DIMPO: DIMENSIONALITY REDUCTION FOR ATTENTION USING PREFERENCE OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A linear projection might reduce the dimension of the query and key vectors in attention without updating the pretrained model, but it remains unclear which training objective best preserves the behavior of that model. The straightforward objective is to match the full attention distribution, for example by minimizing the KL divergence. We instead ask whether a preference over keys relative to a query, and preserving the attention mass of only the highest-weighted keys, provides a better training signal, especially in long-context settings. Existing reference-free preference objectives are mostly pairwise and underuse the full key set that listwise and KL supervision already use. We therefore introduce DimPO, a loss specific to this projection, which combines listwise preference optimization with no reference model and a lightweight top-k cross-entropy term for head-fidelity supervision. To verify the direct effect of our loss on how faithfully attention is preserved, DimPO is trained offline from the attention patterns of a frozen language model, with the query and key projections optimized independently per layer. Across LLaMA3.2-3B, LLaMA3.1-8B, Qwen2.5-7B, and Qwen3-4B Instruct models, pairwise preference objectives outperform the triplet objective baseline and, on short-context tasks, retain 98% of the original score when a projection to half the dimension is applied to at most the last 40% of the attention layers. When applied to more layers or evaluated on long-context RULER, however, they no longer maintain a reasonable performance. In contrast, KL and DimPO, which use every key during training, still retain about 95% of the original RULER 4k score on the 8B model when applied to up to 50% of the layers. KL-based projections nevertheless remain closer to the original attention distribution in terms of KL divergence and closer in the MSE of the attention output, yet DimPO achieves better downstream performance. The difference becomes increasingly pronounced once the projection covers more than 50% of the attention layers, where preserving preferences over keys relative to the query, together with head-fidelity on the highest-weighted keys, improves performance on more complex tasks, including SQuAD, common-word extraction, frequent-word extraction, and variable tracking. These results suggest that, under a reduced query and key dimension, preserving the ordering and concentration of task-relevant attention can matter more than minimizing the discrepancy of the full distribution.

1 INTRODUCTION

Reducing the dimensionality of attention representations is a growing direction for making inference more efficient, both in memory and computation (Wang et al., 2020; Pope et al., 2023). Most work has focused either on reducing how many key and value vectors are represented, by selecting and retaining the most relevant tokens over time (Zhang et al., 2023; Li et al., 2024; Adnan et al., 2024), or on reducing how precisely each vector is represented through quantization (Liu et al., 2024; Lin et al., 2025b). A less explored direction is to reduce the vectors themselves, that is, the number of coordinates used to represent each query, key, or value (Wang et al., 2020; Pope et al., 2023).

The projections used in this direction have so far been linear. Random projections have been used for locality-sensitive hashing (Kitaev et al., 2020; Chen et al., 2025), while Mongoose (Chen et al., 2021b) learns a linear projection with a triplet loss. DASH-KV (Guo et al., 2026) uses asymmetric

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

deep hashing with KL-divergence-based distillation, while other approaches learn projections through end-to-end model distillation (Lin et al., 2025a).

Matching the projected attention distribution to the original provides a natural training objective, with KL divergence as a straightforward choice. Inspired by recent advances in preference optimization for post-training alignment of LLMs (Rafailov et al., 2023; Meng et al., 2024), we instead explore preference optimization as an alternative for dimensionality reduction in attention. Specifically, we ask whether preferences over keys, given a query, can preserve the original attention behavior after projecting query and key vectors to lower dimensions and thereby better preserve downstream performance, and how this compares with a KL objective. Our goal is to learn a linear projection in which a query relates more strongly to keys preferred by the original attention and less strongly to those it prefers less.

Preference optimization has usually relied on a reference model (Rafailov et al., 2023; Ethayarajh et al., 2024; Liu et al., 2025), while more recent losses remove that requirement (Hong et al., 2024; Meng et al., 2024; Xu et al., 2024). These losses are largely pairwise, and a single pair underuses the full list of keys that a KL objective already sees. Listwise losses exist (Liu et al., 2025), but they still rely on a reference model, which this setting does not provide. The weights of the highest-ranked keys matter as well, because the subsequent attention output is sensitive to that mass. We therefore introduce DimPO, a listwise preference loss with no reference model, together with a top-k cross-entropy term for head-fidelity, specific to projecting queries and keys into a lower-dimensional space.

Rather than proposing another end-to-end compression method, this work isolates the problem of learning linear projections that reduce the dimensionality of attention vectors. In particular, we investigate whether matching the full attention distribution is necessary, or whether preserving preferences among keys and the attention mass on the highest-ranked keys provides a better signal for downstream performance. We project queries and keys, while keeping values at their original dimension and leaving their dimensionality reduction to future work.

The projection is trained separately for each attention layer, rather than end-to-end, so that later layers do not learn to compensate for errors introduced by earlier ones (Chen et al., 2021b). This allows us to measure how faithfully the attention of each layer is preserved by its projection. We then apply the projections from the last layer backward, limiting the propagation of projection errors through subsequent layers, and ask how many layers can be projected before performance begins to deviate from that of the original model.

We evaluate our approach on LLaMA3.2-3B, LLaMA3.1-8B, Qwen3-4B, and Qwen2.5-7B Instruct models (Grattafiori et al., 2024; Yang et al., 2024; 2025). We measure the KL divergence of the attention distribution and the MSE of the attention output, and evaluate downstream performance on general short-context tasks (Sutawika et al., 2025) and long-context RULER (Hsieh et al., 2024), using projections trained with triplet loss and KL divergence as baselines.

We focus on two questions. First, whether KL divergence between the projected and original attention distribution is the right training objective, or whether a preference over keys, together with the mass on the highest-weighted keys, provides a better signal for downstream performance? Second, how many layers of the frozen model can be projected, applying the projections from the last layer backward, before performance begins to deviate from the original model?

Our main contributions are as follows:

- We introduce DimPO, a listwise, reference-free preference loss with a top-k cross-entropy term for head-fidelity, designed for dimensionality reduction of query and key attention vectors.
- We show that pairwise preference degrades as more layers are projected, while listwise preference maintains performance even on long-context tasks when applied to up to 50% of the layers.
- We show that better attention-distribution fidelity does not necessarily translate to better downstream performance: KL is closer in both KL divergence and attention-output MSE, while DimPO achieves better downstream performance, with the difference becoming more pronounced as more layers are projected.

2 Related Works

Attention Approximation via Projections. Transformer variants leverage projections to speed up attention. Sparse attention methods (BigBird (Zaheer et al., 2020), Longformer (Beltagy et al., 2020), SparseAxial (Ho et al., 2020)) compute selected blocks or local windows. LSH-based approaches (Reformer (Kitaev et al., 2020), KDEformer (Zandieh et al., 2023), ScatterBrain (Chen et al., 2021a), MagicPIG (Chen et al., 2025)) use locality-sensitive hashing. Mongoose (Chen et al., 2021b) learns projections via triplet loss, while MatryoshkaKV (Lin et al., 2025a) trains orthogonal projection matrices with a distillation objective. DASH-KV (Guo et al., 2026) instead uses asymmetric deep hashing to approximate attention, with KL-divergence-based distillation. Low-rank and linear attention methods (Linformer (Wang et al., 2020), Performer (Choromanski et al., 2021), Nyströmformer (Xiong et al., 2021)) reduce the attention matrix dimension. Top-k mechanisms (Unlimiformer (Bertsch et al., 2023), IceFormer (Mao et al., 2024), ZETA (Zeng et al., 2025)) combine projections and dimension reduction for efficient token selection.

Preference Optimization. Preference optimization aligns models with desired outputs using ranked feedback, often via *chosen/rejected* pairs. DPO (Rafailov et al., 2023) simplifies RLHF (Christiano et al., 2017) by removing the reward model and framing alignment as a single-stage classification, still using a reference model to prevent distributional drift. Variants include ORPO (Hong et al., 2024) (odds-ratio), SimPO (Meng et al., 2024) (average log-probability as implicit reward), CPO (Xu et al., 2024) (contrastive learning for machine translation), KTO (Ethayarajh et al., 2024) (prospect-theory utility for binary labels, still needing a reference), and listwise objectives like LiPO (Liu et al., 2025), considering multiple ranked responses while relying on a reference model. All these methods are extensions of older, generic ranking frameworks, which, however, were originally designed for discrete preference ranking tasks (Burges et al., 2005; Cao et al., 2007).

KV Cache Optimization. The KV cache memory bottleneck in long-context LLMs constrains batch size and maximum prompt length, motivating strategies to reduce key and value vectors while maintaining accuracy. H2O (Zhang et al., 2023), SnapKV (Li et al., 2024), and Keyformer (Adnan et al., 2024) use heuristics during prefilling to select tokens for decoding. Quest (Tang et al., 2024) and Loki (Singhania et al., 2024) apply dynamic sparsity during inference to reduce KV cache loading without eviction. KIVI (Liu et al., 2024) and QServe (Lin et al., 2025b) reduce KV cache via quantization.

Context Compression. Context compression reduces memory and compute while keeping essential information. Gisting (Mu et al., 2023) compresses short instructions into gist activations at once, limited by the model window. CCM (Kim et al., 2024) extends this to multi-turn conversations but not long documents. ICAE (Ge et al., 2024) and AutoCompressor (Chevalier et al., 2023) segment long contexts into chunks before compression. CEPE (Yen et al., 2024) adds a dedicated encoder with cross-attention. Token deletion methods (Jiang et al., 2024; Li et al., 2024) remove less relevant tokens. Activation Beacon (Zhang et al., 2025) compresses attention keys and values directly. LLoCO (Tan et al., 2024) decouples compression and retrieval and is orthogonal to compressor improvements.

3 Learning a Low-Dimensional Attention Projection

In this section, we formalize the problem of reducing the dimensionality of query and key vectors through learnable linear projections. Although more complex non-linear projections could provide a closer approximation of the original attention, their computational cost would be incurred at every generation step. We therefore focus on linear projections and train them in a Siamese framework (Pai et al., 2019), where the same projection is applied to each query and key. For each query, the original attention weights define preferences over its keys and provide the supervision signal for learning the projection. Within this framework, we introduce our proposed training objective, DimPO.

3.1 Problem Formulation

Let l denote a transformer attention layer with query vectors $Q_l \in \mathbb{R}^{N\times d}$ and key vectors $K_{l,q} \in \mathbb{R}^{m\times d}$ for a given $q \in Q_l$ where N is the total number of queries, m is the number of keys per each query and d is the original embedding dimension. Our goal is to learn a shared linear projection

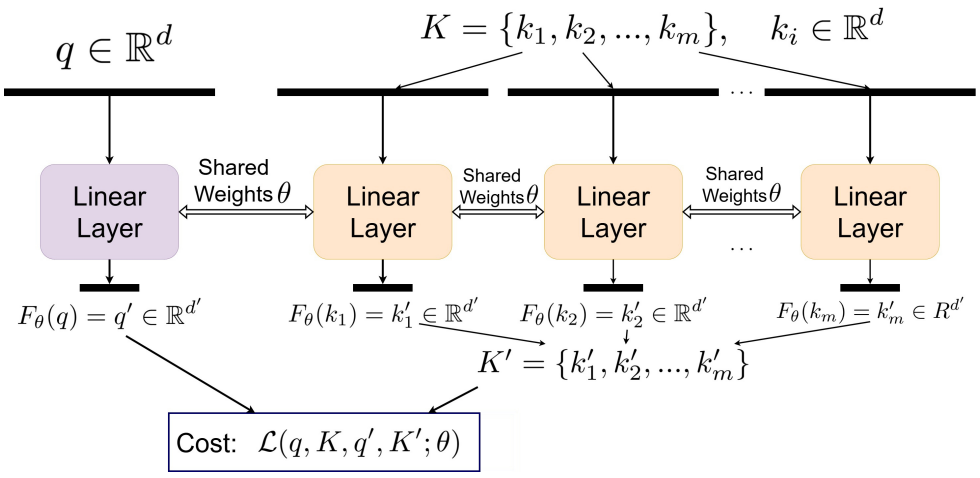


Figure 1: Siamese architecture applying the same linear map $F_\theta : \mathbb{R}^d \to \mathbb{R}^{d'}$ to a query and its keys, mapping them to a shared lower-dimensional space.

function

$$F : \mathbb{R}^d \to \mathbb{R}^{d'},$$

with $d' < d$, that maps both queries and keys into a lower-dimensional space. We compare training objectives for how well that projection preserves the behavior of the original attention.

We restrict F to a single linear projection layer, $F(x) = Wx$ with $W \in \mathbb{R}^{d' \times d}$. We train W using a Siamese framework, as illustrated in Figure 1, where each query is paired with its keys ordered by their original attention weights (for pairwise losses, only two keys are used) according to their importance for the query, derived from the original attention weight distribution. Each attention layer has its own map, trained independently of the others.

The probability assigned by the linear layer parameters θ to a given key k (either from a key pair or the full list $K_{l,q}$ associated with the current training instance) after projection $F(k) = k'$ and the corresponding query $F(q) = q'$ is computed as

$$\pi_\theta(k' \mid q') = \mathrm{Softmax}\left(\frac{q' {K'_{l,q}}^\top}{\sqrt{d'}}\right),$$

where $K'_{l,q} = F(K_{l,q})$ denotes the set of projected keys and $k' \in K'_{l,q}, k \in K_{l,q}$. For pairwise losses, this probability computation is limited to the two selected keys in $K_{l,q}$, whereas for listwise loss functions and the head term, all keys associated with the given query are used.

The value vectors are never projected and remain at full dimension.

3.2 DimPO

We now define the DimPO loss we propose, which combines a listwise preference objective with a top-k cross-entropy term to preserve both preferences over keys and the attention mass on the highest-weighted keys.

To learn a projection that preserves preferences over keys relative to a query, we use the original attention weights to define these preferences. Standard pairwise objectives (e.g., DPO (Rafailov et al., 2023), SimPO (Meng et al., 2024)) are not well suited for this setting, since each query interacts with a large set of keys, making pairwise sampling either information-losing or computationally infeasible. We therefore use a listwise objective that scores the full ordering.

Following recent preference optimization methods (Meng et al., 2024; Liu et al., 2025), the listwise term of DimPO is a reference-model-free listwise preference optimization objective with margin $\gamma \geq 0$ (the formal mathematical derivation is in Appendix A):

$$\mathcal{L}_{\text{list}}(\pi_\theta) = \mathbb{E}_{(x,y,\psi)\sim\mathcal{D}}\left[\sum_{\psi_i > \psi_j} \Delta_{i,j} \log\left(1 + e^{-(s_i - s_j - \gamma)}\right)\right], \tag{1}$$

where $\psi = (\psi_1, \ldots, \psi_K)$ denotes real-valued preference labels derived from the original attention distribution, and

$$\Delta_{i,j} = |G_i - G_j| \cdot \left| \frac{1}{D(\tau(i))} - \frac{1}{D(\tau(j))} \right|, \quad G_i = 2^{\psi_i} - 1, \quad D(\tau(i)) = \log(1 + \tau(i)).$$

Here, $\tau(i)$ is the rank of y_i under the teacher weights ψ, with rank 1 for the largest ψ_i, and the scores are defined as $s_i = \beta \log \pi_\theta(y_i \mid x)$.

The listwise term does not ask the projected softmax to put the right mass on the keys the teacher attends to most. Let p be the softmax of the projected scores over the full key list, and sort the teacher weights ψ in descending order, so that the first k positions are the head. The head term is a cross-entropy on those positions, with the normalizer still taken over the full list,

$$\mathcal{L}_{\text{head}} = -\sum_{i=1}^{k} \psi_i \log p_i. \tag{2}$$

DimPO is the sum

$$\mathcal{L}_{\text{DimPO}} = \mathcal{L}_{\text{list}} + \lambda \mathcal{L}_{\text{head}}. \tag{3}$$

Unless stated otherwise, we use $k = 64$ and $\lambda = 1$ as the default setting for DimPO. This setting provides a lightweight head-fidelity signal while keeping the listwise preference objective as the main training signal. We study different choices of k and λ in the ablation study in Section 4.4. Setting $k = 0$ and $\lambda = 0$ removes the head-fidelity term and reduces DimPO to the listwise preference objective alone.

3.3 Training Setup

We now describe how the projection F is trained and how the training data are constructed. Following Mongoose (Chen et al., 2021b), the projection is trained independently for each transformer layer.

We use `BOOKSUM` (Kryscinski et al., 2022) as a task-agnostic source of long-form text rather than training the projections on downstream task data. This allows us to test whether the learned projections generalize across downstream tasks and to separate the effect of the training objective from potential overfitting to a particular task. For training, we take the first 4096 tokens from 10 chapters of the training split, ensuring that each chapter contains at least 4096 tokens. Each chapter thus provides a set of queries paired with lists of 4096 keys. To balance coverage and computational efficiency, we subsample query instances uniformly across attention computations, resulting in a total of $40{,}960$ training instances per layer. The parameters of the projection function F are then optimized using the Adam optimizer.

For validation, we select 10 chapters from the `BOOKSUM` validation split, each containing at least 4096 tokens, and use the first 4096 tokens from each chapter. For evaluation, we select 10 chapters from the `BOOKSUM` test split in the same manner, using the first 4096 tokens of each chapter. Unlike training and validation, during evaluation, we include all query positions to obtain a complete measure of attention distribution preservation.

Before training, we first derive preference rankings from the key and query vectors. For DimPO, we additionally assign a label to each key by computing the original attention weights $\psi_i = \text{Softmax}\left(\frac{qK_{l,q}^\top}{\sqrt{d}}\right)_i$. Sorting the keys in descending order by ψ_i yields a preference ranking.

For DimPO training, we use all keys of each training instance with their exact attention weights ψ_i and the derived preference ranking. For other methods that use pairwise losses, we select a single key pair per training instance, choosing the highest-ranked key as *chosen* and the lowest-ranked key as *rejected*, following the setup used by Mongoose for triplet-loss training (Chen et al., 2021b). To verify the fairness of the comparison, we additionally perform experiments in Appendix B where pairwise methods are given a comparable number of comparisons during training, showing that they often stagnate or degrade rather than improve. We note that this setting would also substantially increase the number of training instances for pairwise methods, which may further disadvantage DimPO.

All experiments are conducted on NVIDIA RTX PRO 6000 GPUs. Detailed hyperparameters for all methods are provided in Appendix C.

Table 1: Attention fidelity of different projection approaches on Llama3.1-8B-Instruct, measured by KL divergence of the attention weights and MSE of the attention outputs, averaged across layers.

	64		**32**		**16**		**8**		**4**		**2**		**1**	
	KL	**MSE**	**KL**	**MSE**	**KL**	**MSE**	**KL**	**MSE**	**KL**	**MSE**	**KL**	**MSE**	**KL**	**MSE**
Rand	12.87	0.079	15.02	0.096	16.06	0.111	16.58	0.112	16.83	0.116	16.94	0.117	16.84	0.115
PCA	3.10	0.010	5.96	0.011	7.07	0.011	7.15	0.011	6.49	0.014	6.27	0.016	6.45	0.019
Triplet	3.29	0.009	3.77	0.009	4.07	0.009	4.25	0.010	4.32	0.009	4.33	0.010	4.43	0.010
CPO	2.06	0.007	1.64	0.005	2.59	0.007	3.44	0.009	3.95	0.010	4.30	0.010	4.53	0.010
SimPO	2.88	0.008	2.14	0.007	1.97	0.006	2.19	0.007	2.57	0.008	2.91	0.012	3.50	0.013
ORPO	2.55	0.007	1.95	0.006	1.77	0.006	1.95	0.007	2.25	0.008	2.57	0.011	3.10	0.013
DimPO (k=0, λ=0)	0.74	0.005	1.00	0.005	1.27	0.006	1.53	0.007	1.76	0.007	2.07	0.009	2.69	0.012
DimPO (k=64, λ=1)	0.57	0.002	0.85	**0.003**	1.09	0.004	1.32	**0.004**	1.46	**0.004**	**1.76**	**0.005**	**2.31**	0.006
KL	**0.26**	**0.001**	**0.59**	**0.003**	**0.92**	**0.003**	**1.18**	**0.004**	**1.41**	**0.004**	1.80	**0.005**	2.39	**0.005**

4 EXPERIMENTS

In this section, we compare different training approaches based on how well their final trained linear projection of the queries and keys preserves the frozen models Llama3.2-3B, Llama3.1-8B, Qwen2.5-7B and Qwen3-4B Instruct, all with the original dimension d=128. First, we measure two proxy metrics of this preservation. Then we measure downstream performance on five short-context tasks and long-context RULER.

We apply the projection to the last l attention layers at a target dimension d', with l=0 denoting the unmodified model. We apply the projections from the last layer backward, since projection errors introduced in earlier layers can propagate through all subsequent layers, while errors in the last layers affect fewer subsequent computations. This ordering therefore limits the propagation of projection errors and lets us increase the number of projected layers in a controlled way. We first compare the proxy metric that measures the direct similarity of the attention distributions, and then ask whether this attention fidelity translates to downstream performance.

4.1 PROXY METRICS

To see whether a projection preserves the original attention, before we look at downstream tasks, we compare two proxy metrics, defined in Appendix D. The KL divergence measures the difference between the original and projected attention distributions, while the attention-output MSE measures the difference between the corresponding attention outputs obtained by applying these attention weights to the same value vectors. The values are never projected. Lower is better on both. Table 1 reports both metrics on Llama3.1-8B-Instruct, averaged over layers, at every target dimension d'. We report the same sweeps for the other models in Appendix E.

Random projection remains far from the original attention, while the pairwise losses improve over the triplet loss but remain behind listwise approaches. Across models and target dimensions, KL generally gives the closest match to the original attention. We next ask whether this attention fidelity translates to downstream performance.

4.2 DOWNSTREAM TASKS

The proxy metrics compare the objectives based on how closely they reproduce the original attention. We then ask whether this attention fidelity translates to downstream performance. For these experiments, we fix d'=64 and vary l, the number of attention layers to which we apply the projection, to measure how the number of projected layers affects performance.

We first evaluate five zero-shot harness tasks: ARC-Challenge (acc_norm) (Clark et al., 2018), HellaSwag (acc_norm) (Zellers et al., 2019), MMLU (Hendrycks et al., 2021), TruthfulQA (mc2) (Lin et al., 2022), and WinoGrande (Sakaguchi et al., 2021), using the evaluation setup of Sutawika et al. (2025). In Figure 2, we compare all objectives on Llama3.1-8B (for a detailed breakdown by task, the remaining models, and different target dimensions d', see Appendix F). When projecting up to 40% of the layers, the pairwise preference optimization objectives retain approximately 98% of the original score, whereas the triplet baseline shows a substantially larger degradation. KL and both

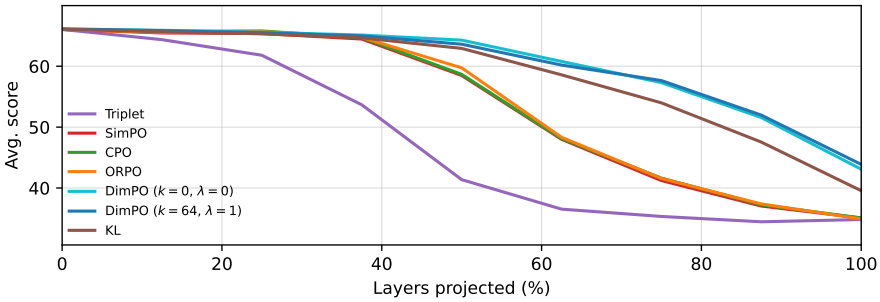


Figure 2: Average score across ARC-Challenge, HellaSwag, MMLU, TruthfulQA, and WinoGrande against the percentage of projected layers on Llama3.1-8B-Instruct, with d'=64.

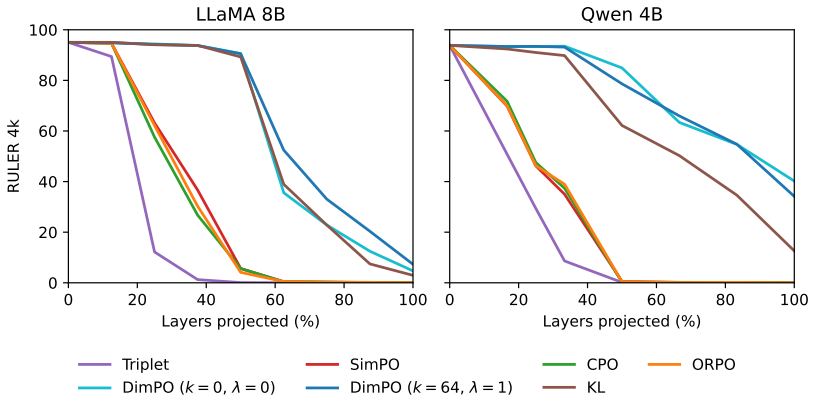


Figure 3: RULER 4k score against the percentage of projected layers for different projection objectives on Llama3.1-8B-Instruct (left) and Qwen3-4B-Instruct (right).

DimPO settings decline much more slowly. As we project more layers, DimPO stays ahead of KL, while DimPO (k=0, λ=0) and DimPO (k=64, λ=1) remain close to each other on this average.

On long-context RULER, the differences between the objectives become much more pronounced, as shown in Figure 3. The pairwise objectives degrade rapidly toward zero from the beginning. In contrast, DimPO and KL, which use the full set of keys during training, retain about 95% of the original RULER 4k score on Llama3.1-8B at 50% of the layers. Beyond 50% of the layers, the two DimPO settings remain close to each other on Llama3.1-8B, while on Qwen3-4B they both substantially outperform KL.

To see which tasks cause this difference, we examine the RULER 4k subtasks where the methods differ most beyond 50% of the layers. For Llama3.1-8B, we use l=20 and report the results in Table 2. With k=0 and λ=0, SQuAD and common-word extraction remain well above KL, while needle-in-a-haystack falls below it. Frequent-word extraction and variable tracking remain close to KL. Setting k=64 and λ=1 leaves SQuAD and common-word extraction at a similar level, while substantially improving needle-in-a-haystack, frequent-word extraction, and variable tracking. This increases the average from 35.6 to 52.4, compared to 38.9 for KL. NIAH is the mean of the eight needle tasks.

We perform the same breakdown on Qwen3-4B at l=24, with results shown in Table 3. The two DimPO settings remain close on average and both outperform KL. SQuAD, common-word extraction, and needle-in-a-haystack remain similar between the two DimPO settings. The head term substantially improves frequent-word extraction and variable tracking, where DimPO (k=64, λ=1) outperforms both DimPO (k=0, λ=0) and KL. Appendix G reports every RULER 4k subtask for Llama3.1-8B, Qwen3-4B, and Llama3.2-3B.

We compare the performance of DimPO (k=64, λ=1) at 4k and 8k RULER context lengths for Llama3.2-3B, Llama3.1-8B, and Qwen3-4B in Table 4. Although the projections are trained on 4k

Table 2: RULER 4k subtasks on Llama3.1-8B-Instruct at l=20 (62.5% of the layers), with d'=64.

Method	NIAH	SQuAD	HotpotQA	VT	CWE	FWE	RULER Avg
Base (l=0)	99.9	81.0	63.2	99.8	99.6	92.3	95.0
DimPO (k=0, λ=0)	27.1	60.1	37.2	28.8	**63.1**	56.7	35.6
DimPO (k=64, λ=1)	**51.3**	**60.3**	**38.4**	**39.6**	61.2	**71.5**	**52.4**
KL	39.1	43.6	32.0	26.7	39.0	51.5	38.9

Table 3: RULER 4k subtasks on Qwen3-4B-Instruct at l=24 (66.7% of the layers), with d'=64.

Method	NIAH	SQuAD	HotpotQA	VT	CWE	FWE	RULER Avg
Base (l=0)	100.0	76.7	62.8	100.0	99.3	81.2	93.8
DimPO (k=0, λ=0)	**75.1**	**47.3**	**36.4**	20.2	**61.3**	57.4	63.4
DimPO (k=64, λ=1)	73.0	46.2	**36.4**	**69.6**	56.5	**64.5**	**65.9**
KL	62.1	32.7	28.0	11.2	31.3	53.1	50.2

Table 4: RULER 4k and 8k scores for Llama3.2-3B-Instruct, Llama3.1-8B-Instruct, and Qwen3-4B-Instruct when applying DimPO (k=64, λ=1) with d'=64 to 25%, 38%, and 50% of the attention layers.

	LLaMA 3B		LLaMA 8B		Qwen 4B	
Layers	**4k**	**8k**	**4k**	**8k**	**4k**	**8k**
0%	92.5	87.7	95.0	93.9	93.8	93.2
25%	87.9	82.0	94.2	91.7	93.4	89.2
40%	88.2	80.1	93.8	90.6	88.6	81.6
50%	76.3	64.9	90.5	87.1	78.6	62.9

Table 5: RULER 4k and 8k average for combining DimPO with SnapKV or KIVI on Llama3.1-8B-Instruct with d'=64. Columns show the percentage of layers projected, with l=0, 8, 12, 16. The l=0 column corresponds to using SnapKV or KIVI without DimPO.

	0% (l=0)		**25%** (l=8)		**38%** (l=12)		**50%** (l=16)	
Method	**4k**	**8k**	**4k**	**8k**	**4k**	**8k**	**4k**	**8k**
Base	95.0	93.9	94.2	91.7	93.8	90.6	90.5	87.1
KIVI 2-bit	94.5	92.6	91.6	86.5	89.3	84.9	83.4	76.2
KIVI 4-bit	95.0	93.9	94.1	91.7	93.8	90.9	90.6	86.7
SnapKV 1024	95.0	93.9	94.1	91.7	93.7	90.6	90.5	87.0
SnapKV 2048	95.0	93.9	94.1	91.7	93.7	90.6	90.5	87.0

contexts from BOOKSUM, they retain reasonable performance when evaluated at 8k context. Up to about 40% of the layers, all three models retain most of their score at both context lengths. At 50% of the layers, Llama3.1-8B still retains about 95% of its 4k score and about 93% of its 8k score. However, Llama3.2-3B and Qwen3-4B show a larger drop.

4.3 Compatibility with Compression Methods

Although we do not propose an end-to-end cache compression method, we ask whether reducing the query and key dimension can be combined with existing compression methods that operate along a different axis, or whether the approximation errors compound when the methods are combined. We combine DimPO (k=64, λ=1) at d'=64 with SnapKV (Li et al., 2024) and KIVI (Liu et al., 2024) on Llama3.1-8B, with results shown in Table 5. The l=0 entry corresponds to the other compression method alone, while the later columns add the projection to the last l layers. SnapKV at both budgets and KIVI at 4-bit follow the corresponding base results, suggesting that the two compression methods can be combined without an additional performance loss. KIVI at 2-bit behaves differently and degrades faster when combined with the projection. We report inference latency for the base model, SnapKV, and DimPO in Appendix H.

Table 6: Ablation of DimPO on RULER 4k and attention KL at $d'=64$. Results are reported at $l=20$ for Llama3.1-8B-Instruct and $l=24$ for Qwen3-4B-Instruct.

	LLaMA 8B		Qwen 4B	
Objective	**RULER 4k**	**KL**	**RULER 4k**	**KL**
DimPO ($k=0$, $\lambda=0$)	35.6	0.74	63.4	0.73
KL	38.9	**0.26**	50.2	**0.27**
DimPO ($k=0$, $\lambda=0$) + KL	35.8	0.27	44.4	**0.27**
DimPO ($k=32$, $\lambda=1$)	48.5	0.68	62.9	0.65
DimPO ($k=64$, $\lambda=1$)	52.4	0.57	65.9	0.53
DimPO ($k=128$, $\lambda=1$)	49.4	0.43	60.9	0.44
DimPO ($k=256$, $\lambda=1$)	52.3	0.38	63.3	0.36
DimPO ($k=64$, $\lambda=0.5$)	47.4	0.52	70.0	0.54
DimPO ($k=64$, $\lambda=2$)	**55.5**	0.56	**73.3**	0.54
DimPO ($k=64$, $\lambda=4$)	51.8	0.56	64.7	0.54

4.4 Ablation of DimPO

For our main experiments, we use DimPO ($k=64$, $\lambda=1$), with this setting fixed before evaluating RULER. The proxy metrics and RULER do not necessarily favor the same setting, so here we separately study the effect of k and λ. Table 6 reports RULER 4k at the separating depth together with the attention KL at $d'=64$. Adding the full KL objective to the listwise term substantially reduces the KL divergence, but does not improve RULER. Increasing k generally reduces the KL divergence, but does not consistently improve RULER. Varying λ shows a similar separation: $\lambda=2$ gives the highest RULER score for both models, while its attention KL remains substantially higher than that of the KL objective. This further shows that the setting with the closest attention match is not the one with the highest RULER score.

5 Conclusion

We approached dimensionality reduction of attention from a preference optimization perspective, treating the keys preferred by the original attention as supervision for learning a lower-dimensional query-key space. Pairwise preference objectives outperform the triplet baseline and retain about 98% of the short-context score when projecting up to 40% of the layers, but degrade as more layers are projected and on long-context tasks. We introduce DimPO, which combines listwise preference optimization with a top-k cross-entropy term to preserve preferences over keys and attention mass on the highest-weighted keys. We train each projection independently per layer, which isolates the direct effect of the objective on attention preservation without allowing other layers to compensate for projection errors. KL divergence produces the closest reconstruction of the original attention and attention output, yet DimPO achieves better downstream performance, retaining about 95% of the original RULER 4k score on Llama3.1-8B when applied to 50% of the layers. These results show that the objective that best reconstructs attention is not necessarily the objective that best preserves model behavior, and suggest that under dimensionality reduction, preserving the relative importance and concentration of attention can be more useful than reproducing the attention distribution.

6 Limitations

Our experiments use instruction-tuned Llama and Qwen models, with Llama3.1-8B as the largest, so further evaluation is needed for larger models and other architectures. The projections are trained on `BOOKSUM` with 4096-token sequences, and their behavior on code, mathematical text, or other training distributions remains unclear. We leave value vectors at their original dimension because DimPO defines preferences from query-key scores, providing no direct criterion for ordering or supervising values; extending the method to value projection is therefore left to future work. The choice of k and λ must also be selected empirically on a development set, particularly when changing the training context length or data distribution. More broadly, this work focuses on understanding how to preserve attention under dimensionality reduction rather than proposing a complete compression method. We view these insights and DimPO as a basis for developing more efficient compression methods in future work.

AI USE STATEMENT

In this work, we used generative AI tools for assistance with debugging and writing code, as well as for manuscript writing assistance and checking its style, grammar, and writing quality. We have not used generative AI tools to generate synthetic data, to develop theoretical models or conceptual frameworks, to formulate mathematical claims, to provide critical ingredients for proving mathematical claims, to assist in the writing of proofs, to propose or refine hypotheses, to design or provide feedback on the research methodology or the experiments, to clean or reformat the datasets, or to interpret the results. Translation and qualitative or thematic analysis are not applicable to this work. We have reviewed all AI-assisted work and verified AI-assisted code through testing. We take responsibility for the final content of this work, including text, claims, code, and other artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The projection, the listwise term, and the head term are specified in Section 3.2 and derived in Appendix A. The training data, query subsampling, and optimizer are described in Section 3.3. The operating point k=64, λ=1 and its ablation are reported in Table 6. Task scores are computed using the released evaluation protocols of `lm-evaluation-harness` v0.4.9.1, including its RULER evaluation. The code is included as supplementary material. For this submission, the repository link is anonymized.

REFERENCES

Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant J. Nair, Ilya Soloveychik, and Purushotham Kamath. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. In P. Gibbons, G. Pekhimenko, and C. De Sa (eds.), *Proceedings of Machine Learning and Systems*, volume 6, pp. 114–127, 2024. URL https://proceedings.mlsys.org/paper_files/paper/2024/file/48fecef47b19fe501d27d338b6d52582-Paper-Conference.pdf.

Alan Agresti. *Categorical data analysis. Hoboken*. NJ: wiley, 2002.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020. URL https://arxiv.org/abs/2004.05150.

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew Gormley. Unlimiformer: Long-range transformers with unlimited length input. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 35522–35543. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6f9806a5adc72b5b834b27e4c7c0df9b-Paper-Conference.pdf.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pp. 144–152, New York, NY, USA, 1992. Association for Computing Machinery. ISBN 089791497X. doi: 10.1145/130385.130401. URL https://doi.org/10.1145/130385.130401.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. ISSN 00063444, 14643510. URL http://www.jstor.org/stable/2334029.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pp. 89–96, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102363. URL https://doi.org/10.1145/1102351.1102363.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 129–136, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273513. URL https://doi.org/10.1145/1273496.1273513.

Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 17413–17426. Curran Associates, Inc., 2021a. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/9185f3ec501c674c7c788464a36e7fb3-Paper.pdf.

Beidi Chen, Zichang Liu, Binghui Peng, Zhaozhuo Xu, Jonathan Lingjie Li, Tri Dao, Zhao Song, Anshumali Shrivastava, and Christopher Re. {MONGOOSE}: A learnable {lsh} framework for efficient neural network training. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=wWK7yXkULyh.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. MagicPIG: LSH sampling for efficient LLM generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=ALzTQUgW8a.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=kp1U6wBPXq.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization, 2024. URL https://arxiv.org/abs/2402.01306.

Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=uREj4ZuGJE.

Aaron Grattafiori et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Jinyu Guo, Zhihan Zhang, Jiehui Xie, Md. Tamim Iqbal, Dongshen Han, Lik-Hang Lee, Sung-Ho Bae, Jie Zou, Yang Yang, and Chaoning Zhang. Dash-kv: Accelerating long-context llm inference via asymmetric kv cache hashing, 2026. URL https://arxiv.org/abs/2604.19351.

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. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers, 2020. URL https://openreview.net/forum?id=H1e5GJBtDr.

Jiwoo Hong, Noah Lee, and James Thorne. ORPO: Monolithic preference optimization without reference model. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11170–11189, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.626. URL https://aclanthology.org/2024.emnlp-main.626/.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=kIoBbc76Sy.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1658–1677, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.91. URL https://aclanthology.org/2024.acl-long.91/.

Jang-Hyun Kim, Junyoung Yeom, Sangdoo Yun, and Hyun Oh Song. Compressed context memory for online language model interaction. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=64kSvC4iPg.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB.

Wojciech Kryscinski, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. BOOKSUM: A collection of datasets for long-form narrative summarization. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 6536–6558, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.488. URL https://aclanthology.org/2022.findings-emnlp.488/.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/28ab418242603e0f7323e54185d19bde-Paper-Conference.pdf.

Bokai Lin, Zihao Zeng, Zipeng Xiao, Siqi Kou, TianQi Hou, Xiaofeng Gao, Hao Zhang, and Zhijie Deng. Matryoshkakv: Adaptive kv compression via trainable orthogonal projection. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 86669–86690, 2025a. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/d7b351608d824a4680344a02b180a947-Paper-Conference.pdf.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*,

pp. 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229/.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. QServe:w4a8KV4 quantization and system co-design for efficient LLM serving. In *Eighth Conference on Machine Learning and Systems*, 2025b. URL https://openreview.net/forum?id=1FfmStySS1.

Tianqi Liu, Zhen Qin, Junru Wu, Jiaming Shen, Misha Khalman, Rishabh Joshi, Yao Zhao, Mohammad Saleh, Simon Baumgartner, Jialu Liu, Peter J Liu, and Xuanhui Wang. LiPO: Listwise preference optimization through learning-to-rank. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 2404–2420, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.121. URL https://aclanthology.org/2025.naacl-long.121/.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen (Henry) Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: a tuning-free asymmetric 2bit quantization for kv cache. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Yuzhen Mao, Martin Ester, and Ke Li. Iceformer: Accelerated inference with long-sequence transformers on CPUs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6RR3wU4mSZ.

Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 124198–124235. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/e099c1c9699814af0be873a175361713-Paper-Conference.pdf.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=2DtxPCL3T5.

Gautam Pai, Ronen Talmon, Alex Bronstein, and Ron Kimmel. Dimal: Deep isometric manifold learning using sparse geodesic sampling. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 819–828, 2019. doi: 10.1109/WACV.2019.00092.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In D. Song, M. Carbin, and T. Chen (eds.), *Proceedings of Machine Learning and Systems*, volume 5, pp. 606–624. Curan, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/file/c4be71ab8d24cdfb45e3d06dbfca2780-Paper-mlsys2023.pdf.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 53728–53741. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a85b405ed65c6477a4fe8302b5e06ce7-Paper-Conference.pdf.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. Loki: Low-rank keys for efficient sparse attention. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 16692–16723. Curran Associates, Inc.,

2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/1e027da6bec9ceb2ec37951ceeccae93-Paper-Conference.pdf.

Lintang Sutawika, Hailey Schoelkopf, Leo Gao, Baber Abbasi, Stella Biderman, Jonathan Tow, ben fattori, Charles Lovering, farzanehnakhaee70, Jason Phang, Anish Thite, Fazz, Aflah, Niklas, Thomas Wang, sdtblck, nopperl, gakada, tttyuntian, researcher2, Julen Etxaniz, Chris, Hanwool Albert Lee, Leonid Sinev, Zdeněk Kasner, Kiersten Stokes, Khalid, KonradSzafer, Jeffrey Hsu, and Anjor Kanekar. Eleutherai/lm-evaluation-harness: v0.4.9.1, August 2025. URL https://doi.org/10.5281/zenodo.16737642.

Sijun Tan, Xiuyu Li, Shishir G Patil, Ziyang Wu, Tianjun Zhang, Kurt Keutzer, Joseph E. Gonzalez, and Raluca Ada Popa. LLoCO: Learning long contexts offline. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17605–17621, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.975. URL https://aclanthology.org/2024.emnlp-main.975/.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: query-aware sparsity for efficient long-context llm inference. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Heather Turner and David Firth. Bradley-terry models in r: The bradleyterry2 package. *Journal of Statistical Software*, 48(9):1–21, 2012. doi: 10.18637/jss.v048.i09. URL https://www.jstatsoft.org/index.php/jss/article/view/v048i09.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020. URL https://arxiv.org/abs/2006.04768.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14138–14148, May 2021. doi: 10.1609/aaai.v35i16.17664. URL https://ojs.aaai.org/index.php/AAAI/article/view/17664.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. Contrastive preference optimization: pushing the boundaries of llm performance in machine translation. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

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, 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. *CoRR*, abs/2412.15115, 2024. URL https://doi.org/10.48550/arXiv.2412.15115.

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, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Howard Yen, Tianyu Gao, and Danqi Chen. Long-context language modeling with parallel context encoding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2588–2610, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.142. URL https://aclanthology.org/2024.acl-long.142/.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17283–17297. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.

Amir Zandieh, Insu Han, Majid Daliri, and Amin Karbasi. KDEformer: Accelerating transformers via kernel density estimation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 40605–40623. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/zandieh23a.html.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

QIUHAO Zeng, Jerry Huang, Peng Lu, Gezheng Xu, Boxing Chen, Charles Ling, and Boyu Wang. ZETA: Leveraging z-order curves for efficient top-k attention. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=j9VVzueEbG.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. Long context compression with activation beacon. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=1eQT9OzfNQ.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6ceefa7b15572587b78ecfcebb2827f8-Paper-Conference.pdf.

A Derivation of the Listwise Component of DimPO

Preference optimization research has predominantly focused on pairwise comparisons (Rafailov et al., 2023; Hong et al., 2024; Xu et al., 2024; Meng et al., 2024). While effective for many tasks, this is not practical for dimensionality reduction of query-key interactions, where each query is compared against many keys. Using a pairwise loss, one can either select a single positive-negative key pair for each query, losing information about relationships with other keys, or consider all possible key pairs, creating $m(m-1)/2$ training instances for each query for m context tokens, which dramatically increases computational and memory costs making the training infeasible to complete. For this reason, it is more practical in this setting to adopt listwise preference optimization losses. One of the most popular pairwise preference optimization methods is DPO (Rafailov et al., 2023):

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\Big[\log\sigma\big(\beta\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta\log\frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\big)\Big],$$

which fits naturally into the Bradley-Terry (BT) (Bradley & Terry, 1952) ranking model:

$$p(y_w \succ y_l \mid x) = \sigma\big(r(x, y_w) - r(x, y_l)\big),$$

where y_w denotes the preferred response, y_l the non-preferred response, and $r(x, y)$ is the reward function. LiPO (Liu et al., 2025) generalizes this BT model to a list of responses $y = (y_1, \ldots, y_K)$:

$$p(y_1 \succ y_2 \succ \cdots \succ y_K \mid x) = \prod_{i=1}^{K} \frac{\exp(s_i)}{\sum_{j=i}^{K}\exp(s_j)},$$

where $s_i = r(x, y_i)$ denotes the score of response y_i. This reduces exactly to the pairwise BT model when $K = 2$. In the formulation of listwise loss LiPO, the training dataset consists of lists of responses with corresponding real-valued labels $\psi = (\psi_1, \ldots, \psi_K)$, and a ranking loss is applied over all pairs within the list:

$$\mathcal{L}_{\lambda\text{-loss}}(\pi_\theta) = \mathbb{E}_{(x,y,\psi)\sim\mathcal{D}}\left[\sum_{\psi_i>\psi_j}\Delta_{i,j}\log\big(1 + e^{-(s_i-s_j)}\big)\right],$$

where

$$\Delta_{i,j} = |G_i - G_j| \cdot \left|\frac{1}{D(\tau(i))} - \frac{1}{D(\tau(j))}\right|, \quad G_i = 2^{\psi_i} - 1, \quad D(\tau(i)) = \log(1 + \tau(i)).$$

Here, $\tau(i)$ denotes the rank position of y_i in the permutation induced by the scores s, and the scores are defined as

$$s_i = \beta\log\frac{\pi_\theta(y_i \mid x)}{\pi_{\text{ref}}(y_i \mid x)},$$

with $\beta > 0$ controlling the sharpness of the preference optimization.

In our setting, the reference model π_{ref} is not available. Furthermore, SimPO argues that using a reference model during training is inconsistent with inference, in which no reference is present, which can generate inaccurate responses (Meng et al., 2024). However, in our setting, we treat π_{ref} as a uniform distribution and approximate it with a constant $1/c$, which cancels in the log-ratio in the $e^{-(s_i-s_j)}$ term of the sum in the LiPO loss equation

$$\begin{aligned} s_i - s_j &= \beta\log\frac{\pi_\theta(y_i \mid x)}{\pi_{\text{ref}}(y_i \mid x)} - \beta\log\frac{\pi_\theta(y_j \mid x)}{\pi_{\text{ref}}(y_j \mid x)} && (4)\\ &= \beta\Big(\log\pi_\theta(y_i \mid x) - \log\pi_{\text{ref}}(y_i \mid x) - \log\pi_\theta(y_j \mid x) + \log\pi_{\text{ref}}(y_j \mid x)\Big) && (5)\\ &= \beta\Big(\log\pi_\theta(y_i \mid x) - \log\pi_\theta(y_j \mid x)\Big), && (6) \end{aligned}$$

yielding

$$s_i = \beta\log\pi_\theta(y_i \mid x).$$

Finally, following SimPO's BT adaptation, which introduces a target reward margin $\gamma \geq 0$ to ensure that the score difference between better and worse responses is at least γ (a margin known to improve

generalization capabilities of classifiers (Boser et al., 1992; Cortes & Vapnik, 1995; Agresti, 2002; Turner & Firth, 2012)) the pairwise margin-adjusted BT model is defined as

$$p(y_w \succ y_l \mid x) = \sigma\big(r(x, y_w) - r(x, y_l) - \gamma\big).$$

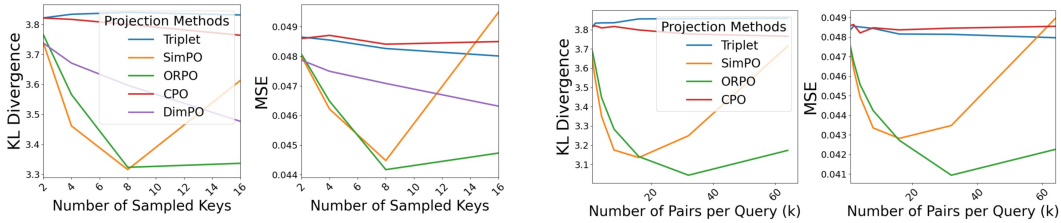
Building on this, we define the listwise component of DimPO:

$$\mathcal{L}_{list}(\pi_\theta) = \mathbb{E}_{(x,y,\psi)\sim\mathcal{D}}\left[\sum_{\psi_i > \psi_j} \Delta_{i,j} \log\big(1 + e^{-(s_i - s_j - \gamma)}\big)\right], \tag{7}$$

with $\Delta_{i,j}$, G_i and $D(\tau(i)) = \log(1 + \tau(i))$ as above, where $\tau(i)$ is the rank of y_i under the teacher weights ψ, with rank 1 for the largest ψ_i, and $s_i = \beta \log \pi_\theta(y_i|x)$. This formulation provides a reference-model-free, listwise, margin-aware preference optimization objective, capturing all key-query interactions efficiently. When $K = 2$, the loss reduces to a pairwise logistic loss with margin γ and a positive weight $\Delta_{i,j}$. This corresponds to the SimPO formulation, except that the pair is weighted by $\Delta_{i,j}$ rather than equally.

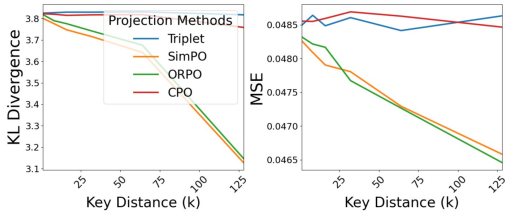
B Effect of Key-Pair Selection on Pairwise Losses

The observation that the listwise term outperforms the pairwise losses raises the question of whether this advantage comes from a better inductive bias or simply from receiving more training signal. Although all methods use the same number of training instances, the listwise term uses all keys associated with a given query, whereas pairwise losses rely on only two keys (*chosen* and *rejected*). Providing every possible key pair to pairwise methods would be computationally prohibitive due to the combinatorial growth in training examples, but it is still informative to study whether their weaker performance is caused by this information bottleneck. To this end, we perform three controlled experiments on the attention layers of Llama3.2-1B-Instruct, Llama3.2-3B-Instruct, Llama3.1-8B-Instruct, and Qwen3-4B-Instruct, reporting averages across all models. For efficiency, training is performed on 128-token subsequences sampled from 10 chapters, and evaluation uses a validation set of 10 full-length chapters (4096 tokens each) from BOOKSUM.



(a) All key pairs

(b) Multiple distinct pairs



(c) Level of key diversity

Figure 4: Effect of key-pair selection on pairwise losses. (a) All key pairs from a sampled subset. The DimPO curve is the listwise term, DimPO (k=0, λ=0), and it keeps improving as more keys are included, whereas pairwise methods plateau or degrade. (b) Training with multiple non-overlapping pairs per query does not yield improvements, suggesting that additional pairs introduce noise rather than a meaningful signal. (c) Increasing the diversity between chosen and rejected keys consistently improves performance, indicating that pairwise losses benefit most from highly diverse key pairs.

All Key Pairs. We first investigate pairwise methods and the listwise term by sampling $k \in \{2, 4, 8, 16\}$ keys from the 128 available keys per sequence and using all possible key pairs within this subset for training. Figure 4a reports the KL divergence between the original and projected attention weights, and the MSE between the original and projected attention outputs, averaged across all attention layers and target dimensions $d' \in \{1, 2, 4, 8, 16, 32, 64\}$. The listwise term benefits consistently from having access to more keys, while Triplet and CPO remain largely unaffected. ORPO and SimPO initially seem to gain from additional keys, but their performance quickly plateaus or even degrades, suggesting that the increased combinatorial complexity hinders training rather than helping. This emphasizes that even if pairwise methods were trained on all possible key pairs, they would likely still fall short of DimPO's performance, highlighting the advantage of its listwise formulation.

Multiple Distinct Pairs. We next investigate training with multiple distinct pairs per query, ensuring that no key is used more than once. For each query, we generate $k \in \{1, 2, 4, 8, 16, 32, 64\}$ training pairs, where each chosen key comes from the top half of the attention-weight ranking and each rejected key from the bottom half. Figure 4b shows that pairwise methods consistently perform best when using only a single pair per query, confirming that adding more pairs introduces noise rather than additional useful signal.

Level of Key Diversity. The main tables train pairwise methods by maximizing the diversity between chosen and rejected keys. One might wonder whether using more similar key pairs could be beneficial. Figure 4c uses a rank distance $k \in \{1, 8, 16, 32, 64, 127\}$ between the chosen and the rejected key. CPO and Triplet remain largely unaffected, while SimPO and ORPO require highly diverse key pairs to reach the performance reported in the main table.

C HYPERPARAMETERS OF DIMENSIONALITY REDUCTION APPROACHES

All methods are trained using the Adam optimizer, with learning rates selected via validation on a held-out split of BOOKSUM. Final configurations are reported in Table 7, while Table 8 summarizes all explored hyperparameter values.

The approximate training time per attention layer for pairwise methods (Triplet, SimPO, ORPO, CPO) is $\sim$10s per layer when sampling a single key pair per instance. The listwise term requires $\sim$5 minutes per layer due to full listwise likelihood computations. If pairwise methods were extended to capture the same amount of comparison information as the listwise setup, this would require approximately $\sim$970 days per layer, making it infeasible in practice. Importantly, all methods are trained independently per transformer layer, allowing full parallelization across layers, and are trained only once prior to deployment.

Table 7: Final hyperparameter settings for all preference optimization methods.

Method	β	γ	Learning rate	Batch size	Time/layer
DimPO	1.0	0.0001	0.0001	1	$\sim$5 min
SimPO	1.0	1.0	0.001	32	$\sim$10 s
Triplet	–	–	0.0001	32	$\sim$10 s
ORPO	0.1	–	0.001	32	$\sim$10 s
CPO	1.0	0.1	0.0001	32	$\sim$10 s

Table 8: Hyperparameter values explored during tuning for all methods, where applicable.

Hyperparameter	Tested values
β	0.0001, 0.001, 0.01, 0.1, 1.0, 2, 2.5, 5.0
γ	0, 0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0
Learning rate	1e-5, 1e-4, 1e-3, 1e-2, 0.1
Batch size	1, 2, 4, 8, 16, 32, 64

D ATTENTION FIDELITY METRICS

We evaluate the quality of the learned projection F using two complementary metrics, averaged over all N evaluation instances $(q, K_{l,q})$ for all $q \in Q_l$, where $V_{l,q} \in \mathbb{R}^{m \times d}$ denotes the value vectors for layer l associated with query q.

1. **Attention weights KL Divergence:** the average Kullback–Leibler divergence between the original attention distribution and the projected one for each query-key set:

$$\mathrm{KL} = \frac{1}{N} \sum_{q \in Q_l} \mathrm{KL}\left(\mathrm{Softmax}\Big(\frac{q K_{l,q}^\top}{\sqrt{d}} \Big), \mathrm{Softmax}\Big(\frac{F(q) F(K_{l,q})^\top}{\sqrt{d'}} \Big) \right), \tag{8}$$

2. **Attention output MSE:** the average mean squared error between the original attention output and the output after projection for each query-key set:

$$\mathrm{MSE} = \frac{1}{N} \sum_{q \in Q_l} \frac{1}{d} \left\| \mathrm{Softmax}\Big(\frac{q K_{l,q}^\top}{\sqrt{d}} \Big) V_{l,q} - \mathrm{Softmax}\Big(\frac{F(q) F(K_{l,q})^\top}{\sqrt{d'}} \Big) V_{l,q} \right\|_2^2. \tag{9}$$

E PROXY METRICS ON THE OTHER MODELS

Tables 9–11 report the same proxy metrics as Table 1, averaged over layers, for every model other than Llama3.1-8B. Lower is better, and the minimum in each column is bold. On Llama3.2-3B, Qwen3-4B and Qwen2.5-7B the KL objective is the closest match at $d'=64$ on both KL and MSE. On Llama3.2-3B, DimPO ($k=64$, $\lambda=1$) is ahead on KL at $d'=2$ and $d'=1$, and the two tie on MSE at $d'=8$, $d'=4$ and $d'=2$. On Qwen3-4B and Qwen2.5-7B the KL objective is the closest match at every target dimension.

Table 9: Attention fidelity of different projection approaches on Llama3.2-3B-Instruct, measured by attention-weight KL divergence and attention-output MSE, averaged across layers.

	64		**32**		**16**		**8**		**4**		**2**		**1**	
	KL	**MSE**	**KL**	**MSE**	**KL**	**MSE**	**KL**	**MSE**	**KL**	**MSE**	**KL**	**MSE**	**KL**	**MSE**
Rand	12.56	0.084	14.50	0.104	15.78	0.119	16.52	0.129	16.60	0.128	16.82	0.132	16.70	0.126
PCA	2.65	0.011	5.71	0.013	7.28	0.013	7.57	0.013	6.66	0.016	6.44	0.019	6.49	0.021
Triplet	3.19	0.014	3.63	0.014	3.94	0.014	4.12	0.014	4.13	0.014	4.19	0.015	4.28	0.014
CPO	2.20	0.009	1.69	0.007	2.19	0.009	3.36	0.014	3.86	0.014	4.17	0.014	4.39	0.014
SimPO	2.90	0.010	2.12	0.009	1.95	0.008	2.20	0.009	2.58	0.011	2.92	0.014	3.69	0.019
ORPO	2.72	0.010	1.89	0.008	1.71	0.007	1.96	0.007	2.31	0.010	2.65	0.014	3.08	0.017
DimPO ($k=0$, $\lambda=0$)	0.67	0.005	0.97	0.006	1.29	0.007	1.57	0.008	1.83	0.009	2.10	0.011	2.62	0.014
DimPO ($k=64$, $\lambda=1$)	0.55	0.003	0.81	0.004	1.09	0.005	1.32	**0.005**	1.49	**0.005**	**1.74**	**0.006**	**2.31**	0.008
KL	**0.23**	**0.002**	**0.55**	**0.003**	**0.91**	**0.004**	**1.19**	**0.005**	**1.42**	**0.005**	1.80	**0.006**	2.33	**0.007**

Table 10: Attention fidelity of different projection approaches on Qwen3-4B-Instruct, measured by attention-weight KL divergence and attention-output MSE, averaged across layers.

	64		**32**		**16**		**8**		**4**		**2**		**1**	
	KL	**MSE**	**KL**	**MSE**	**KL**	**MSE**	**KL**	**MSE**	**KL**	**MSE**	**KL**	**MSE**	**KL**	**MSE**
Rand	14.07	1.269	14.85	1.577	15.49	1.837	15.75	2.045	15.89	2.131	16.01	2.173	15.88	1.929
PCA	4.42	0.217	6.51	0.267	8.01	0.360	9.16	0.503	10.02	0.554	10.34	0.533	10.36	0.534
Triplet	2.50	0.149	2.91	0.169	3.20	0.174	3.38	0.177	3.55	0.179	3.63	0.183	3.69	0.180
CPO	2.52	0.216	2.14	0.187	2.55	0.190	3.04	0.161	3.38	0.184	3.63	0.184	3.80	0.193
SimPO	1.44	0.115	1.74	0.156	2.21	0.146	2.85	0.177	3.56	0.224	3.95	0.202	4.65	0.371
ORPO	1.39	0.116	1.65	0.143	2.02	0.134	2.52	0.174	2.96	0.208	3.17	0.180	3.59	0.323
DimPO ($k=0$, $\lambda=0$)	0.73	0.052	1.15	0.076	1.57	0.089	1.89	0.107	2.07	0.117	2.27	0.132	2.74	0.223
DimPO ($k=64$, $\lambda=1$)	0.53	0.037	0.90	0.061	1.29	0.087	1.62	0.098	1.85	0.115	2.14	0.118	2.52	0.136
KL	**0.27**	**0.019**	**0.66**	**0.044**	**1.11**	**0.066**	**1.48**	**0.089**	**1.77**	**0.096**	**2.07**	**0.107**	**2.48**	**0.127**

Table 11: Attention fidelity of different projection approaches on Qwen2.5-7B-Instruct, measured by attention-weight KL divergence and attention-output MSE, averaged across layers.

	64		**32**		**16**		**8**		**4**		**2**		**1**	
	KL	**MSE**	**KL**	**MSE**	**KL**	**MSE**	**KL**	**MSE**	**KL**	**MSE**	**KL**	**MSE**	**KL**	**MSE**
Rand	14.65	1.181	15.58	1.459	15.82	1.594	16.02	1.789	16.08	1.911	16.09	1.739	15.96	1.705
PCA	3.65	0.826	7.49	0.891	10.39	0.936	11.32	0.939	11.40	0.985	11.43	1.138	11.41	1.205
Triplet	2.51	0.104	2.95	0.085	3.22	0.088	3.44	0.088	3.52	0.090	3.64	0.093	3.69	0.094
CPO	2.53	0.120	2.24	0.087	2.72	0.088	3.17	0.090	3.55	0.092	3.73	0.095	3.87	0.101
SimPO	1.58	0.094	1.86	0.083	2.38	0.096	3.21	0.129	3.95	0.146	4.45	0.178	4.69	0.210
ORPO	1.61	0.096	1.86	0.079	2.32	0.091	2.92	0.111	3.43	0.134	3.69	0.141	3.82	0.141
DimPO ($k=0$, $\lambda=0$)	0.78	0.057	1.15	0.078	1.62	0.102	2.07	0.111	2.43	0.111	2.75	0.126	3.10	0.736
DimPO ($k=64$, $\lambda=1$)	0.59	0.034	0.95	0.050	1.40	0.065	1.82	0.073	2.12	0.077	2.41	0.083	2.88	0.089
KL	**0.29**	**0.021**	**0.70**	**0.041**	**1.19**	**0.057**	**1.66**	**0.068**	**2.02**	**0.074**	**2.38**	**0.079**	**2.84**	**0.087**

F HARNESS TASKS

Tables 12–15 report the average of ARC-Challenge, HellaSwag, MMLU, TruthfulQA and WinoGrande for DimPO (k=64, λ=1). Each cell is that average together with the percentage of the KV cache removed at the corresponding d' and l. Storing keys and values at their original dimension in every layer is a reduction of 0%. The value vectors are never projected, so discarding the keys entirely would still leave half of the cache, and 50% is the largest reduction this projection can reach. Projecting the keys of the last l layers from dimension d to d' removes

$$50 \cdot \frac{l}{N}\left(1 - \frac{d'}{d}\right) \tag{10}$$

percent of the KV cache, where N is the number of layers. Figures 5–8 show the five tasks separately for each model. Figure 9 is the average of those five tasks, with one panel per model. Across every measured d' and l, the score at a given reduction is the highest cell, and a point is kept only when that score is still higher than the scores at larger reductions, so the curve falls as the reduction grows.

Table 12: Average of ARC-Challenge, HellaSwag, MMLU, TruthfulQA and WinoGrande on Llama3.1-8B-Instruct. Each cell is the score and the percentage of the KV cache removed, for DimPO (k=64, λ=1).

d' / l	0	2	4	8	12	16	20	24	28	30	32
64	66.2/0	66.0/1.6	65.8/3.1	65.5/6.2	65.0/9.4	63.6/12.5	60.2/15.6	57.6/18.8	52.0/21.9	48.0/23.4	43.9/25
16	66.2/0	65.1/2.7	65.2/5.5	64.3/10.9	62.4/16.4	53.7/21.9	43.7/27.3	37.9/32.8	34.7/38.3	34.7/41	34.7/43.8
4	66.2/0	64.9/3	64.5/6.1	62.4/12.1	55.2/18.2	42.2/24.2	37.0/30.3	34.9/36.3	34.9/42.4	34.8/45.4	35.1/48.4
1	66.2/0	64.8/3.1	64.2/6.2	60.3/12.4	52.3/18.6	39.7/24.8	35.2/31	34.5/37.2	34.8/43.4	35.4/46.5	35.8/49.6

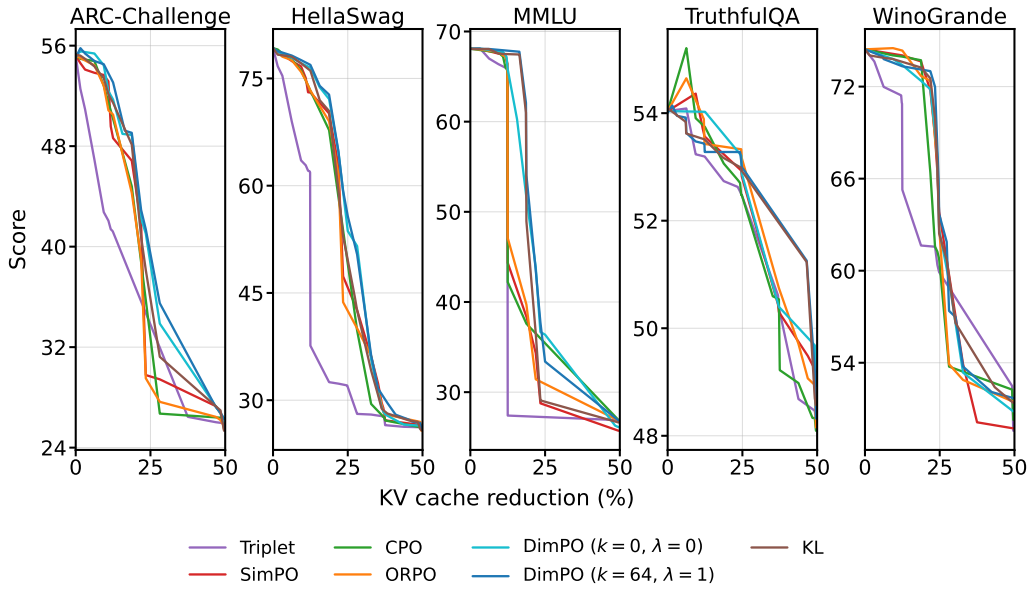


Figure 5: Harness task scores for different projection methods on Llama3.1-8B-Instruct against the percentage of the KV cache removed.

Table 13: Average of ARC-Challenge, HellaSwag, MMLU, TruthfulQA and WinoGrande on Qwen3-4B-Instruct. Each cell is the score and the percentage of the KV cache removed, for DimPO (k=64, λ=1).

d' / l	0	2	4	6	9	12	15	21	27	32	36
64	65.7/0	65.7/1.4	65.6/2.8	65.7/4.2	65.5/6.2	65.1/8.3	63.7/10.4	60.9/14.6	57.3/18.8	56.3/22.2	55.0/25
16	65.7/0	64.7/2.4	64.4/4.9	63.7/7.3	62.5/10.9	56.4/14.6	48.5/18.2	41.5/25.5	38.2/32.8	35.5/38.9	34.1/43.8
4	65.7/0	64.2/2.7	63.1/5.4	62.0/8.1	59.7/12.1	49.5/16.1	43.0/20.2	38.7/28.3	36.5/36.3	34.8/43.1	35.2/48.4
1	65.7/0	64.2/2.8	63.1/5.5	61.4/8.3	59.0/12.4	47.1/16.5	41.8/20.7	38.1/28.9	35.5/37.2	34.8/44.1	34.9/49.6

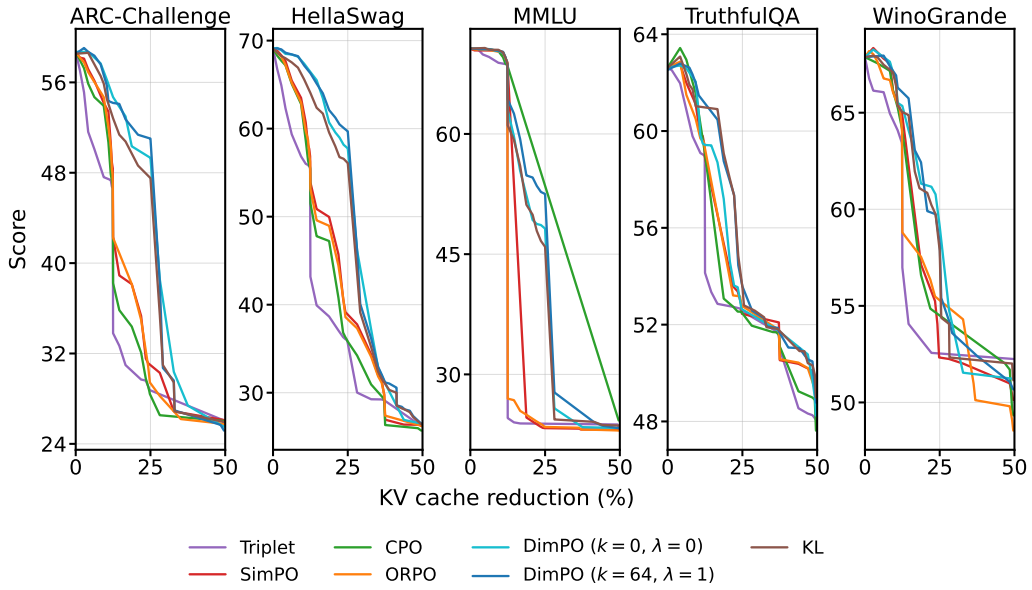


Figure 6: Harness task scores for different projection methods on Qwen3-4B-Instruct against the percentage of the KV cache removed.

Table 14: Average of ARC-Challenge, HellaSwag, MMLU, TruthfulQA and WinoGrande on Llama3.2-3B-Instruct. Each cell is the score and the percentage of the KV cache removed, for DimPO (k=64, λ=1).

d' / l	0	2	4	7	10	12	14	16	18	21	24	26	28
64	58.8/0	58.8/1.8	58.7/3.6	58.5/6.2	58.0/8.9	57.9/10.7	56.2/12.5	54.2/14.3	52.1/16.1	47.4/18.8	45.3/21.4	43.3/23.2	40.6/25
16	58.8/0	58.1/3.1	57.8/6.2	57.1/10.9	53.8/15.6	52.1/18.8	44.4/21.9	41.2/25	37.6/28.1	35.7/32.8	34.5/37.5	35.2/40.6	34.8/43.8
4	58.8/0	57.9/3.5	55.6/6.9	52.0/12.1	43.7/17.3	39.9/20.8	36.9/24.2	35.8/27.7	34.9/31.1	35.0/36.3	34.5/41.5	34.5/45	35.4/48.4
1	58.8/0	57.6/3.5	54.5/7.1	50.2/12.4	40.9/17.7	36.9/21.3	35.9/24.8	34.8/28.3	34.8/31.9	34.5/37.2	33.9/42.5	34.2/46.1	35.0/49.6

Table 15: Average of ARC-Challenge, HellaSwag, MMLU, TruthfulQA and WinoGrande on Qwen2.5-7B-Instruct. Each cell is the score and the percentage of the KV cache removed, for DimPO (k=64, λ=1).

d' / l	0	2	4	7	10	12	14	16	18	21	24	26	28
64	68.7/0	68.4/1.8	68.7/3.6	68.2/6.2	65.2/8.9	64.4/10.7	63.4/12.5	62.6/14.3	61.5/16.1	60.8/18.8	60.2/21.4	57.9/23.2	56.7/25
16	68.7/0	68.0/3.1	67.6/6.2	65.7/10.9	53.1/15.6	50.3/18.8	47.3/21.9	44.6/25	43.0/28.1	40.8/32.8	39.8/37.5	37.8/40.6	34.9/43.8
4	68.7/0	67.9/3.5	66.4/6.9	64.2/12.1	49.9/17.3	46.7/20.8	43.5/24.2	41.3/27.7	40.3/31.1	37.1/36.3	37.2/41.5	35.0/45	34.4/48.4
1	68.7/0	67.7/3.5	66.2/7.1	64.0/12.4	49.5/17.7	46.5/21.3	42.8/24.8	40.4/28.3	39.6/31.9	36.1/37.2	36.5/42.5	34.6/46.1	33.9/49.6

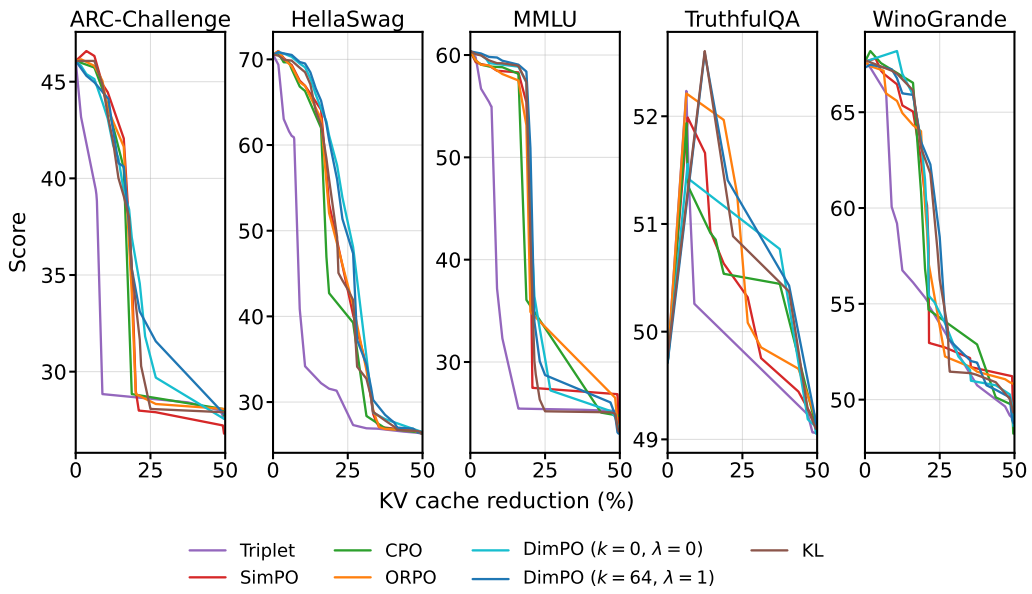


Figure 7: Harness task scores for different projection methods on Llama3.2-3B-Instruct against the percentage of the KV cache removed.

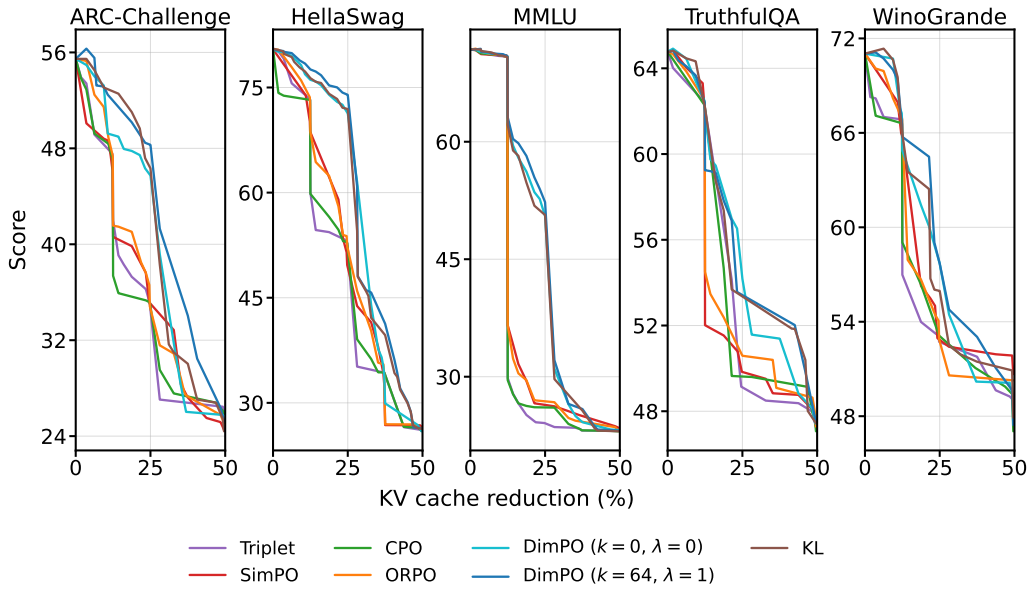


Figure 8: Harness task scores for different projection methods on Qwen2.5-7B-Instruct against the percentage of the KV cache removed.

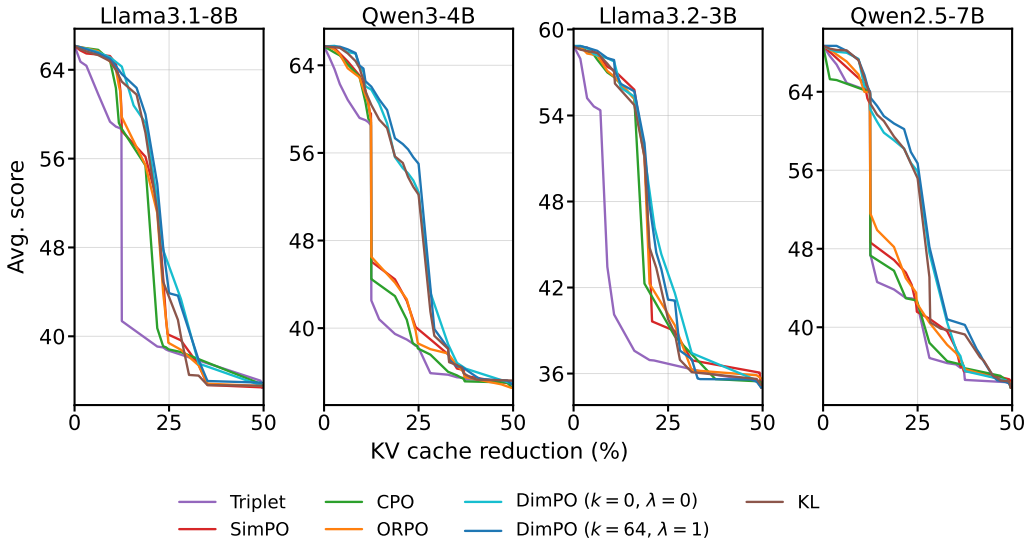


Figure 9: Average of ARC-Challenge, HellaSwag, MMLU, TruthfulQA, and WinoGrande against the percentage of the KV cache removed.

G RULER 4K SUBTASKS

Tables 16–18 report RULER 4k scores broken down by individual subtasks after projecting approximately 60% of the attention layers into a lower-dimensional space with d'=64 on Llama3.1-8B-Instruct, Llama3.2-3B-Instruct, and Qwen3-4B-Instruct. For each model, we compare DimPO (k=0, λ=0), DimPO (k=64, λ=1), and KL. Figure 10 shows a comparison of the different projection methods on RULER across different values of l for Llama3.2-3B-Instruct.

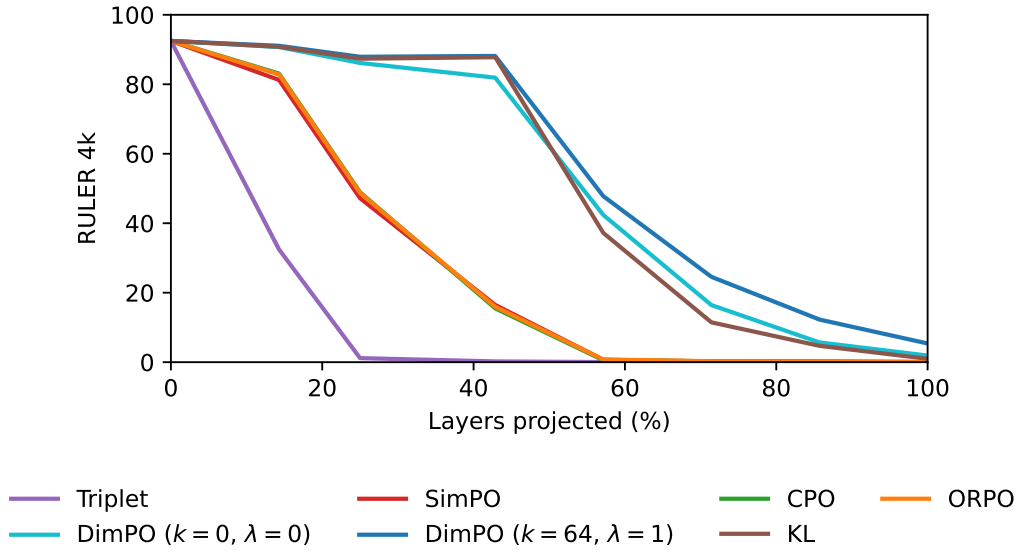


Figure 10: RULER 4k score against the percentage of projected layers for different projection objectives on Llama3.2-3B-Instruct.

Table 16: RULER 4k subtask scores on Llama3.1-8B-Instruct after projecting l=20 (62.5%) attention layers to d'=64.

Subtask	**Base**	**DimPO** (k=0, λ=0)	**DimPO** (k=64, λ=1)	**KL**
NIAH single 1	100.0	55.8	**87.6**	70.4
NIAH single 2	99.8	24.4	**83.4**	76.4
NIAH single 3	100.0	0.4	**8.2**	0.2
NIAH multi-key 1	100.0	37.8	**67.8**	64.8
NIAH multi-key 2	99.8	44.8	**66.6**	38.4
NIAH multi-key 3	100.0	0.2	**0.8**	0.0
NIAH multi-query	100.0	25.8	**47.4**	26.9
NIAH multi-value	100.0	27.5	**48.8**	36.0
Variable tracking	99.8	28.8	**39.6**	26.7
Common-word extraction	99.6	**63.1**	61.2	39.0
Frequent-word extraction	92.3	56.7	**71.5**	51.5
SQuAD	81.0	60.1	**60.3**	43.6
HotpotQA	63.2	37.2	**38.4**	32.0

Table 17: RULER 4k subtask scores on Llama3.2-3B-Instruct after projecting l=16 (57.1%) attention layers to d'=64.

Subtask	**Base**	**DimPO** (k=0, λ=0)	**DimPO** (k=64, λ=1)	**KL**
NIAH single 1	100.0	75.4	**79.8**	70.6
NIAH single 2	100.0	**80.4**	59.0	75.2
NIAH single 3	100.0	0.6	**3.4**	0.0
NIAH multi-key 1	100.0	63.8	**66.8**	62.2
NIAH multi-key 2	100.0	36.8	37.0	**45.8**
NIAH multi-key 3	100.0	**0.0**	**0.0**	**0.0**
NIAH multi-query	99.9	52.1	**66.3**	37.7
NIAH multi-value	100.0	46.2	**60.5**	34.8
Variable tracking	87.6	30.3	**43.5**	22.4
Common-word extraction	96.3	33.6	**60.8**	30.0
Frequent-word extraction	92.7	63.5	**68.5**	44.1
SQuAD	72.7	41.5	**44.2**	32.9
HotpotQA	53.0	26.8	**31.4**	28.0

Table 18: RULER 4k subtask scores on Qwen3-4B-Instruct after projecting l=24 (66.7%) attention layers to d'=64.

Subtask	**Base**	**DimPO** (k=0, λ=0)	**DimPO** (k=64, λ=1)	**KL**
NIAH single 1	100.0	98.4	**99.6**	98.6
NIAH single 2	100.0	98.8	97.4	**99.0**
NIAH single 3	99.8	**29.0**	23.6	28.0
NIAH multi-key 1	100.0	**86.4**	85.2	68.8
NIAH multi-key 2	100.0	**86.4**	74.6	50.4
NIAH multi-key 3	100.0	**25.6**	24.6	1.8
NIAH multi-query	100.0	**89.5**	87.7	67.9
NIAH multi-value	100.0	87.0	**91.2**	82.2
Variable tracking	100.0	20.2	**69.6**	11.2
Common-word extraction	99.3	**61.3**	56.5	31.3
Frequent-word extraction	81.2	57.4	**64.5**	53.1
SQuAD	76.7	**47.3**	46.2	32.7
HotpotQA	62.8	**36.4**	**36.4**	28.0

H INFERENCE LATENCY OF DIMPO AND SNAPKV

The projection shortens the keys while leaving the values at their original dimension. We therefore measure whether this reduction translates into lower inference latency when generating long sequences despite the linear projection overhead. We measure a single `generate` call with an 8k input and 2048 generated tokens on `BOOKSUM`, using batch size 4, greedy decoding, and disabled EOS termination. The base model takes 116.9s, while DimPO at l=20 (62.5% of the layers) takes 105.7s. SnapKV (Li et al., 2024) takes 54.0s with a budget of 1024 tokens and 63.9s with a budget of 2048 tokens. Adding the projection reduces these times to 51.1s and 59.0s, respectively.

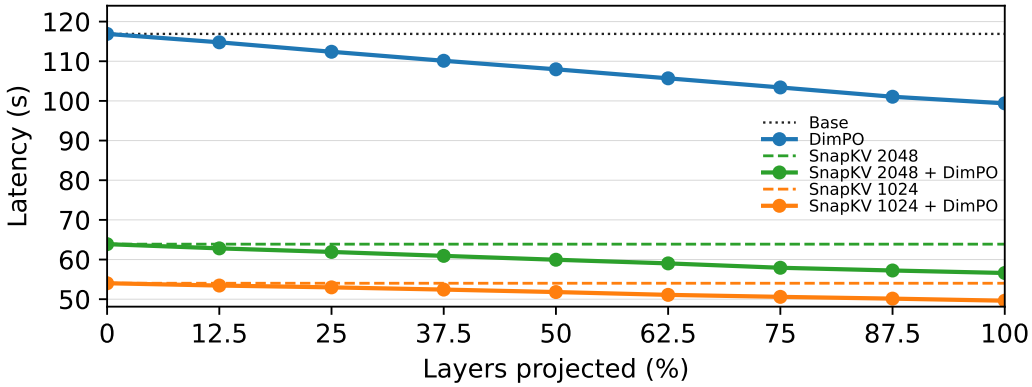


Figure 11: Inference latency with an 8k input and 2048 generated tokens. Dashed lines show base model and SnapKV alone, while marked lines add the projection to the last l layers. The x-axis shows the percentage of layers receiving the projection.

I MAGICPIG

We also explore the interaction between the listwise projection and MagicPIG, which optimizes the KV cache and the attention computation for long-context tasks using LSH-based random projections and a CPU-GPU co-design (Chen et al., 2025). Table 19 compares Llama3.1-8B-Instruct, its MagicPIG variant (K=8, L=75), and the same variant with DimPO (k=0, λ=0) applied to the last 4, 8, and 12 layers at d'=64. This run does not use the head term from the main experiments. Performance is evaluated on LongBench (Bai et al., 2024) and RULER (Hsieh et al., 2024), averaging scores over all available subtasks with harness 0.4.9.1 (Sutawika et al., 2025). We observe that on the last 4 layers the drop is minimal. With the projection on more layers, the degradation becomes more pronounced on longer context lengths, although even at 12 layers the projection still behaves well on the shorter contexts.

Table 19: Llama3.1-8B-Instruct, MagicPIG (K=8, L=75), and MagicPIG with DimPO (k=0, λ=0) at d'=64 on the last ℓ layers.

	LongBench	RULER				
		4K	8K	16K	32K	65K
Llama3.1-8B-Instruct	37.83	95.05	93.94	93.39	87.74	84.75
MagicPIG	35.84	92.63	92.35	91.64	86.71	83.67
MagicPIG, ℓ=4	34.96	91.60	90.27	87.90	83.74	81.42
MagicPIG, ℓ=8	33.38	89.95	87.12	83.83	81.83	72.08
MagicPIG, ℓ=12	32.58	87.59	83.41	79.56	75.29	63.66