TransKV: A Data-Driven Pruning Method for Large Foundation Models

Supplementary Material

A.1. Low-rank Properties in ViTs

In this section, we first describe the methodology for obtaining the singular value distributions of both static weight pair (CLOVER) [8] and data-driven weight pair (TransKV). We then present additional examples of singular value distributions from the Vision Transformer (ViT) models [5] across different layers and attention heads.

Static Weight Pair (CLOVER). To compute the singular value distribution for static query-key weight pair, we focus on the pretrained query and key projection matrices (value-output pair follows the similar procedure). Let d denote the hidden dimension and d' the per-head attention dimension. For the i-th attention head, the pretrained query weight matrix $W_Q^i \in \mathbb{R}^{d\times d'}$ and key weight matrix $W_K^i \in \mathbb{R}^{d\times d'}$ are combined along the hidden dimension using an Einstein summation (Esum) operation, followed by singular value decomposition (SVD):

$$\text{Esum}('lD, ld \to Dd', W_Q^i, W_K^i) = U_{d'}\Sigma_{d'}V_{d'}. \quad (1)$$

where $U_{d'} \in \mathbb{R}^{d'\times d'}$ and $V_{d'} \in \mathbb{R}^{d'\times d'}$ are the left and right singular vector matrices, respectively, and $\Sigma_{d'} \in \mathbb{R}^{d'\times d'}$ is the diagonal matrix containing the singular values (in descending order). The diagonal elements of $\Sigma_{d'}$ are taken as the singular value distribution of the static query-key weight pair for the i-th attention head.

Data-Driven Weight Pair (TransKV). For the data-driven counterpart, consider an arbitrary input sequence $X \in \mathbb{R}^{L\times d}$, where L is the sequence length. The corresponding query and key matrices for the i-th attention head are $Q^i = XW_Q^i \in \mathbb{R}^{L\times d'}$ and $K^i = XW_K^i \in \mathbb{R}^{L\times d'}$. The data-driven singular value distribution is obtained analogously by performing SVD on the outer product projected along the sequence dimension:

$$\text{Esum}('lD, ld \to Dd', XW_Q^i, XW_K^i) = U_{d'}^{'}\Sigma_{d'}^{'}V_{d'}^{'}. \quad (2)$$

Here, the diagonal elements of $\Sigma_{d'}^{'}$ represent the singular value distribution of the data-driven query-key pair for the i-th attention head.

Visualization on ViT variants. We apply the aforementioned methodology to ViT variants, including ViT-Tiny (ViT-T), ViT-Base (ViT-B), and ViT-Large (ViT-L), focusing on layers 6 and 12 (with layer indexing starting from 1). For visualization clarity, we report results for only the first three attention heads in each model (out of 3 heads in ViT-T, 12 heads in ViT-B, and 16 heads in ViT-L). As shown in Figures 1, 2, and 3, the singular value distributions of static query-key weight pairs and their data-driven counterparts exhibit remarkable inconsistency. This phenomenon persists across different layers and attention heads in all examined ViT variants.

A.2. Task Sensitivity Completed Results

In the main paper, we present the average performance across common-sense reasoning datasets in the task sensitivity experiments. Here, we provide the complete results for each individual dataset in Table 1.

As shown in the table, TransKV driven by task-oriented projection matrices consistently and significantly outperform their counterparts driven by WT2 [9] or PTB [7] across various pruning ratios and datasets. This suggests that higher-quality, task-relevant projection matrices possess a superior ability to identify and eliminate latent redundancies compared to those derived from generic corpora. Furthermore, we observe that WT2-driven variants generally achieve better performance than PTB-driven ones. A plausible explanation is that the WT2 corpus contains substantially more commonsense knowledge than PTB, which partially compensates for its lower task specificity.

A.3. Scalability Completed Results

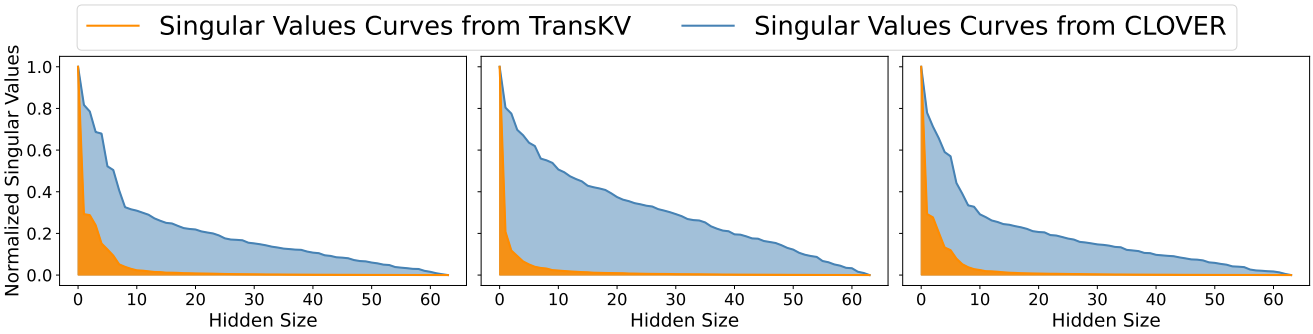
In the main paper, we demonstrate the scalability of TransKV on Qwen-2.5 models [10] ranging from 3B to 14B parameters. Here, we present the complete results across all Qwen-2.5 variants (0.5B, 1.5B, 3B, 7B, and 14B) by comparing TransKV with FLAP [1] under various pruning ratios.

As shown in Table 2, TransKV consistently and significantly outperforms FLAP across all model sizes and pruning ratios. These complete results reveal that TransKV excels at identifying both mild redundancies in smaller models (e.g., 0.5B and 1.5B) and substantial redundancies in larger models (e.g., 3B, 7B, and 14B). Moreover, TransKV exhibits particularly strong performance in high-redundancy scenarios, making it especially well-suited for compressing LFMs with massive parameter counts.

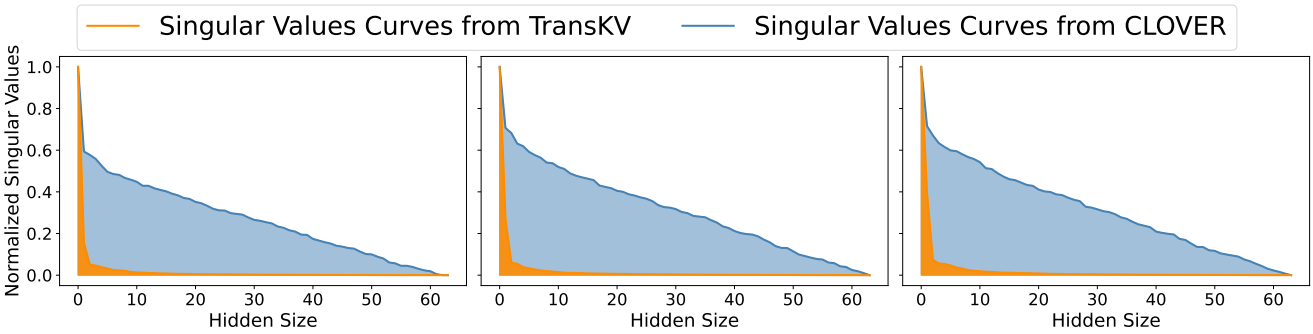
A.4. Speed Test Completed Results

In the main paper, we report the average speed test results of TransKV on a single H20 GPU. Here, we provide the complete evaluation across batch sizes ranging from 8 to 64, reporting minimum, maximum, and average values for each metric. We compare the original LLaMA-3-8B model with its 50%-pruned version produced by TransKV.

As shown in Table **??**, we measure end-to-end latency, time-to-first-token (T2F), inter-token latency (I.Tok.), and

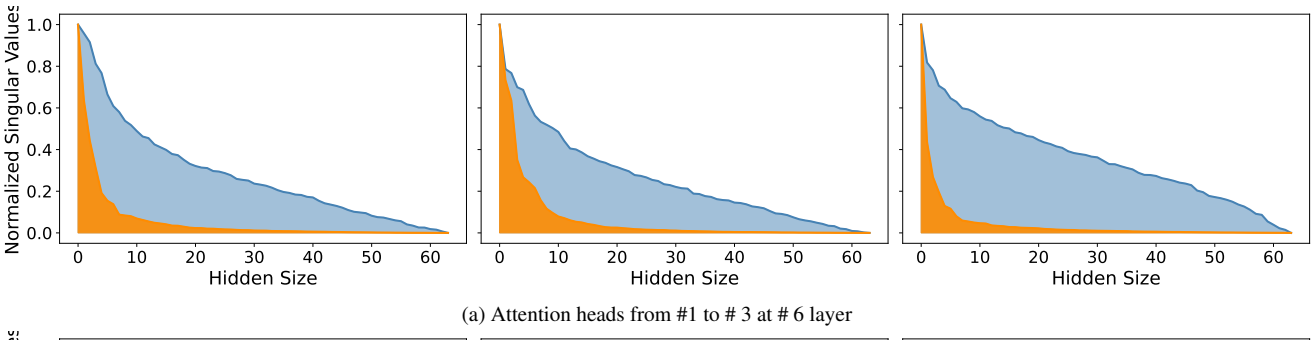


(a) Attention heads from #1 to # 3 at # 6 layer

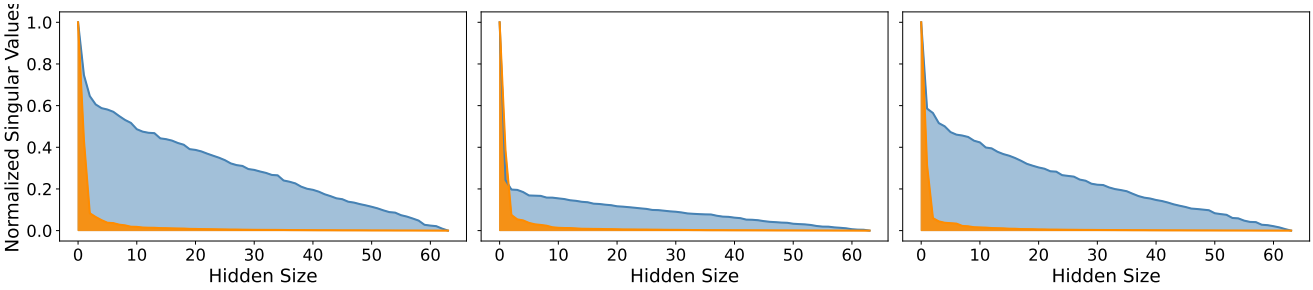


(b) Attention heads from #1 to # 3 at # 12 layer

Figure 1. ViT-T singular value distribution examples from CLOVER and TransKV

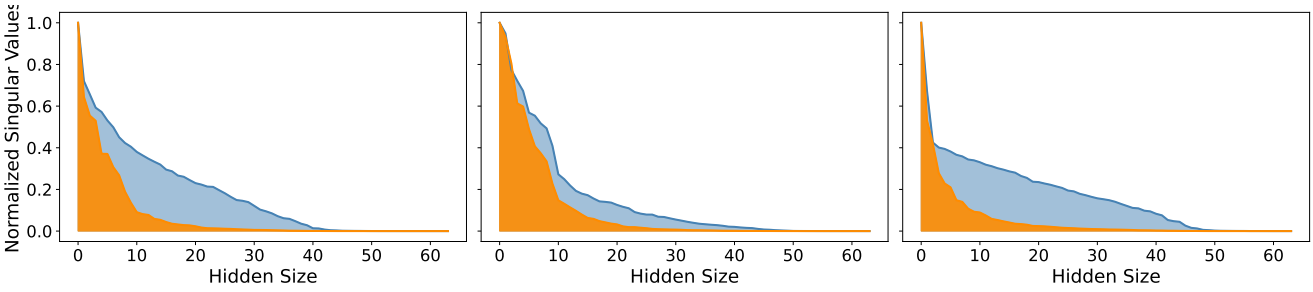


(a) Attention heads from #1 to # 3 at # 6 layer

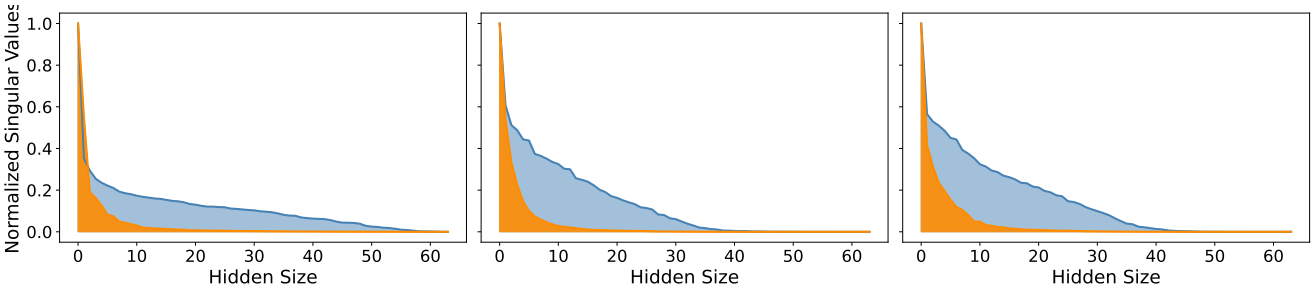


(b) Attention heads from #1 to # 3 at # 12 layer

Figure 2. ViT-B singular value distribution examples from CLOVER and TransKV



(a) ViT-L Attention heads from #1 to # 3 at # 6 layer



(b) ViT-L Attention heads from #1 to # 3 at # 12 layer

Figure 3. ViTs singular value distribution examples from CLOVER and TransKV

Table 1. TransKV task sensitivity Completed Results. Task, WT2, and PTB denote projection matrices are constructed by corresponding tasks, Wiki-text 2, and penn treebank.

Ratio	Proj.	BoolQ	PIQA	HellaS	WinoG	ARC-e	ARC-c	OBQA	Avg↑
0%	-	81.01	79.49	60.15	73.48	79.88	50.00	34.20	65.46
10%	Task	**79.57**	**78.94**	**58.66**	**73.24**	**79.08**	49.49	**34.60**	**64.80**
	WT2	73.61	78.29	56.40	72.69	78.83	**49.57**	33.00	63.20
	PTB	73.61	78.78	56.40	68.75	76.85	**49.57**	27.00	61.57
20%	Task	**75.23**	**78.07**	**56.51**	**73.24**	**79.59**	**48.38**	**34.80**	**63.69**
	WT2	61.62	75.95	49.83	64.80	75.34	43.77	23.40	56.39
	PTB	61.62	75.19	49.83	64.80	70.29	43.77	27.00	56.07
30%	Task	**71.04**	**77.20**	**53.15**	**73.32**	**77.99**	**46.42**	**35.60**	**62.10**
	WT2	50.43	72.20	41.24	58.80	68.81	35.41	24.00	50.13
	PTB	50.43	69.48	41.24	54.54	59.68	35.41	22.00	47.54
40%	Task	**61.16**	**75.68**	**48.73**	**68.67**	**76.47**	**43.60**	**35.00**	**58.47**
	WT2	40.83	66.59	33.39	52.64	58.71	23.63	17.60	41.91
	PTB	40.83	64.15	33.39	51.46	45.62	23.63	15.80	39.27
50%	Task	**56.02**	**74.70**	**43.01**	**64.01**	**73.86**	**38.74**	**32.20**	**54.65**
	WT2	38.01	60.88	28.77	51.93	45.33	19.20	15.20	37.05
	PTB	38.01	59.09	28.77	49.88	35.02	19.20	13.60	34.79

throughput (Thr.) on a single H20 GPU. As batch size and sequence length increase, TransKV consistently and substantially outperforms the original LLaMA-3-8B across all metrics. This demonstrates that the additional latency previously introduced by RoPE-aware pruning in TransKV is effectively eliminated once the batch size and sequence length exceed a certain threshold, fully realizing the acceleration benefits of KV cache compression in practical deployment.

A.5. Plug and Play Analysis Completed Results

In the main paper, we demonstrate the plug-and-play compatibility of TransKV by integrating it with LLM-Pruner. Here, we present the complete results when replacing or

Table 2. Scalability test on Qwen2.5. FLAP's performance is shown in gray color and TransKV's performance is shown in black color. The **bold** value denotes the best performance.

Size	Ratio	RTE	SST2	COPA
0.5B	0%	58.84	54.13	74.00
	10%	55.23/**55.60**	**51.38**/49.89	72.00/**74.00**
	20%	52.35/**56.32**	**57.57**/50.11	74.00/**77.00**
	30%	**53.07**/49.82	**56.08**/51.49	72.00/**73.00**
	40%	**53.43**/48.38	**51.38**/49.20	**76.00**/70.00
	50%	**49.10**/47.65	**57.11**/49.89	**72.00**/61.00
1.5B	0%	70.04	88.42	83.00
	10%	68.59/**71.84**	**89.45**/86.24	83.00/83.00
	20%	**69.68**/58.12	**89.22**/49.77	**80.00**/77.00
	30%	**64.98**/57.76	**87.61**/72.94	77.00/**81.00**
	40%	**55.23**/49.82	**86.47**/48.62	**75.00**/60.00
	50%	48.74/**50.54**	**81.77**/49.77	**68.00**/63.00
3B	0%	75.45	90.14	85.00
	10%	75.81/**81.23**	89.22/**89.56**	**84.00**/83.00
	20%	**79.78**/78.34	**88.76**/74.66	**84.00**/83.00
	30%	**74.73**/71.12	**89.45**/69.04	79.00/**86.00**
	40%	62.82/**71.12**	**88.30**/85.55	80.00/**83.00**
	50%	**57.40**/53.79	**87.27**/48.74	**76.00**/66.00
7B	0%	81.59	91.86	91.00
	10%	**80.87**/79.06	91.63/**91.86**	**90.00**/87.00
	20%	**80.51**/77.26	**91.74**/89.11	89.00/**90.00**
	30%	**80.14**/71.12	**90.25**/80.39	91.00/**92.00**
	40%	**76.17**/55.96	**90.37**/64.68	**86.00**/85.00
	50%	**69.31**/52.35	**85.21**/51.03	**83.00**/60.00
14B	0%	80.14	89.56	90.00
	10%	**80.87**/79.42	90.25/**91.17**	**91.00**/90.00
	20%	**81.23**/77.62	90.71/**91.51**	91.00/91.00
	30%	79.78/**80.51**	**90.83**/80.85	**89.00**/88.00
	40%	**77.26**/69.31	**90.71**/54.24	**89.00**/87.00
	50%	**60.65**/53.07	**87.61**/49.31	**88.00**/55.00

combining TransKV's attention pruning module with two representative structured pruning methods—FLAP [1] and SliceGPT [2]—on common-sense reasoning benchmarks (Figures 4 and 5).

As shown in Figure 4, directly substituting TransKV for FLAP's original attention pruning mechanism consistently improves performance across all evaluated pruning ratios. This confirms that TransKV's task-sensitive singular value-guided attention pruning is strictly superior to FLAP's heuristic-based approach under identical experimental conditions.

For SliceGPT (Figure 5), we retain SliceGPT's attention pruning pipeline while incorporating TransKV for attention module compression (note that SliceGPT's MLP importance scores are computed on attention outputs necessitate this sequential combination). The results reveal that TransKV boosts SliceGPT's performance at high pruning ratios (e.g., 60%–70%), whereas the improvement is marginal or even slightly negative at lower ratios. This phenomenon can be attributed to the fundamental difference in pruning granularity: SliceGPT primarily reduces the hidden dimension via its projection matrix, whereas TransKV prunes the per-head attention dimension. At low pruning ratios, attention-dimension reduction by TransKV may disturb the principal components estimated by SliceGPT on the original attention outputs, thereby altering the data distribution assumed by subsequent MLP pruning and introducing minor performance degradation. In contrast, when overall sparsity is high, the abundant redundancy in large language models allows TransKV's more accurate importance evaluation to dominate, yielding clear gains even in hybrid settings.

A.6. Ablation Study on Projection Matrices Completed Results

In the main paper, we present ablation results focusing on the value-output pairs, evaluating three candidate projection matrices: P_V^i, P_O^i, and $P_V^i(P_V^i)^\top P_O^i$. Here, we provide the complete ablation study that includes both query-key pairs (P_Q^i, P_K^i, and $P_Q^i(P_Q^i)^\top P_K^i$) and value-output pairs (P_V^i, P_O^i, and $P_V^i(P_V^i)^\top P_O^i$).

As shown in Table 3, the choice of projection matrix has a substantial impact on final pruning performance. For the value-output pair, P_V^i emerges as the optimal projection matrix for W_V^i, while its transpose $(P_V^i)^\top$ yields the best results when applied to W_O^i. In the query-key pair, P_Q^i and P_K^i exhibit nearly identical effectiveness when used to prune W_Q^i, and correspondingly, their transposes perform comparably on W_K^i. Consequently, we recommend using P_K^i to project W_Q^i and $(P_K^i)^\top$ to project W_K^i. This symmetric strategy based on the key projection matrix is not only empirically strong but also universally applicable—it works seamlessly for both MHA and GQA architectures.

A.7. Speech Processing Completed Result

In the main paper, we present comparison results on the English subset, evaluating Taylor, CLOVER, and TransKV across various pruning ratios. Here, we provide the complete results on both English and Cantonese subsets, including additional baselines (L1, L2, and Random).

As shown in Table 5, when pruning the encoder components, TransKV significantly outperforms all methods on both the English and Cantonese subsets across all pruning ratios. For decoder-only pruning, TransKV maintains consistent superiority at moderate pruning ratios (10%–40%), but experiences a sharp performance degradation at higher ratios 50%, dropping to levels comparable with other methods. This sudden collapse occurs because the redundancy in the decoder has reached its detectable limit; more aggressive compression exceeds the model's tolerance, causing an

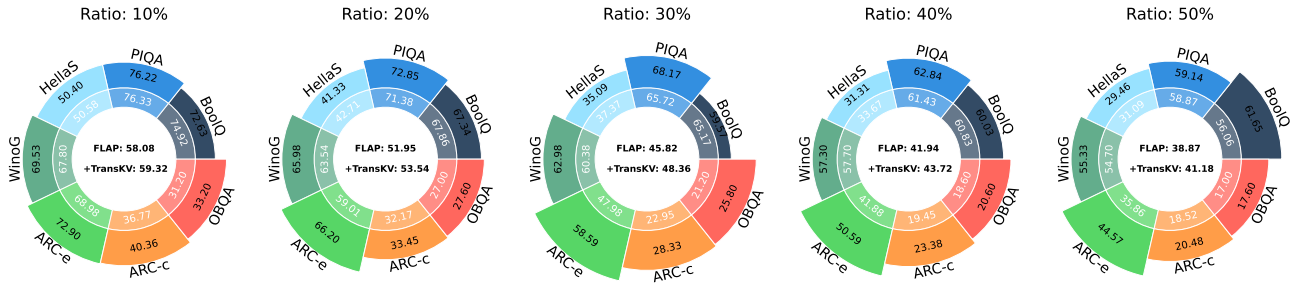


Figure 4. TransKV plugs into FLAP. Performance of inner- ring is the original FLAP and performance of outer-ring is the TransKV enhanced FLAP.

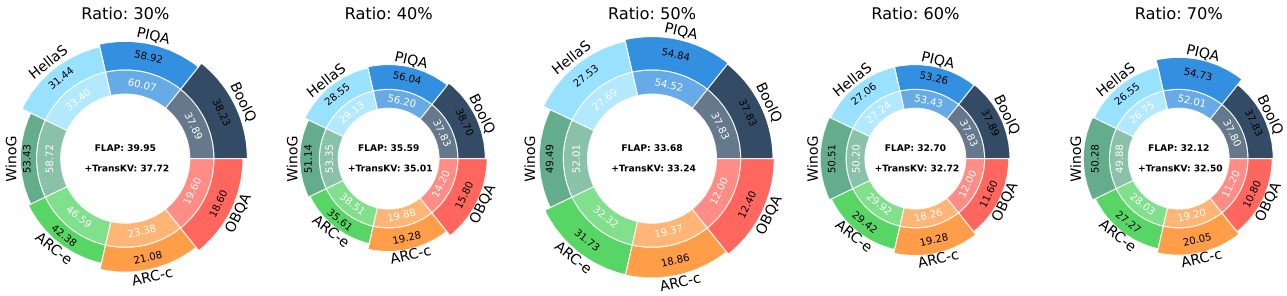


Figure 5. TransKV plugs into SliceGPT. Performance of inner- ring is the original SliceGPT and performance of outer-ring is the TransKV enhanced SliceGPT.

abrupt loss of critical information. A similar pattern and underlying mechanism are observed when the encoder and decoder are pruned jointly.

A.8. Compare to EigenAttention and MatryoshkaKV

Commonality & Distinction. While TransKV shares prune-then-recover framework with mentioned papers, our core contribution lies in a data-driven, training-free efficiency that fundamentally differs from the baselines.

Difference from EigenAttention (PCA). 1. Task-Specific vs. Generic. EigenAttention derives projections from generic corpora via SVD. In contrast, TransKV is data-driven, constructing projection matrices from task-specific data to better capture task-relevant features (see Table 4). **2. Peak Memory Efficiency**. EigenAttention reconstructs the pruned Key during attention, causing full-rank calculation overhead. TransKV operates on RoPE(K) to conduct online pruning, strictly maintaining low-rank calculations and significantly reducing peak memory.

Difference from MatryoshkaKV (Mat.KV). 1. Training-Free vs. Heavy Pre-training. MatryoshkaKV relies on 'learned' projection matrices requiring 56 GPU hours on 7 $\times$ 96GB H20s (reproduced by us) and massive generic data (1.2B). TransKV employs a training-free statistical initialization. **2. Total Budget & Coexistence**. As shown in Table 4, while LoRA enables general performance recovery, TransKV uniquely bridges the initialization gap via low-cost tuning. Crucially, TransKV+LoRA outperforms Mat.KV+LoRA with a lower total budget. Since both final performance converges close to baseline, TransKV stands out as the most efficient alternative where extensive pre-training is infeasible.

A.9. TransKV for Intra-ImageNet-1K sublabels Sensitivity

In the main paper, we conduct a cross-dataset sensitivity analysis for TransKV. Here, in the appendix, we further investigate its sensitivity to different label subsets within the same dataset. We use ImageNet-1K [4] classification with ViT-Base as the target task and model. The 1,000 ImageNet classes are sequentially partitioned into 10 equal subsets (100 classes each): the first subset contains classes 0–99, the second contains classes 100–199, and so on. This yields 10 subclass-specific sub-datasets. We then construct 11 different projection matrices using identical class-balanced sampling strategies: one from a 15% random sample of the full ImageNet training set, and the remaining ten from 15% samples of each corresponding sub-dataset.

Tables 6 and 7 report the pruning performance of TransKV at 30% and 50% compression ratios, respectively. For each table, we evaluate 11 variants (each equipped with

Table 3. Ablation study on projection matrices in both query-key pair and value-output pair.

Dataset				Ratio	CIFAR10	CIFAR100	iNaturalist
Baseline				0%	97.68	90.56	73.80
Proj. for W_Q^i	Proj. for W_K^i	Proj. for W_V^i	Proj. for W_O^i				
P_Q^i	$(P_Q^i)^\top$	P_V^i	$(P_V^i)^\top$	10%	<u>97.62</u>	<u>90.08</u>	**62.11**
P_Q^i	$(P_Q^i)^\top$	P_O^i	$(P_O^i)^\top$		97.39	88.55	59.64
P_Q^i	$(P_Q^i)^\top$	$P_V^i(P_V^i)^\top P_O^i$	$(P_O^i)^\top$		97.33	88.20	58.55
$P_Q^i(P_Q^i)^\top P_K^i$	$P_K^\top$	P_V^i	$(P_V^i)^\top$		97.60	90.14	<u>61.91</u>
$P_Q^i(P_Q^i)^\top P_K^i$	$P_K^\top$	P_O^i	$(P_O^i)^\top$		97.38	88.45	59.67
$P_Q^i(P_Q^i)^\top P_K^i$	$P_K^\top$	$P_V^i(P_V^i)^\top P_O^i$	$(P_O^i)^\top$		97.26	88.05	58.28
P_K	$P_K^\top$	P_V^i	$(P_V^i)^\top$		**97.65**	**90.19**	61.88
P_K	$P_K^\top$	P_O^i	$(P_O^i)^\top$		97.37	88.58	59.47
P_K	$P_K^\top$	$P_V^i(P_V^i)^\top P_O^i$	$(P_O^i)^\top$		97.31	88.27	58.65
P_Q^i	$(P_Q^i)^\top$	P_V^i	$(P_V^i)^\top$	20%	97.50	**89.80**	**60.73**
P_Q^i	$(P_Q^i)^\top$	P_O^i	$(P_O^i)^\top$		96.25	85.75	55.05
P_Q^i	$(P_Q^i)^\top$	$P_V^i(P_V^i)^\top P_O^i$	$(P_O^i)^\top$		95.76	84.66	52.67
$P_Q^i(P_Q^i)^\top P_K^i$	$P_K^\top$	P_V^i	$(P_V^i)^\top$		<u>97.51</u>	89.69	60.36
$P_Q^i(P_Q^i)^\top P_K^i$	$P_K^\top$	P_O^i	$(P_O^i)^\top$		96.18	85.52	54.36
$P_Q^i(P_Q^i)^\top P_K^i$	$P_K^\top$	$P_V^i P_{V^T} P_O^i$	$(P_O^i)^\top$		95.58	84.58	52.44
P_K	$P_K^\top$	P_V^i	$(P_V^i)^\top$		**97.52**	<u>89.79</u>	<u>60.56</u>
P_K	$P_K^\top$	P_O^i	$(P_O^i)^\top$		96.28	86.19	55.12
P_K	$P_K^\top$	$P_V^i(P_V^i)^\top P_O^i$	$(P_O^i)^\top$		95.62	85.16	52.77
P_Q^i	$(P_Q^i)^\top$	P_V^i	$(P_V^i)^\top$	30%	**97.25**	**88.72**	**58.91**
P_Q^i	$(P_Q^i)^\top$	P_O^i	$(P_O^i)^\top$		93.06	76.93	47.92
P_Q^i	$(P_Q^i)^\top$	$P_V^i(P_V^i)^\top P_O^i$	$(P_O^i)^\top$		92.55	73.40	41.82
$P_Q^i(P_Q^i)^\top P_K^i$	$P_K^\top$	P_V^i	$(P_V^i)^\top$		<u>97.16</u>	88.67	58.51
$P_Q^i(P_Q^i)^\top P_K^i$	$P_K^\top$	P_O^i	$(P_O^i)^\top$		92.86	76.04	47.36
$P_Q^i(P_Q^i)^\top P_K^i$	$P_K^\top$	$P_V^i(P_V^i)^\top P_O^i$	$(P_O^i)^\top$		92.31	72.65	41.16
P_K	$P_K^\top$	P_V^i	$(P_V^i)^\top$		**97.25**	<u>88.70</u>	<u>58.68</u>
P_K	$P_K^\top$	P_O^i	$(P_O^i)^\top$		93.20	77.96	48.12
P_K	$P_K^\top$	$P_V^i(P_V^i)^\top P_O^i$	$(P_O^i)^\top$		92.62	74.50	42.24
P_Q^i	$(P_Q^i)^\top$	P_V^i	$(P_V^i)^\top$	40%	**96.87**	<u>86.53</u>	**56.44**
P_Q^i	$(P_Q^i)^\top$	P_O^i	$(P_O^i)^\top$		83.15	59.88	36.73
P_Q^i	$(P_Q^i)^\top$	$P_V^i(P_V^i)^\top P_O^i$	$(P_O^i)^\top$		80.03	53.58	27.29
$P_Q^i(P_Q^i)^\top P_K^i$	$P_K^\top$	P_V^i	$(P_V^i)^\top$		96.68	86.22	55.31
$P_Q^i(P_Q^i)^\top P_K^i$	$P_K^\top$	P_O^i	$(P_O^i)^\top$		92.86	76.04	47.36
$P_Q^i(P_Q^i)^\top P_K^i$	$P_K^\top$	$P_V^i(P_V^i)^\top P_O^i$	$(P_O^i)^\top$		92.31	72.65	41.16
P_K	$P_K^\top$	P_V^i	$(P_V^i)^\top$		<u>96.86</u>	**86.66**	<u>56.11</u>
P_K	$P_K^\top$	P_O^i	$(P_O^i)^\top$		84.50	62.34	37.26
P_K	$P_K^\top$	$P_V^i(P_V^i)^\top P_O^i$	$(P_O^i)^\top$		81.27	56.37	28.02
P_Q^i	$(P_Q^i)^\top$	P_V^i	$(P_V^i)^\top$	50%	**96.08**	**82.90**	<u>49.77</u>
P_Q^i	$(P_Q^i)^\top$	P_O^i	$(P_O^i)^\top$		59.81	24.80	18.84
P_Q^i	$(P_Q^i)^\top$	$P_V^i(P_V^i)^\top P_O^i$	$(P_O^i)^\top$		50.09	16.96	8.78
$P_Q^i(P_Q^i)^\top P_K^i$	$P_K^\top$	P_V^i	$(P_V^i)^\top$		95.60	81.89	48.58
$P_Q^i(P_Q^i)^\top P_K^i$	$P_K^\top$	P_O^i	$(P_O^i)^\top$		62.85	21.98	18.22
$P_Q^i(P_Q^i)^\top P_K^i$	$P_K^\top$	$P_V^i(P_V^i)^\top P_O^i$	$(P_O^i)^\top$		53.11	19.07	10.26
P_K	$P_K^\top$	P_V^i	$(P_V^i)^\top$		<u>95.85</u>	<u>82.77</u>	**50.92**
P_K	$P_K^\top$	P_O^i	$(P_O^i)^\top$		62.24	27.75	20.17
P_K	$P_K^\top$	$P_V^i(P_V^i)^\top P_O^i$	$(P_O^i)^\top$		51.70	19.67	10.03

Table 4. Results on BoolQ, PIQA, HellaS, WinoG, ARC-e, ARC-c, and OBQA at 50% pruning rate.

Method	Total Budget (Prune-then-Recover)				Acc.
	Train	GPU (GB)	Param	Tokens	Avg.
Baseline	-	-	-	-	65.45
PCA	0 h	0	0 M	157.1 K	32.73
TransKV	0 h	0	0 M	7.0 M	**54.65**
Mat.KV	56 h	88 × 7	470 M	1.2 B	59.71
TransKV + LoRA	0 h + 0.5 h	0 + 8.3 × 7	0 M + 25.7 M	7.0 M + 298.0 K	**60.61**
Mat.KV + LoRA	56 h + 0.5 h	88 × 7 + 8.3 × 7	470 M + 25.7 M	1.2 B + 298.0 K	62.76
TransKV + LoRA	0 h + 1.3 h	0 + 8.3 × 7	0 M + 25.7 M	7.0 M + 1.1 M	**63.00**
Mat.KV + LoRA	56 h + 1.4 h	88 × 7 + 8.3 × 7	470 M + 25.7 M	1.2 B + 1.1 M	65.15
TransKV + LoRA	0 h + 1.7 h	0 + 8.4 × 7	0 M + 241.4 M	7.0 M + 1.1 M	**65.23**

one of the 11 projection matrices) on both the full ImageNet validation set and the 10 sub-datasets. The results clearly demonstrate that TransKV achieves the highest performance when the projection matrix is derived from the same data distribution as the evaluation set. This consistent matching effect strongly confirms TransKV's robust data-driven capability for identifying intra-dataset redundancies.

A.10. TransKV for SmolLM3-3B on MMLU Benchmark

Here, we further evaluate TransKV on SmolLM3-3B [3], a state-of-the-art 3-billion-parameter dense language model that incorporates both RoPE and non-RoPE operations. Performance is measured on the Massive Multitask Language Understanding (MMLU) benchmark [6], which comprises multiple-choice questions spanning 57 diverse subjects and serves as a standard evaluation suite for large language models. We compare TransKV against several baselines (L1, L1, L2, Random, Taylor, and CLOVER) under two settings: (1) Non-RoPE pruning only, and (2) joint RoPE + Non-RoPE pruning, across a range of pruning ratios.

As shown in Table 8, in the Non-RoPE pruning setting, TransKV consistently outperforms L1, L2, Random, and Taylor across all pruning ratios in both individual subtasks and overall MMLU score. TransKV achieves performance comparable to CLOVER in this setting, which is expected since Non-RoPE pruning affects only the final 9 layers of SmolLM3-3B, leaving relatively limited redundancy for compression. When RoPE and Non-RoPE components are pruned jointly, TransKV substantially surpasses all competing methods, whereas CLOVER is inapplicable due to its lack of RoPE support. These results highlight the broader applicability and superior effectiveness of TransKV when handling diverse attention mechanisms and positional encoding schemes in modern language models.

References

[1] 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*, pages 10865–10873, 2024. 1, 4

[2] Saleh Ashkboos, Maximilian L Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. In *The Twelfth International Conference on Learning Representations*, 2024. 4

[3] Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3, 2025. 7

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[6] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. 7

[7] Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Using Large Corpora*, 273: 31, 1994. 1

[8] Fanxu Meng, Pingzhi Tang, Fan Jiang, and Muhan Zhang. CLOVER: Cross-layer orthogonal vectors pruning. In *International Conference on Machine Learning*, 2025. 1

[9] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. 1

[10] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. 1

Table 5. Comparative performance across different pruning rates on Common Voice 17. The traditional pruning methods are applied exclusively to the linear layers (Query, Key, Value, and Output) within the Attention module. On English, TransKV computes its rotation matrices using 5% of the training data; on Cantonese, 100% of the training data is used for the same purpose.

	Ratio	Encoder				Decoder				Encoder-Decoder			
		English		Cantonese		English		Cantonese		English		Cantonese	
		WER	CER	WER	CER	WER	CER	WER	CER	WER	CER	WER	CER
Baseline	0%	21.43	9.54	8.56	1.27	21.43	9.54	8.56	1.27	21.43	9.54	8.56	1.27
L1	10%	181.38	171.73	99.25	344.27	96.78	97.61	1028.05	439.77	99.63	99.71	103.23	98.66
L2		129.63	115.03	95.19	166.15	97.62	98.06	2556.74	1105.09	102.84	102.71	3992.6	1373.19
Random		167.34	145.18	99.59	339.87	123.29	111.8	99.14	110.54	101.53	97.53	100.0	206.05
Taylor		23.7	11.06	14.87	2.52	28.87	14.77	46.94	9.21	30.69	15.97	47.09	9.36
CLOVER		24.24	11.41	14.76	2.58	86.7	74.57	43.11	43.57	89.06	74.32	48.33	34.81
TransKV		**21.8**	**9.77**	**9.61**	**1.42**	**21.63**	**9.73**	**10.29**	**1.53**	**21.89**	**9.89**	**10.44**	**1.56**
L1	20%	99.75	89.48	199.85	392.83	101.94	97.86	388.62	180.91	99.57	97.72	772.47	362.4
L2		106.89	99.15	413.14	668.12	537.25	283.5	19075.07	3939.98	2321.04	1566.55	10549.98	4071.31
Random		123.01	113.62	100.0	1547.37	99.31	99.6	100.0	103.84	99.87	101.5	100.0	160.49
Taylor		33.2	17.75	36.31	8.51	340.63	227.48	1681.86	750.38	86.57	72.32	979.27	424.38
CLOVER		28.48	14.57	33.16	7.68	173.68	156.04	126.89	126.68	229.95	203.35	129.22	118.26
TransKV		**22.29**	**10.09**	**10.1**	**1.66**	**22.05**	**9.98**	**13.11**	**1.97**	**22.54**	**10.26**	**15.06**	**2.38**
L1	30%	100.65	90.45	4102.93	2432.81	100.0	196.65	100.0	2200.72	99.92	95.3	100.0	2200.72
L2		101.28	85.66	825.8	472.47	99.99	98.01	100.0	100.81	98.62	96.46	100.0	100.0
Random		918.9	397.88	130.49	194.18	100.0	98.33	110.4	198.16	2302.09	779.17	21846.6	4336.71
Taylor		83.45	61.61	90.88	75.46	119.88	103.79	3009.84	1039.11	99.69	98.04	712.24	273.45
CLOVER		60.95	45.53	72.4	35.37	122.01	104.35	200.53	132.9	156.97	132.02	196.24	128.92
TransKV		**23.41**	**10.87**	**13.67**	**2.66**	**24.41**	**11.6**	**21.55**	**3.43**	**25.8**	**12.82**	**28.09**	**4.96**
L1	40%	117.34	100.27	14704.39	8735.62	100.0	382.85	100.0	2200.72	100.0	410.92	100.0	2200.72
L2		132.9	111.35	904.54	523.08	99.99	102.31	100.0	2200.72	100.0	386.89	100.0	2200.72
Random		354.26	1132.77	132.22	2077.18	100.42	298.52	100.0	103.59	100.0	100.0	100.0	100.0
Taylor		157.73	158.86	160.08	448.78	96.89	97.16	998.95	447.64	100.05	410.93	100.0	702.2
CLOVER		149.68	130.99	90.54	104.54	93.42	93.59	100.0	100.0	107.28	101.28	102.48	99.98
TransKV		**26.36**	**13.09**	**20.84**	**3.87**	**35.23**	**19.39**	**47.99**	**15.06**	**35.47**	**18.84**	**56.93**	**17.82**
L1	50%	920.59	1114.77	559.37	1519.47	99.99	98.37	**100.0**	**100.0**	100.0	98.67	**100.0**	2200.72
L2		582.19	535.54	448.59	1259.23	99.97	98.5	**100.0**	2117.88	96.11	95.43	9539.5	5018.09
Random		99.97	562.43	341.38	3725.61	100.0	410.9	**100.0**	986.21	652.36	272.75	**100.0**	3669.87
Taylor		1227.33	1194.97	100.08	545.07	98.42	168.64	**100.0**	100.01	99.98	98.34	**100.0**	**100.0**
CLOVER		205.37	180.69	99.96	436.67	**96.86**	96.58	**100.0**	**100.0**	99.76	101.28	**100.0**	**100.0**
TransKV		**38.49**	**23.22**	**36.99**	**9.08**	106.44	**82.12**	100.3	178.14	**85.5**	**60.6**	101.92	139.25

Table 6. TransKV results on ImageNet with a 30% pruning ratio on ViT-Base (accuracy). The global projection for ImageNet is constructed using 15% of training data. For sub-tasks (e.g., 000–099), the projection matrices are derived from subsets of this 15% training data, restricted to samples belonging to the corresponding class labels of each sub-task. The box highlight the best result.

Projection \ Dataset	Whole	000-099	100-199	200-299	300-399	400-499	500-599	600-699	700-799	800-899	900-999
Proj. from whole	67.93	73.26	70.94	68.88	76.06	63.26	64.86	63.16	66.46	62.18	70.28
Proj. from 000-099	66.21	76.08	70.34	67.94	76.18	60.10	61.36	59.10	63.14	58.46	69.44
Proj. from 100-199	67.19	73.46	75.24	68.70	75.36	60.66	62.66	60.92	64.26	60.54	70.14
Proj. from 200-299	66.88	72.64	69.94	71.38	75.22	60.96	62.64	61.30	64.86	60.20	69.64
Proj. from 300-399	66.59	73.60	70.84	68.56	78.66	60.12	61.76	59.82	63.80	59.72	68.98
Proj. from 400-499	66.28	68.24	66.56	63.78	72.30	68.48	65.04	62.92	65.16	62.00	68.32
Proj. from 500-599	66.41	68.26	66.44	63.54	72.24	63.50	70.46	62.50	66.32	62.44	68.36
Proj. from 600-699	66.47	68.24	67.46	64.58	73.04	62.92	65.18	68.12	65.46	62.12	67.54
Proj. from 700-799	66.24	68.52	66.76	63.80	71.48	62.98	65.30	62.38	70.92	62.38	67.84
Proj. from 800-899	66.55	68.76	67.08	64.58	72.42	63.26	65.30	63.12	65.24	67.22	68.54
Proj. from 900-999	66.11	71.14	67.44	64.18	73.54	61.18	63.10	60.48	63.54	61.24	75.26

Table 7. TransKV results on ImageNet with a 50% pruning ratio on ViT-Base (accuracy). The global projection for ImageNet is constructed using 15% of training data. For sub-label dataset (e.g., 000–099), the projection matrices are derived from subsets of this 15% training data, restricted to samples belonging to the corresponding class labels of each sub-task. The box highlight the best result.

Projection \ Dataset	Whole	000-099	100-199	200-299	300-399	400-499	500-599	600-699	700-799	800-899	900-999
Proj. from whole	67.93	73.26	70.94	68.88	76.06	63.26	64.86	63.16	66.46	62.18	70.28
Proj. from 000-099	66.21	76.08	70.34	67.94	76.18	60.10	61.36	59.10	63.14	58.46	69.44
Proj. from 100-199	67.19	73.46	75.24	68.70	75.36	60.66	62.66	60.92	64.26	60.54	70.14
Proj. from 200-299	66.88	72.64	69.94	71.38	75.22	60.96	62.64	61.30	64.86	60.20	69.64
Proj. from 300-399	66.59	73.60	70.84	68.56	78.66	60.12	61.76	59.82	63.80	59.72	68.98
Proj. from 400-499	66.28	68.24	66.56	63.78	72.30	68.48	65.04	62.92	65.16	62.00	68.32
Proj. from 500-599	66.41	68.26	66.44	63.54	72.24	63.50	70.46	62.50	66.32	62.44	68.36
Proj. from 600-699	66.47	68.24	67.46	64.58	73.04	62.92	65.18	68.12	65.46	62.12	67.54
Proj. from 700-799	66.24	68.52	66.76	63.80	71.48	62.98	65.30	62.38	70.92	62.38	67.84
Proj. from 800-899	66.55	68.76	67.08	64.58	72.42	63.26	65.30	63.12	65.24	67.22	68.54
Proj. from 900-999	66.11	71.14	67.44	64.18	73.54	61.18	63.10	60.48	63.54	61.24	75.26

Table 8. Zero-shot accuracy results on SmolLM3-3B in MMLU benchmark (57 sub-tasks). Attention in SmolLM3-3B is GQA. It has 36 attention layers, where each three RoPE GQA layer follows by a None-RoPE GQA layer. In the table, Soc. Sci., Hum. and Oth. denote Social Sciences, Humanities, and Other, respectively. Note that, the average metric is calculated by each sub-task accuracy.

		Non-RoPE (9 Layers)					RoPE and Non-RoPE (36 Layers)				
	Ratio	STEM	Soc. Sci.	Hum.	Oth.	Avg.	STEM	Soc. Sci.	Hum.	Oth.	Avg.
Baseline	0%	0.55	0.68	0.66	0.61	0.61	0.55	0.68	0.66	0.61	0.61
L1	10%	0.41	0.52	0.54	0.50	0.49	0.23	0.25	0.25	0.24	0.24
L2		0.43	0.52	0.53	0.50	0.49	0.23	0.26	0.24	0.24	0.24
Random		0.52	0.64	0.65	0.60	0.60	0.35	0.44	0.42	0.42	0.40
Taylor		0.53	0.65	0.64	0.59	0.59	0.40	0.51	0.50	0.50	0.46
CLOVER		**0.54**	0.66	0.65	0.60	0.60	unsupported				
TransKV		**0.54**	**0.68**	**0.66**	**0.62**	**0.61**	**0.48**	**0.60**	**0.57**	**0.56**	**0.54**
L1	20%	0.40	0.50	0.53	0.49	0.48	0.25	0.25	0.25	0.24	0.25
L2		0.41	0.50	0.53	0.47	0.47	0.24	0.28	0.25	0.25	0.25
Random		0.49	0.62	0.60	0.57	0.56	0.23	0.23	0.24	0.25	0.24
Taylor		0.50	0.61	0.60	0.57	0.56	0.27	0.28	0.24	0.26	0.26
CLOVER		0.54	0.66	0.64	**0.62**	**0.61**	unsupported				
TransKV		**0.55**	**0.67**	**0.65**	0.61	**0.61**	**0.45**	**0.57**	**0.53**	**0.53**	**0.51**
L1	30%	0.40	0.53	0.53	0.50	0.48	0.26	0.31	0.25	0.25	0.26
L2		0.41	0.51	0.52	0.48	0.47	0.24	0.22	0.24	0.26	0.24
Random		0.46	0.57	0.56	0.53	0.52	0.22	0.21	0.24	0.24	0.23
Taylor		0.47	0.58	0.56	0.56	0.54	0.22	0.21	0.24	0.24	0.23
CLOVER		0.53	0.66	**0.64**	**0.61**	**0.60**	unsupported				
TransKV		**0.55**	**0.67**	**0.64**	**0.61**	**0.60**	**0.43**	**0.52**	**0.49**	**0.47**	**0.47**
L1	40%	0.42	0.54	0.53	0.51	0.49	0.22	0.21	0.24	0.24	0.23
L2		0.42	0.54	0.53	0.50	0.49	0.22	0.21	0.24	0.25	0.23
Random		0.43	0.53	0.52	0.48	0.48	0.22	0.21	0.24	0.24	0.23
Taylor		0.45	0.56	0.53	0.54	0.51	0.22	0.21	0.24	0.24	0.23
CLOVER		0.51	0.63	0.62	0.58	0.57	unsupported				
TransKV		**0.54**	**0.65**	**0.64**	**0.59**	**0.60**	**0.39**	**0.46**	**0.44**	**0.45**	**0.43**
L1	50%	0.43	0.55	0.53	0.50	0.49	0.22	0.21	0.24	0.24	0.23
L2		0.42	0.54	0.52	0.50	0.49	0.22	0.21	0.24	0.24	0.23
Random		0.43	0.53	0.51	0.48	0.48	0.28	0.33	0.24	0.26	0.27
Taylor		0.45	0.58	0.54	0.54	0.52	0.28	0.29	0.25	0.24	0.26
CLOVER		0.52	0.61	0.61	0.57	0.57	unsupported				
TransKV		**0.52**	**0.65**	**0.63**	**0.58**	**0.58**	**0.35**	**0.43**	**0.38**	**0.40**	**0.38**