11988.

Andrew Gordon, Zornitsa Kozareva, and Melissa Roemmele. 2012. SemEval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 394–398, Montréal, Canada. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. Learning both weights and connections for efficient neural networks. *Preprint*, arXiv:1506.02626.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? *Preprint*, arXiv:2404.06654.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Donghyun Lee, Je-Yong Lee, Genghan Zhang, Mo Tiwari, and Azalia Mirhoseini. 2024. Cats: Contextually-aware thresholding for sparsity in large language models. *Preprint*, arXiv:2404.08763.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. 2023a. Evaluating object hallucination in large vision-language models. *Preprint*, arXiv:2305.10355.

Yucheng Li, Bo Dong, Chenghua Lin, and Frank Guerin. 2023b. Compressing context to enhance inference efficiency of large language models. *arXiv preprint arXiv:2310.06201*.

Andy T. Liu, Shang-Wen Li, and Hung-yi Lee. 2021. Tera: Self-supervised learning of transformer encoder representation for speech. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2351–2366.

James Liu, Pragaash Ponnusamy, Tianle Cai, Han Guo, Yoon Kim, and Ben Athiwaratkun. 2025. Training-free activation sparsity in large language models. *Preprint*, arXiv:2408.14690.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *Preprint*, arXiv:2305.11627.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *Preprint*, arXiv:1609.07843.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. *Preprint*, arXiv:2403.12968.

Yifan Peng, Kwangyoun Kim, Felix Wu, Prashant Sridhar, and Shinji Watanabe. 2023. Structured pruning of self-supervised pre-trained models for speech recognition and understanding. *Preprint*, arXiv:2302.14132.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2023. On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. 2024. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *Preprint*, arXiv:2407.08608.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024. A simple and effective pruning approach for large language models. *Preprint*, arXiv:2306.11695.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *Preprint*, arXiv:1811.00937.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. 2025. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *Preprint*, arXiv:2502.11089.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, and 1 others. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Technical Appendices

A Implementation Details

Hardware and Framework. All experiments were run on NVIDIA GPUs (a mix of RTX A6000, RTX 6000 Ada, A100, and L40S). Inference was implemented in PyTorch using HuggingFace Transformers, with bf16 precision unless otherwise specified. We use the official pre-trained checkpoints without any fine-tuning or additional training. We write the attention and mlp module by ourselves and use monkey patch to replace the original module.

Inference Setup. For our experiments, a global Retention Ratio (RR) was specified (e.g., from 0.9 to 0.5). RMM dynamically recomputes the retained subspace at every layer and token, applied directly within the forward pass of attention and MLP blocks. Batch sizes were chosen to fit GPU memory (typically 8–16 for 7B/8B models, 1–2 for 70B models). For long-context benchmarks such as RULER, we evaluate up to the maximum sequence length provided by the dataset (up to 30K tokens).

Metrics. Perplexity is reported for language modeling tasks (WIKITEXT, BOOKCORPUS). For QA and reasoning benchmarks (COPA, PIQA, COMMONSENSEQA, MMLU, etc.), we follow the standard `lm-eval-harness` framework, running all tasks in the zero-shot setting with default accuracy as the evaluation metric. Rouge-1/2/L and BERTScore are reported for summarization (CNN/DAILYMAIL), and task-specific metrics are used for multimodal benchmarks (POPE, BLINK ART STYLE, BLINK FORENSIC DETECTION, BLINK COUNTING). All evaluations are performed in the zero-shot setting without task-specific adaptation.

A.1 Why Dynamic Pruning

A natural starting point for reducing inference cost is *static* pruning: one identifies a fixed subset of weights or feature dimensions by analyzing model behavior on a reference dataset, and then permanently removes the remaining components. This paradigm has inspired many existing methods, and we initially explored similar directions. From a systems perspective, static pruning is particularly attractive: if a fixed low-dimensional subspace can be identified during prefilling, the same subspace could be reused during decoding, allowing a reduced KV cache and significantly lower memory bandwidth and I/O overhead.

However, our empirical investigations revealed that this approach does not generalize. When a static subspace is derived from a specific dataset (e.g., WikiText), the resulting pruned model often performs well on that dataset, but degrades sharply when evaluated on other tasks or distributions. We observed similar behavior when attempting to identify "unimportant" dimensions within attention heads: while certain dimensions appear consistently inactive for a given task, such patterns are not stable across tasks, layers, or inputs.

These findings suggest that high-dimensional representations in Transformers do not admit a single, globally valid low-dimensional subspace. Instead, the semantic information encoded in hidden states appears to migrate across dimensions depending on the input, the layer, and the decoding step. Consequently, any fixed pruning rule implicitly assumes a static allocation of information, and must either sacrifice generalization or incur severe accuracy loss under aggressive pruning.

Even with dynamic selection, we observe that sufficiently aggressive pruning (e.g., 50% retention) can still lead to performance degradation on certain tasks. This highlights a broader limitation of training-free acceleration: it is unlikely that one can simultaneously achieve large speedups, no retraining, and negligible accuracy loss in all settings. Nevertheless, our results provide strong evidence that substantial redundancy exists in large language models, and that *dynamic, input-adaptive subspace selection* is a principled way to exploit this redundancy without assuming task-specific structure.

More broadly, these observations point to a promising research direction: incorporating compressibility and adaptive computation directly into the pretraining objective, rather than relying solely on post-hoc pruning or scaling. RMM represents an initial step toward this goal by demonstrating that dynamic subspace computation can serve as a viable abstraction for inference-time efficiency.

A.2 A Speculative Theoretical Perspective on Dynamic Subspaces

The results in Section A.1 indicate that no single, fixed low-dimensional subspace can consistently approximate Transformer activations across different inputs, layers, and decoding steps. Here, we offer a speculative theoretical perspective that may help explain this phenomenon. We do not claim formal guarantees; rather, the following discussion is intended as an interpretive framework grounded in existing principles of representation learning and high-dimensional geometry.

Let $h_t^\ell \in \mathbb{R}^d$ denote the hidden state at layer ℓ and time step t. Static pruning implicitly assumes the existence of a global subspace $U \subset \mathbb{R}^d$ such that $h_t^\ell \approx \Pi_U(h_t^\ell)$ for all (t, ℓ) and all inputs. This corresponds to assuming a single low-rank structure shared across all inference contexts.

Our observations suggest that such a global subspace does not exist. Instead, the effective representation subspace appears to vary with the input, the layer, and the decoding step. One possible explanation is that semantic information is not tied to fixed coordinate axes, but is encoded in a *distributed* manner across dimensions. As a result, the set of dimensions that carry the most information may change from one context to another.

A complementary perspective arises from viewing deep Transformers as nonlinear dynamical systems, where each layer applies an input-dependent transformation, $h^{\ell+1} = f^\ell(h^\ell)$. Under this view, imposing a fixed low-dimensional projection at intermediate layers amounts to injecting a structured perturbation into the system. Because the subsequent transformations are nonlinear, small projection errors can be amplified across layers, causing deviations that depend strongly on the input trajectory.

From a geometric viewpoint, one may imagine that the activations for a given input lie near a low-dimensional manifold whose local tangent space changes across inputs and layers. Static pruning corresponds to projecting onto a single global linear subspace, while RMM instead performs a *local, input-dependent projection* that adapts to the current tangent directions. Although this interpretation is heuristic, it provides an intuitive explanation for why dynamic selection generalizes more robustly than static pruning in our experiments.

Taken together, these perspectives suggest that the effectiveness of RMM may stem from aligning computation with the evolving geometry of high-dimensional representations. We view this as an open direction for future work, including the possibility of incorporating adaptive subspace structure directly into model training.

B Supplementary Results

B.1 Additional Results on QA and Language Modeling

To complement the QA and generation analysis in Section 5, we provide full perplexity results on WIKITEXT and BOOKCORPUS for smaller-scale models. In addition to the 7B/8B/32B/70B models reported in the main paper, we include Llama-3.2-1B and Llama-3.2-3B. Figure 3 and Table 9 present the complete results under different retention ratios (RR).

The results show the same trend as larger models: perplexity increases gradually as RR decreases, with a sharp degradation once RR drops below 0.6. Moreover, the 3B model consistently shows greater robustness than the 1B model (Figure 3), reinforcing our claim that representational redundancy grows with scale.

B.2 Additional Results on Summarization

We also expand the summarization results on CNN/DAILYMAIL beyond those in the main paper. Table 10 includes Llama-3.2-1B and 3B alongside the larger 7B/8B models.

RMM matches the baseline at RR = 0.8 across all scales, while static and random pruning degrade severely. At RR = 0.5, smaller models drop more sharply, but RMM remains consistently better than all baselines. This confirms that our method preserves summarization quality even in small-scale models, while redundancy increases with size, enabling more aggressive pruning at larger scales.

RR	1.0	0.9	0.8	0.7	0.6	0.5
WikiText Perplexity ↓						
Llama3.2 1B	20.04	20.66	22.77	31.29	68.52	151.64
Llama3.2 3B	15.89	16.23	17.01	18.82	25.04	42.54
Qwen3.1 7B	28.64	31.49	37.80	54.00	93.52	219.31
Llama3.1 8B	13.39	14.34	15.22	17.03	21.35	32.65
Qwen3 32B	13.97	14.62	15.41	15.71	15.99	18.53
Llama3.1 70B	7.24	7.46	8.38	14.14	42.62	167.78
BookCorpus Perplexity ↓						
Llama3.1 1B	21.18	22.03	24.39	39.63	96.81	192.81
Llama3.2 3B	17.79	18.07	18.86	21.84	34.47	54.53
Qwen3.1 7B	31.95	34.21	43.64	62.91	113.57	322.63
Llama3.1 8B	15.25	15.59	16.48	20.97	32.63	53.80
Qwen3 32B	17.43	17.72	18.12	18.93	22.21	29.68
Llama3.1 70B	12.20	12.29	13.06	19.55	36.55	126.18

Table 9: Perplexity matrix across RR

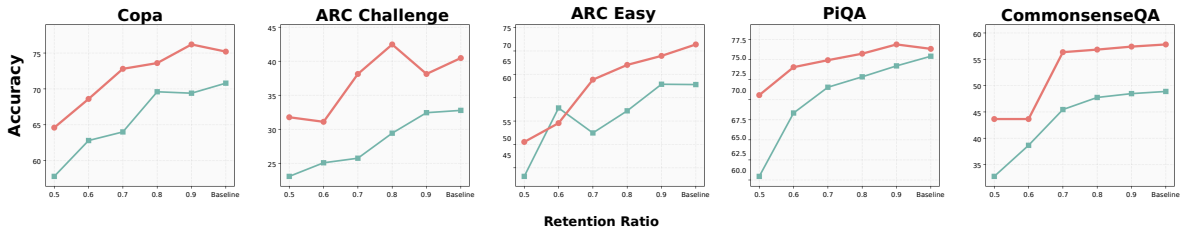

Figure 3: Llama-3.2-3B and Llama-3.2-1B of Different Tasks and RR. Red Line is Llama-3.2-3B

C Ablation Study Details

Sensitivity to different components. We provide comprehensive ablation results on Llama3.1-8B, summarized in Table 11. This analysis examines pruning applied to individual projections (Q, K, V), combinations of QKV and attention computations, and individual MLP submodules (*up*, *gate*, *down*), as well as pruning the entire MLP block or combinations with attention. By presenting the full set of results, we highlight consistent patterns across different targets and retention ratios.

Attention-side pruning. Pruning restricted to attention-related operations (Q-only, QKV projections, or the attention matrix) generally yields moderate accuracy degradation. Performance remains above 90% of the baseline even at RR = 0.7 for most QA tasks, confirming that attention modules contain considerable redundancy. Among these, pruning Q or QKV projections is more stable, while directly pruning the attention matrix leads to steeper drops at lower retention ratios. These results support our claim that attention is a safe target for aggressive pruning.

MLP-side pruning. In contrast, pruning within MLPs has a more severe effect. Removing hidden channels in the *up*, *gate*, or *down* projections degrades performance faster than in attention. The *down* projection is slightly more robust, but once the entire MLP block is pruned, accuracy collapses rapidly: at RR = 0.5, performance drops to nearly 40% of the baseline average. This suggests that MLP computations are less redundant and contribute critically to model accuracy.

Hybrid strategies. When attention and MLP pruning are combined, degradation compounds quickly, with accuracy falling below 50% at RR = 0.6. This indicates that uniformly applying high pruning ratios across all modules is not effective. Instead, hybrid strategies should prioritize pruning attention-side computations more aggressively, while applying more conservative pruning to MLPs.

Takeaways. Overall, the ablations provide three key insights: (i) Attention modules exhibit high redundancy and can sustain substantial pruning with minimal loss. (ii) MLP blocks, while larger in

Model	Method	RR	Rouge-1	Rouge-2	Rouge-L	Rouge-Lsum	BERTScore
Llama3.1 8B	Baseline	–	37.44	15.56	24.31	31.29	86.76
	RMM	0.8	**37.54**	**15.70**	**24.23**	**31.35**	**86.72**
	RMM	0.5	**34.15**	**13.60**	**22.03**	**28.70**	**85.75**
	Static	0.8	37.36	15.63	24.24	31.24	86.67
	Static	0.5	28.01	9.85	19.28	24.34	84.02
	Random	0.8	6.90	0.60	6.20	6.69	77.97
	Random	0.5	5.67	0.20	5.23	5.54	81.42
	H2O	0.8	24.38	9.26	16.28	21.88	82.72
	H2O	0.5	24.38	9.26	16.28	21.88	82.72
Llama3.2 1B	Baseline	–	36.51	15.24	23.55	30.59	86.41
	RMM	0.8	**37.27**	**15.71**	**23.62**	**30.96**	**86.35**
	RMM	0.5	**11.90**	**2.35**	**9.80**	**11.15**	**79.55**
	Static	0.8	36.77	15.36	23.39	30.68	86.35
	Static	0.5	4.73	0.07	4.26	4.58	75.53
	Random	0.8	7.30	0.02	6.58	7.08	79.15
	Random	0.5	6.26	0.40	5.53	6.01	76.85
	H2O	0.8	23.22	8.12	15.56	20.30	83.76
	H2O	0.5	23.22	8.12	15.56	20.30	83.76
Llama3.2 3B	Baseline	–	36.72	15.08	23.56	30.63	86.55
	RMM	0.8	**36.72**	**15.16**	**23.58**	**30.66**	**86.54**
	RMM	0.5	**28.31**	**9.83**	**19.45**	**24.60**	**84.74**
	Static	0.8	35.96	14.81	23.24	30.13	86.31
	Static	0.5	19.91	6.06	14.69	17.83	82.06
	Random	0.8	6.67	0.39	5.93	6.44	79.72
	Random	0.5	5.85	0.04	5.33	5.70	77.09
	H2O	0.8	19.49	6.32	13.83	17.03	82.78
	H2O	0.5	19.49	6.32	13.83	17.03	82.78
Qwen3-1 7B	Baseline	–	36.56	13.05	22.86	29.75	85.91
	RMM	0.8	**35.32**	**12.08**	**22.31**	**28.99**	**86.81**
	RMM	0.5	**21.91**	**5.89**	**14.54**	**18.60**	**83.33**
	Static	0.8	34.43	11.63	22.05	28.47	86.67
	Static	0.5	4.12	0.05	3.79	4.01	78.09
	Random	0.8	10.23	0.03	8.04	9.66	76.31
	Random	0.5	6.75	0.06	6.09	6.54	76.46
	H2O	0.8	4.20	9.10	3.68	3.98	73.46
	H2O	0.5	4.20	9.10	3.68	3.98	73.46

Table 10: Performance comparison of different pruning methods on CNN summarization task. RMM consistently outperforms baseline methods across different models and retention ratios.

parameter count, are more sensitive to pruning, particularly when entire blocks are reduced. (iii) Effective deployment should adopt heterogeneous pruning policies that allocate higher budgets to attention and lower budgets to MLPs. These findings reinforce the general message of our method: redundancy exists across multiple components, but its distribution is uneven, and exploiting this structure is critical for robust efficiency gains.

D Efficiency Analysis

D.1 Complexity Analysis

We provide a detailed analysis of how RMM affects the two dominant regimes of Transformer inference: (i) attention and (ii) MLP computation. Notation: let N be the sequence length, D the head dimension, d_v the value dimension, and m the intermediate hidden width of the MLP. For retention ratios we write $\rho_{\text{feat}} = K/D$ for features, $\rho_{\text{tok}} = \ell/N$ for tokens, and $\rho_{\text{hid}} = r/m$ for hidden channels.

(i) Attention. Self-attention consists of two main multiplications: the score matrix $QK^\top$ and the value aggregation $\mathrm{softmax}(QK^\top)V$. Computing $QK^\top$ over N tokens scales as $O(N^2D)$. Restricting to $\rho_{\text{feat}}D$ features reduces this to $O(\rho_{\text{feat}}N^2D)$, with both FLOPs and activation traffic reduced proportionally to ρ_{feat}. For value aggregation, the cost drops from $O(N^2d_v)$ to $O(N\ell d_v) = O(\rho_{\text{tok}}N^2d_v)$ by keeping only $\rho_{\text{tok}}N$ keys per query. Thus, attention complexity scales down multiplicatively with $(\rho_{\text{feat}}, \rho_{\text{tok}})$.

During autoregressive decoding at step t, each new token attends to t cached keys and values. The score computation reduces from $O(tD)$ to $O(\rho_{\text{feat}}tD)$. Likewise, reading V vectors reduces from $O(td_v)$ to $O(\rho_{\text{tok}}td_v)$. This lowers both FLOPs and memory traffic, while KV-cache reads shrink by ρ_{tok}—a critical factor since cache bandwidth often dominates latency in long-context decoding.

(ii) MLPs. A standard two-layer feed-forward network with hidden width m has cost $O(NDm+NmD)$. Applying RMM to retain only $\rho_{\text{hid}}m$ hidden channels reduces this to $O(\rho_{\text{hid}}NDm)$, saving computation

linearly with ρ_{hid}. Importantly, since the same channel subset is used in the *gate*, *up*, and *down* projections, the pruning applies consistently across all multiplications without introducing auxiliary parameters or kernels. This yields reductions in both compute and memory proportional to ρ_{hid}.

Comparison to other pruning strategies. Unlike token-level pruning (which reduces sequence length) or weight pruning (which reduces parameters), RMM targets the *matrix multiplications themselves*. This is the atomic operation underlying both attention and MLP blocks, and thus improvements directly translate to reductions in FLOPs, memory movement, and cache bandwidth. Since FLOP count, memory traffic, and cache I/O all scale with $(\rho_{\text{feat}}, \rho_{\text{tok}}, \rho_{\text{hid}})$, RMM exposes a smooth tradeoff between accuracy and efficiency that can be tuned by a single retention ratio.

Practical implications. In our implementation, reductions are realized at the level of activation selection. If integrated with custom CUDA kernels that skip the pruned dimensions (rather than simply masking after multiplication), the theoretical savings above can directly yield wall-clock speedups and memory reductions. This means that RMM, once implemented natively, can achieve real acceleration comparable to other efficiency methods such as FlashAttention or block-sparse kernels.

Summary. Across both attention and MLPs, RMM consistently reduces FLOPs, activation traffic, and cache bandwidth in direct proportion to the retention ratios. Because the method operates deterministically and input-adaptively, it provides fine-grained control of the accuracy–efficiency frontier and is fully compatible with future kernel-level implementations that can convert the savings into end-to-end inference speedups.

E Theoretical Analysis of RMM

We provide theoretical justification for the design of RMM. We first establish that TopK selection by column norm is minimax optimal under a natural constraint (Section E.1), then derive the approximation error bound (Section E.2).

E.1 Minimax Optimality of Activation-Aware Selection

In Transformer inference, the activation matrix A is observed, while the other operand B is not known at selection time. We show that under this information asymmetry, TopK selection by column norm is the optimal dimension selection strategy.

We decompose the matrix product AB as a sum of rank-one terms over the shared dimension:

$$AB = \sum_{j=1}^{d} A_{:,j}\, B_{j,:}. \tag{1}$$

Each term $A_{:,j}\, B_{j,:}$ contributes independently along dimension j, with contribution magnitude $\|A_{:,j}\, B_{j,:}\|_F = \|A_{:,j}\|_2\, \|B_{j,:}\|_2$. When we discard dimension j, the error contribution from that dimension is $\|A_{:,j}\|_2\, \|B_{j,:}\|_2$. Since we observe A but not B, we formulate the selection problem as a minimax game over independent per-dimension adversaries.

Theorem 1 (Minimax Optimality)**.** *Let $A \in \mathbb{R}^{n\times d}$ be a fixed activation matrix, and let $k \in \{1, \dots, d\}$. Define*

$$\mathcal{I}^* = \arg\min_{|\mathcal{I}|=k} \max_{\substack{b_j \geq 0,\ j=1,\dots,d \\ \sum_{j=1}^{d} b_j^2 \leq 1}} \sum_{j \notin \mathcal{I}} \|A_{:,j}\|_2\, b_j, \tag{2}$$

where $b_j = \|B_{j,:}\|_2$ represents the unknown row energy of B. Then $\mathcal{I}^* = \mathrm{TopK}\big(\{\|A_{:,j}\|_2\}_{j=1}^{d},\ k\big)$.

Proof. By Proposition 1, the approximation error satisfies

$$\|AB - A_{:,\mathcal{I}} B_{\mathcal{I},:}\|_F \ \leq\ \sum_{j \notin \mathcal{I}} \|A_{:,j}\|_2\, \|B_{j,:}\|_2. \tag{3}$$

This upper bound depends on B only through the row norms $b_j = \|B_{j,:}\|_2$. We therefore analyze the minimax problem over this upper bound, which yields a valid (and tight) surrogate for the original problem.

Denote $\alpha_j = \|A_{:,j}\|_2$ and $\bar{\mathcal{I}} = [d] \setminus \mathcal{I}$. For fixed $\mathcal{I}$, the inner maximization is

$$\max_{\substack{b_j \geq 0 \\ \sum_j b_j^2 \leq 1}} \sum_{j \in \bar{\mathcal{I}}} \alpha_j \, b_j. \tag{4}$$

By the Cauchy–Schwarz inequality, $\sum_{j\in\bar{\mathcal{I}}} \alpha_j \, b_j \leq \big(\sum_{j\in\bar{\mathcal{I}}} \alpha_j^2\big)^{1/2} \big(\sum_{j\in\bar{\mathcal{I}}} b_j^2\big)^{1/2}$, with equality when $b_j \propto \alpha_j$ for $j \in \bar{\mathcal{I}}$ and $b_j = 0$ for $j \in \mathcal{I}$. Since the adversary can place all energy on $\bar{\mathcal{I}}$ and set $\sum_{j\in\bar{\mathcal{I}}} b_j^2 = 1$, the inner maximum equals

$$\left(\sum_{j \in \bar{\mathcal{I}}} \|A_{:,j}\|_2^2 \right)^{1/2} = \|A_{:,\bar{\mathcal{I}}}\|_F. \tag{5}$$

The outer minimization then becomes

$$\min_{|\mathcal{I}|=k} \|A_{:,\bar{\mathcal{I}}}\|_F = \min_{|\mathcal{I}|=k} \left(\sum_{j \notin \mathcal{I}} \|A_{:,j}\|_2^2 \right)^{1/2}. \tag{6}$$

Minimizing $\sum_{j\notin\mathcal{I}} \|A_{:,j}\|_2^2$ over all subsets $\mathcal{I}$ of size k is equivalent to maximizing $\sum_{j\in\mathcal{I}} \|A_{:,j}\|_2^2$, which is achieved by selecting the k dimensions with the largest column norms:

$$\mathcal{I}^* = \mathrm{TopK}\big(\{\|A_{:,j}\|_2\}_{j=1}^d, \, k\big). \tag{7}$$

□

Remark 1 (Tightness). The inner maximization (5) is achieved by setting $b_j = \alpha_j / \|A_{:,\bar{\mathcal{I}}}\|_F$ for $j \in \bar{\mathcal{I}}$ and $b_j = 0$ otherwise. This corresponds to a matrix B whose row norms are proportional to the activation column norms in the discarded dimensions—precisely the worst case for any fixed selection. TopK selection minimizes the impact of this worst case.

Remark 2 (Interpretation). Theorem 1 states that among all deterministic selection rules that observe only A and retain k dimensions, TopK by column norm minimizes the worst-case error bound. The result holds for any B and does not require assumptions on the structure of B. This provides a principled justification for the design of RMM: the selection rule is the provably optimal choice under the information asymmetry inherent in Transformer inference.

Remark 3 We note that the optimality established here is with respect to selection rules that depend only on A. If B were also observable at selection time, a jointly optimal rule could achieve lower error. However, in Transformer inference, the selection must be made before the matrix multiplication is executed, making the one-sided setting the natural formulation.

E.2 Approximation Error Bound

Proposition 1 (RMM Approximation Error Bound)**.** *Let $A \in \mathbb{R}^{n\times d}$ and $B \in \mathbb{R}^{d\times m}$, and let $\mathcal{I} \subseteq [d]$ with $|\mathcal{I}| = \lceil \rho d \rceil$ be the index set selected by RMM. Denote the complement $\bar{\mathcal{I}} = [d] \setminus \mathcal{I}$. Then the approximation error satisfies*

$$\|AB - A_{:,\mathcal{I}} \, B_{\mathcal{I},:}\|_F \;\leq\; \sum_{j\in\bar{\mathcal{I}}} \|A_{:,j}\|_2 \, \|B_{j,:}\|_2. \tag{8}$$

Proof. The full matrix product decomposes as $AB = \sum_{j=1}^d A_{:,j} \, B_{j,:}$. The RMM approximation retains only the terms indexed by $\mathcal{I}$, so the error is

$$AB - A_{:,\mathcal{I}} \, B_{\mathcal{I},:} = \sum_{j\in\bar{\mathcal{I}}} A_{:,j} \, B_{j,:}. \tag{9}$$

Applying the triangle inequality and $\|uv^\top\|_F = \|u\|_2 \|v\|_2$ yields the bound. □

E.3 Tighter Bound via Cauchy–Schwarz

Corollary 1. *Under the same notation as Proposition 1,*

$$\|AB - A_{:,\mathcal{I}}\, B_{\mathcal{I},:}\|_F \;\le\; \|A_{:,\bar{\mathcal{I}}}\|_F\, \|B_{\bar{\mathcal{I}},:}\|_F. \tag{10}$$

Proof. Applying the Cauchy–Schwarz inequality to the right-hand side of Eq. (8):

$$\sum_{j\in\bar{\mathcal{I}}} \|A_{:,j}\|_2\, \|B_{j,:}\|_2 \;\le\; \left(\sum_{j\in\bar{\mathcal{I}}} \|A_{:,j}\|_2^2\right)^{1/2} \left(\sum_{j\in\bar{\mathcal{I}}} \|B_{j,:}\|_2^2\right)^{1/2} = \|A_{:,\bar{\mathcal{I}}}\|_F\, \|B_{\bar{\mathcal{I}},:}\|_F. \tag{11}$$

□

E.4 Interpretation

The bound in Corollary 1 admits a clear interpretation: the approximation error is controlled by the product of the *residual energy* in the discarded columns of A and the discarded rows of B. Define the *discarded energy ratio* of A as

$$\epsilon_A(\rho) \triangleq \frac{\|A_{:,\bar{\mathcal{I}}}\|_F^2}{\|A\|_F^2}, \tag{12}$$

and analogously $\epsilon_B(\rho)$ for B. Then the relative error satisfies

$$\frac{\|AB - A_{:,\mathcal{I}}\, B_{\mathcal{I},:}\|_F}{\|A\|_F\, \|B\|_F} \;\le\; \sqrt{\epsilon_A(\rho)\, \epsilon_B(\rho)}. \tag{13}$$

This bound, combined with Theorem 1, reveals two key properties:

(1) Energy concentration implies small error. When the activation energy of A is concentrated in a small number of dimensions—as is widely observed in Transformer hidden states—the discarded energy ratio $\epsilon_A(\rho)$ is small even at aggressive retention ratios. For instance, if the top 70% of dimensions capture 95% of the total energy, then $\epsilon_A(0.7) = 0.05$, and the relative error is bounded by $\sqrt{0.05 \cdot \epsilon_B(0.7)}$.

(2) TopK selection minimizes the A-side residual. By Theorem 1, RMM selects the index set that minimizes $\|A_{:,\bar{\mathcal{I}}}\|_F^2$ among all choices of size $\lceil \rho d \rceil$. That is, for any alternative selection $\mathcal{I}'$ with $|\mathcal{I}'| = |\mathcal{I}|$,

$$\|A_{:,\bar{\mathcal{I}}}\|_F^2 \;\le\; \|A_{:,\bar{\mathcal{I}}'}\|_F^2. \tag{14}$$

This means that among all activation-only selection rules at the same retention ratio, RMM achieves the tightest possible bound on the approximation error.

Method	Ratio	Copa	ARC-C	ARC-E	PiQA	CommQA	AVG
Prune Q Projection	Baseline	77.2	49.5	76.32	79.92	66.01	69.79
	0.9	77.4	48.16	76.67	79.6	66.18	69.60
	0.8	77.6	49.83	76.84	79.71	66.42	70.08
	0.7	77.6	50.84	76.67	79.82	65.11	70.01
	0.6	77.2	47.83	76.14	78.94	65.68	69.16
	0.5	77.2	44.15	73.86	79.27	64.54	67.80
Prune QKV Projection	Baseline	77.2	49.5	76.32	79.92	66.01	69.79
	0.9	76.6	48.16	75.26	79.16	65.44	68.92
	0.8	77.2	47.49	75.09	79.05	64.70	68.71
	0.7	77.0	46.82	72.81	77.48	62.65	67.35
	0.6	73.4	37.46	68.60	77.48	59.38	63.26
	0.5	70.6	36.79	62.98	76.65	51.92	59.79
Prune Attention	Baseline	77.2	49.5	76.32	79.92	66.01	69.79
	0.9	78.2	48.49	75.79	79.22	65.57	69.45
	0.8	74.8	47.83	75.44	79.05	64.95	68.41
	0.7	78.0	44.15	70.70	78.89	63.14	66.98
	0.6	76.6	39.13	68.42	78.73	59.71	64.52
	0.5	74.6	32.11	62.11	74.76	54.22	59.56
Prune Attention&Q	Baseline	77.2	49.5	76.32	79.92	66.01	69.79
	0.9	77.4	48.16	76.67	79.60	66.18	69.60
	0.8	77.6	49.83	76.84	79.71	66.42	70.08
	0.7	77.6	50.84	76.67	79.82	65.11	70.01
	0.6	77.2	47.83	76.14	78.94	65.68	69.16
	0.5	77.2	44.15	73.86	79.27	64.54	67.80
Prune Attention&QKV	Baseline	77.2	49.5	76.32	79.92	66.01	69.79
	0.9	76.6	48.20	75.30	79.20	65.40	68.94
	0.8	77.2	47.50	75.10	79.10	64.70	68.72
	0.7	77.0	46.80	72.80	78.10	62.70	67.48
	0.6	73.4	37.50	68.60	77.50	59.40	63.28
	0.5	70.6	36.80	63.00	76.60	51.90	59.78
Prune MLP Up	Baseline	77.2	49.5	76.32	79.92	66.01	69.79
	0.9	73.0	46.49	70.70	78.07	60.20	65.69
	0.8	73.6	38.46	62.98	76.17	58.31	61.90
	0.7	68.2	41.47	60.00	74.59	55.12	59.88
	0.6	69.0	34.55	58.07	73.72	53.15	57.70
	0.5	66.8	28.76	51.05	69.15	46.44	52.44
Prune MLP Gate	Baseline	77.2	49.5	76.32	79.92	66.01	69.79
	0.9	73.6	44.82	74.74	79.33	60.69	66.64
	0.8	74.2	47.16	70.00	77.42	60.03	65.76
	0.7	74.8	41.81	69.12	75.35	59.30	64.08
	0.6	74.4	42.81	66.49	74.27	56.51	62.90
	0.5	69.4	34.11	63.86	71.49	50.61	57.89
Prune MLP Down	Baseline	77.2	49.5	76.32	79.92	66.01	69.79
	0.9	76.8	47.16	73.33	78.84	61.02	67.43
	0.8	76.0	46.49	73.51	78.89	59.71	66.92
	0.7	74.8	45.48	72.81	77.20	58.48	65.75
	0.6	74.0	42.81	66.84	76.71	53.71	62.81
	0.5	72.0	40.80	64.04	76.22	53.73	61.36
Prune Whole MLP	Baseline	77.2	49.5	76.32	79.92	66.01	69.79
	0.9	69.6	43.81	67.02	76.66	58.23	63.06
	0.8	68.6	37.46	61.75	75.39	54.55	59.55
	0.7	70.0	31.77	57.54	71.44	48.89	55.93
	0.6	65.2	25.08	48.95	63.76	42.42	49.08
	0.5	53.6	23.75	35.44	57.56	31.04	40.28
Prune MLP&Attention	Baseline	77.2	49.5	76.32	79.92	66.01	69.79
	0.9	69.6	41.47	65.61	77.15	58.97	62.56
	0.8	69.8	35.12	56.32	73.72	55.86	58.16
	0.7	68.4	33.11	52.28	67.85	48.73	54.07
	0.6	56.6	27.76	44.56	57.51	36.69	44.62
	0.5	56.4	25.08	31.05	52.88	28.01	38.68

Table 11: Comprehensive comparison of different pruning methods on Llama-8B across multiple benchmarks. All methods show similar patterns with gradual performance degradation as the retention ratio decreases.