X-EcoMLA: Upcycling Pre-Trained Attention into MLA for Efficient and Extreme KV Compression

Guihong Li[∗]**, Mehdi Rezagholizadeh**[∗]**, Mingyu Yang**[∗]**, Vikram Appia, Emad Barsoum**
Advanced Micro Devices, Inc. (AMD)
{guihong.li,mehdi.rezagholizadeh,mingyu.yang}@amd.com

Abstract

Multi-head latent attention (MLA) is designed to optimize KV cache memory through low-rank key-value joint compression. Rather than caching keys and values separately, MLA stores their compressed latent representations, reducing memory overhead while maintaining the performance. While MLA improves memory efficiency without compromising language model accuracy, its major limitation lies in its integration during the pre-training phase, requiring models to be trained from scratch. This raises a key question: *can we use MLA's benefits fully or partially in models that have already been pre-trained with different attention mechanisms?* In this paper, we propose X-EcoMLA to deploy post training distillation to enable the upcycling of Transformer-based attention into an efficient hybrid MLA variant through lightweight post-training adaptation, bypassing the need for extensive pre-training. We demonstrate that leveraging the dark knowledge of a well-trained model can enhance training accuracy and enable extreme KV cache compression in MLA without compromising model performance. The experimental results show that our proposed method can effectively compress the KV cache while preserving the performance on the benchmarks; specifically, for Llama3.2-1B-Instruct baseline, a 6.4× compression achieves the same average score by using only 3.6B training tokens and 70 GPU hours on AMD MI300, whereas a 10.6× compression have less than 0.1% average score drop with 7B training tokens and 140 GPU hours. The code for this work is available at https://github.com/AMD-AIG-AIMA/AMD-Hybrid-Models.

1 Introduction

Large language models (LLMs) have become ubiquitous, revolutionizing both academic research and industrial applications Brown et al. (2020); OpenAI (2023); Chowdhery et al. (2022); Liu et al. (2024a); Guo et al. (2025). Their success largely stems from pre-training and instruction tuning on vast amounts of data, the power of self-attention in Transformer architectures, and the computational capabilities of GPU accelerators.

Despite their widespread adoption, Transformers face two major challenges in their self-attention mechanism: quadratic computational complexity and high memory demands for KV cache storage, particularly when processing long sequences. Significant efforts have been made to mitigate these challenges by: (a) replacing self-attention with new sub-quadratic architectures such as state-space models (SSMs) Gu & Dao; Dao & Gu (2024); Poli et al. (2024); (b) enhancing the efficiency of existing Transformer self-attention mechanisms Arora et al. (2024); Ainslie et al. (2023); Zhang et al. (2024b); Yang et al. (2024); Qin et al. (2024); and (c) developing hybrid solutions that combine the advantages of both quadratic and sub-quadratic models Lieber et al. (2024); Dong et al. (2024); Wang et al. (2025); Bick et al. (2024). Among these approaches, the second category is particularly appealing, as it requires minimal architectural changes while capitalizing on existing hardware optimized for Transformers. This work focuses on improving the efficiency of existing self-attention mechanisms within this category.

Multi-head attention (MHA) is a fundamental mechanism in Transformer architectures. However, during inference, MHA requires the saving of large amount of key-value (KV) cache, resulting in high

---

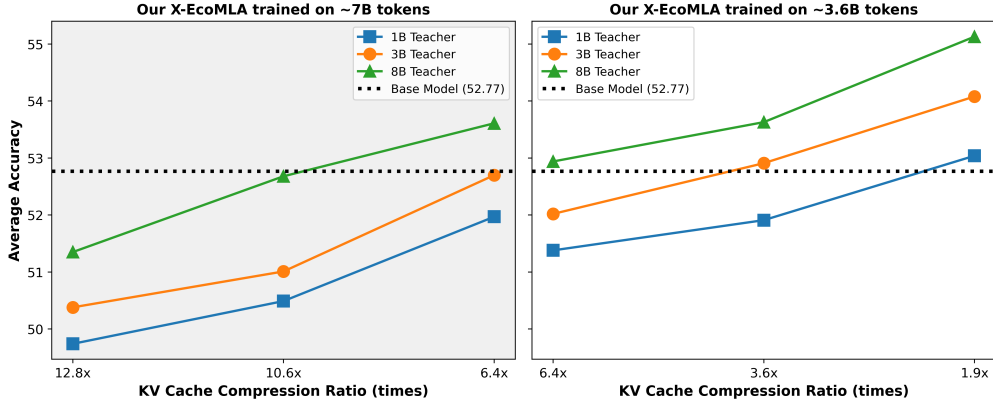[∗]Equal Contribution First Authors, with order determined alphabetically.

Figure 1: X-EcoMLA with Different Teacher Sizes: [Right] Our results show that using Llama3.2-1B, 3B, and 8B teacher models enables KV cache compression of Llama3.2-1B by 1.9×, 3.6×, and 6.4× using 3.6B tokens respectively, without compromising average accuracy across multiple tasks on the LM Harness Evaluation benchmark. [Left] With 7B training tokens, we can further compress the KV cache to 10.6× and 12.8x, while maintaining competitive accuracy.

memory consumption. To address this challenge, DeepSeek (Liu et al., 2024a;b) recently proposed multi-head latent attention (MLA), a novel approach that compresses the KV cache while maintaining the performance of LLMs. The mainstream training paradigm for MLA has relied on pre-training from scratch using vast amounts of data and computational resources. For example, pre-training the Deepseek-v3 model required 2.664M GPU hours on Nvidia H800 clusters. This highlights a major challenge: developing models with a new attention mechanism demands significant computational resources during pre-training. Given the substantial effort already invested in training Transformer models, a natural question arises—can we transfer the rich pre-training knowledge from trained LLMs into more efficient MLA models without training them from scratch?

Existing approaches showcase strong evidence supporting the feasibility of knowledge transfer with architectural adaptations for LLMs, such as MambaInLlama (Wang et al., 2025), MOHAWK (Bick et al., 2024), and HedgeHog (Zhang et al., 2024b). Inspired by these solutions, we introduce X-EcoMLA, a cost-effective knowledge transfer approach designed to upcycle pre-trained multi-head attention into MLA. In X-EcoMLA, we initialize MLA from its corresponding pre-trained attention blocks using our static or dynamic SVD approach, followed by distillation from a well-trained teacher model. By leveraging the dark knowledge of a high-quality model, we enhance training accuracy and achieve extreme KV cache compression in MLA without sacrificing performance. Our results demonstrate that an 8B teacher model enables 6.4× KV cache compression of the Llama3.2-1B-Inst baseline while preserving 100% of its average score across multiple tasks on the LM Harness Evaluation benchmark. This requires only 3.4B training tokens and about only 70 GPU hours on AMD MI300, while pre-training the Llama3.2-1B model requires 370K GPU hours[1]. Furthermore, we achieve a 10.6× compression with 7B training tokens and around 140 GPU hours, with less than 0.1% average score drop. We summarize our major contributions as follows:

- We propose a lightweight post-training approach to upcycle pre-trained attention to MLA, significantly reducing computational costs by eliminating the need for training from scratch.

- We develop static and dynamic SVD-based initialization techniques to improve the convergence and accuracy of MLA adaptation.

- We demonstrate that leveraging a larger teacher model enables extreme KV cache compression while maintaining model performance, achieving up to 10.6× compression with minimal accuracy loss. (see Fig. 1).

- We validate the effectiveness of our approach through extensive experiments on the LM Harness Evaluation benchmark, showcasing its efficiency across various settings and LLMs.

---

[1] https://huggingface.co/meta-Llama/Llama-3.2-1B

2 Related Work

This section gives a brief overview of related work; a detailed version is available in Appendix A.1.

KV Cache Management in Transformers   Transformers store key-value (KV) vectors for each token and each layer during auto-regressive generation, leading to high memory requirements during inference, especially for long sequences. Several methods have been proposed to reduce the KV cache size in Transformers, broadly divided into training-based and post-training approaches Shi et al. (2024). Post-training techniques are easier to apply but may degrade performance due to information loss. These include KV eviction (e.g., Heavy Hitter Zhang et al. (2023)), sliding window attention Arora et al. (2024); Beltagy et al. (2020), low-rank projection approach for KV-cache compression Chang et al. (2025), and quantization Kang et al. (2024); Zhang et al. (2024a). Some strategies, like Attention Sink Xiao et al. (2023) and KV merging Wang et al. (2024), aim to mitigate information loss while keeping memory usage low. This paper, however, focuses on training-based solutions, which tend to offer a better trade-off between efficiency and accuracy.

Training-based KV Cache Management   Training-based methods modify attention mechanisms to reduce memory use during inference. Multi-query attention (MQA)Shazeer (2019) and grouped-query attention (GQA)Ainslie et al. (2023) reduce KV cache size by sharing keys/values among heads. YOCO Sun et al. (2024) reduces redundancy with a shared KV cache across layers. DeepSeek-V2 Liu et al. (2024a) introduces multi-head latent attention (MLA), which compresses hidden states via low-rank projection, reducing cache size while outperforming standard MHA. Inspired by MLA's efficiency, this work explores adapting MLA to pre-trained model and we try to address this question: Can we upcycle pre-trained models to their MLA counterparts without costly retraining?

Upcycling Attention   Upcycling refers to upgrading pre-trained models with minimal computation Komatsuzaki et al. (2022). In attention upcycling, existing attention blocks are adapted into efficient forms like MLA without full retraining. GQA Ainslie et al. (2023) and Hedgehog Zhang et al. (2024b) achieve this via light fine-tuning or distillation. Hybrid models like MambaInLlama Wang et al. (2025) and MOHAWK Bick et al. (2024) distill knowledge from Transformer attention into Mamba layers. MHA2MLA Ji et al. (2025) introduces a data-efficient fine-tuning approach for converting MHA to MLA via partial RoPE removal and joint SVD-based low-rank approximation. In contrast, X-EcoMLA adopts a unified RoPE design with a shared Key-RoPE vector across all heads (similar to DeepSeek MLA), and employs structured initialization along with knowledge-distillation-based efficient training to enable effective MLA-based upcycling.

3 Background

In this section, we formalize the mathematical framework of MHA and MLA, following the notation from the original DeepSeek-V2 technical report Liu et al. (2024a) with some slight modifications.

3.1 Multi-Head Attention (MHA)

MHA projects the input hidden state H into three distinct spaces using learned weight matrices:

$$Q = HW^Q, \quad K = HW^K, \quad V = HW^V, \tag{1}$$

where $H \in \mathbb{R}^{l \times d}$ is the input sequence representation with l being sequence length and d internal hidden state dimension, and $W^Q, W^K, W^V \in \mathbb{R}^{d \times n_h d_h}$ are the learnable projection matrices where n_h is the number of attention heads, and d_h is the head dimension. The attention scores and final outputs are computed as:

$$A = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right), \quad O = AVW^O \tag{2}$$

where $W^O \in \mathbb{R}^{d \times d}$ is the output transformation matrix. During inference, MHA requires caching K and V for all past tokens, leading to a storage requirement of $2n_h d_h l$.

3.2 Multi-Head Latent Attention (MLA)

MLA introduces a low-rank joint compression strategy for keys and values, reducing the KV cache size. Instead of caching K and V, MLA compresses them into a lower-dimensional latent representa-

tion C^{KV}:

$$C^{KV} = HW^{DKV}, \tag{3}$$

where $W^{DKV} \in \mathbb{R}^{d \times r_{kv}}$ is the down-projection matrix, and $r_{kv} \ll d_h n_h$ is the compressed dimension for keys and values. The keys and values are then reconstructed from C^{KV} using:

$$K^C = C^{KV}W^{UK}, \quad V^C = C^{KV}W^{UV}, \tag{4}$$

where $W^{UK} \in \mathbb{R}^{r_{kv} \times n_h d_{qk}}$ and $W^{UV} \in \mathbb{R}^{r_{kv} \times n_h d_h}$ are the up-projection matrix for keys and values. Notice that in this paper we consider a more flexible setting where the queries and keys could have different dimensionality d_{qk} other than d_h. During inference, the learned matrices can be absorbed into the existing projection layers: W^{UK} and W^{UV} can be merged into W^Q and W^O respectively.

In the meantime, such low-rank compression can be also applied to the queries to reduce the memory usage while training:

$$C^Q = HW^{DQ}, \quad Q^C = C^Q W^{UQ}, \tag{5}$$

where $C^Q \in \mathbb{R}^{l \times r_q}$ represents the compressed latent vector for queries, r_q denotes and query compression dimension, $W^{DQ} \in \mathbb{R}^{d \times r_q}$ is the down-projection matrix and $W^{UQ} \in \mathbb{R}^{r_q \times n_h d_{qk}}$ denotes the up-projection matrix.

However, such low-rank KV compression is not compatible with Rotary Position Embedding (RoPE) as it breaks the matrix absorbing mechanism. As a result, the authors in Liu et al. (2024a) propose decoupled Rotary Position Embeddning where additional multi-head queries Q^R and a shared key K^R are applied to carry RoPE, which can be expressed as:

$$Q^R = \text{RoPE}(C^Q W^{QR}), \quad K^R = \text{RoPE}(HW^{KR}), \tag{6}$$

where $W^{QR} \in \mathbb{R}^{r_q \times n_h d_r}$ and $W^{KR} \in \mathbb{R}^{r_q \times d_r}$ represent the matrices to produce the decoupled keys and queries. Then, the RoPE embeddings Q^R, K^R and Non-RoPE (NoPE) embeddings Q^C, K^C are concatenated to perform the attention operation:

$$Q = [Q^C; Q^R], \quad K = [K^C; \text{repeat}(K^R)], \tag{7}$$

where repeat(.) denotes duplicating K^R for each head. After concatenation, the same attention operation is applied as in Equation 2. During inference, MLA requires caching only C^{KV} and K^R, reducing the storage requirement to $(r_{kv} + d_r)l$ which is significantly smaller than the standard MHA cache size.

4 Methodology of X-EcoMLA

We begin with a pre-trained Transformer model referred to as the "base model". Our methodology in this paper concerns upcycling the attention modules in the base model into MLA modules to save the KV cache memory while remaining minimum training efforts and performance degradation. To achieve that, we first propose our SVD-based weight initialization approach to better inherit the knowledge from the pre-trained model. Additionally, our initialization approach offers both static and dynamic rank selection. After initialization, we adopt the knowledge distillation training process in MambaInLlama Wang et al. (2025), which includes: end-to-end knowledge distillation, and direct preference optimization (DPO).

4.1 SVD-based Weight Initialization

As introduced in Section 3.2, the MLA module comprises several key parameters: W^{DQ}, W^{UQ}, W^{QR}, W^{DKV}, W^{KR}, W^{UK}, and W^{UV}. Correct initialization of these matrices is crucial to ensure a smooth transition and to preserve as much of the original model's knowledge as possible. For other parameters such as the output transformation matrix W^O and the feedforward module, we directly copy them from the pre-trained base model as initialization.

Although MLA and MHA are fundamentally different, MLA closely approximates a low-rank version of MHA if we disregard the positional embedding and intermediate layernorms. Based on this observation, we propose an SVD-based initialization method that initializes the MLA weights using

SVD-decomposed low-rank matrices derived from the pre-trained attention weights. Our method is simple and can be illustrated within few lines of pseudocode as shown in Algorithm 1. As demonstrated in our experiments, such straightforward initialization could significantly enhance the knowledge distillation performance compared to random initialization.

For simplicity, we assume the pre-trained attention block employs MHA with weight matrices $W^Q, W^K, W^V \in \mathbb{R}^{d \times d_h n_h}$. However, our method can be readily extended to MQA Shazeer (2019) and GQA Ainslie et al. (2023) by duplicating the key and value weight matrices to align with the total number of attention heads. Our approach supports varying query and key dimensions under the constraint $d_{qk} = d_r \leq d_h$. For the value dimension, we assume that the MLA and MHA modules share the same dimensionality d_h.

To initialize W^{DQ}, W^{UQ} and W^{QR}, we begin by performing SVD on the pretrained weight matrix W^Q to obtain:

$$W^Q = U_q \Sigma_q V_q^T, \tag{8}$$

where $U_q \in \mathbb{R}^{d \times r_q}, \Sigma_q \in \mathbb{R}^{r_q \times r_q}$, and $V_q \in \mathbb{R}^{d_h n_h \times r_q}$. For the down-projection matrix W^{DQ}, we directly set $W^{DQ} = U_q$. For the up-projection matrices, we first compute $\Sigma_q V_q^T$ and reshape it to $\overline{W}^{UQR} \in \mathbb{R}^{r_q \times n_h \times d_h}$. Then we partition it along the last dimension to derive the two up-projection matrices: W^{UQ} for the first d_{qk} elements and W^{QR} for the last d_r. This can be expressed as:

$$W^{UQ} = \text{reshape}(\overline{W}^{UQR}[:,:,: d_{qk}]), \quad W^{QR} = \text{reshape}(\overline{W}^{UQR}[:,:,-d_r :]), \tag{9}$$

where $\text{reshape}(.)$ denotes the reshape function that merges the last two dimensions of the input tensor.

For the remaining MLA weight matrices associated with keys and values, we first perform a joint SVD on the concatenated W^K and W^V:

$$[W^K, W^V] = U_{kv} \Sigma_{kv} V_{kv}^T, \tag{10}$$

where $U_{kv} \in \mathbb{R}^{d \times r_{kv}}, \Sigma_{kv} \in \mathbb{R}^{r_{kv} \times r_{kv}}$, and $V_{kv} \in \mathbb{R}^{2d_h n_h \times r_{kv}}$. For the down-projection matrix W^{DKV}, we directly set $W^{DKV} = U_{kv}$. To derive the up-projection matrix W^{UV}, we set it equal to the last $d_h n_h$ columns of $\Sigma_{kv} V_{kv}'$. For the up-projection matrix for keys W^{UK}, we first extract the first $d_h n_h$ columns of $\Sigma_{kv} V_{kv}'$ and reshape them into $\overline{W}^{UK} \in \mathbb{R}^{r_{kv} \times n_h \times d_h}$. Subsequently, we select the first d_{qk} elements along the last dimension of $\overline{W}^{UK}$ and reshape it back to obtain W^{UK}, which can be expressed as:

$$W^{UK} = \text{reshape}(\overline{W}^{UK}[:,:,: d_{qk}]). \tag{11}$$

Lastly, for the RoPE key embedding matrix W^{KR}, we employ a different initialization strategy as all attention heads share the same RoPE key embedding in MLA. We first compute the average key projection matrix $W^K_{avg} \in \mathbb{R}^{d \times d_h}$ across all attention heads. Then, we extract the last d_r columns from it to initialize W^{KR}, which can be expressed as:

$$W^{KR} = W^K_{avg}[:, -d_r :]. \tag{12}$$

Clearly, it is crucial to determine the rank values r_q and r_{kv} for the Q and KV matrices. We propose two methods for the rank selection: (i) **Fixed Rank Selection**, where constant rank values are used across all transformer blocks. (ii) **Dynamic Rank Selection**, which determines rank values based on two predefined energy thresholds $\delta_q, \delta_{kv} \in (0, 1]$. Taking W_Q as an example, we use SVD to decompose it and obtain the singular values:

$$\sigma_j, \quad j \in [\min(d, n_h d_h)], \quad \text{where } \sigma_j \geq \sigma_{j+1} \tag{13}$$

To determine the optimal rank, we define the cumulative energy based on the squared singular values. The rank r_{q_i} for the i-th transformer block is selected as follows:

$$r_{q_i} = \arg\min_R \left\{ \sum_{j=1}^{R} \sigma_j^2 \geq \delta_q E \right\}, \quad \text{where } E = \sum_{j=1}^{\min(d,\, n_h d_h)} \sigma_j^2 \tag{14}$$

This approach ensures that the selected rank captures at least δ_q fraction of the total energy E. The same energy-based dynamic rank selection process can be applied to the KV weight matrix as well.

4.2 Training Process

End-to-End Knowledge Distillation The primary training stage involves an end-to-end distillation using a Supervised Fine-Tuning (SFT) dataset introduced in Wang et al. (2025). During this stage, the goal is to minimize the KL divergence loss between the outputs of the student model (i.e., X-EcoMLA) and the teacher model, which can be expressed as:

$$\mathcal{L}_\theta = \sum_{t=1}^{T} \mathbf{KL}[p(\cdot|y_{1:t}, x, \theta_T)||p(\cdot|y_{1:t}, x, \theta)], \tag{15}$$

where θ is the trainable parameters of the student model and θ_T is the parameters of the teacher model, which is kept frozen while the distillation process. Such distillation step is crucial for transferring the rich, pre-trained knowledge from the teacher model. In Section A.5.1, we show that this distillation process is more effective than plain cross-entropy loss. Note that the teacher model can be any strong, pre-trained model—even one that is different from the base Transformer model that is used to construct the student model.

Direct Preference Optimization (DPO) In the final training stage, we perform DPO which is a binary cross entropy loss to adjust the preference of the student model. Following Wang et al. (2025), we set the distilled student model itself as the reference model as it makes the training much stabler and produces a sufficient performance gain, which can be observed from the experiments section.

5 Experiments

5.1 Experimental Setup

Model In this paper, we primarily focus on SmolLM-series models [2] (SmolLM-135M-Instruct, SmolLM-360M-Instruct, SmolLM-1.7B-Instruct) and Llama 3-series models [3] (Llama3.2-1B-Instruct, Llama3.2-3B-Instruct, Llama3.1-8B-Instruct) as our base models which span a variety of scales. All the models employ GQA as their attention module. For MLA, we utilize the same number of attention heads and the same head dimension for values. We tried multiple r_q, r_{kv}, and $d_{qk}(d_r)$ while maintaining a similar amount of parameters as the base model.

As mentioned earlier, we employ knowledge distillation, transferring knowledge from teacher models to the base model. In our experiments, we explore the impact of different teacher model sizes, including Llama3.2-1B-Instruct, Llama3.2-3B-Instruct, and Llama3.1-8B-Instruct Grattafiori et al. (2024). This approach allows us to analyze how the varying sizes of the teacher model impact the performance and efficiency of the distilled student model.

Training procedure Our approach begins with a pre-trained model, so we focus primarily on post-training rather than pre-training the base model again. We use a two-stage training procedure. Specifically, in the first stage (knowledge distillation), our X-EcoMLA model is trained with SFT using the KL loss between the output of the base model and the teacher model. We use AdamW optimizer with $\beta = (0.9, 0.98)$ and set the training batch size at 96. We use the same dataset as used in the paper Wang et al. (2025). It is constructed by aggregating and reformatting data from multiple publicly available sources, including *OpenHermes-2.5* Teknium (2023), *GenQA* Chen et al. (2024), and *Infinity-Instruct of Artificial Intelligence* (BAAI). For the second stage (DPO), based on finetuned models, we use the combination of three datasets for DPO preference tuning: *Llama3-ultrafeedback* Wang (2024), *orca_dpo_pairs* Lian et al. (2023), and *ultrafeedback_binarized* Cui et al. (2023). We set the training batch size at 64 and still use AdamW as the optimizer. We unfreeze all parameters in our X-EcoMLA model and train them for one epoch for both SFT and DPO.

5.2 Results

Similar to MambaInLlama Wang et al. (2025), we adopt the LM Harness Eval benchmark Gao et al. (2023) (branch big-refactor) to perform zero-shot evaluation on 9 different tasks: ARC-Challenge (ARC) Clark et al. (2018), ARC-Easy (ARE) Clark et al. (2018),HellaSwag (HS) Zellers et al. (2019), MMLU (MM) Hendrycks et al. (2020), OpenBookQA (OBQA) Mihaylov et al. (2018), PIQA Bisk

---

[2]https://huggingface.co/collections/HuggingFaceTB/smollm-6695016cad7167254ce15966
[3]https://huggingface.co/collections/meta-Llama/Llama-32-66f448ffc8c32f949b04c8cf

Model and Setting	Init. Method	KV-Size	ARC	ARE	HS	MMLU	OBQA	PIQA	PBMD	RA	WG	Avg.
SmolLM135M-Inst	Base Model	100%	27.39	43.64	41.91	24.11	33.80	67.30	55.80	32.06	51.22	**41.91**
↑X-EcoMLA	Random ($r_{kv} = 160$)	50%	21.08	30.68	25.27	22.95	25.00	53.54	55.20	22.87	49.57	34.03
↑X-EcoMLA + DPO	Random ($r_{kv} = 160$)	50%	21.59	30.56	25.17	22.95	26.20	53.97	55.20	22.97	49.33	34.22
↑X-EcoMLA	Fixed ($r_{kv} = 160$)	50%	27.05	43.31	40.10	24.51	31.80	66.97	55.40	31.39	51.30	41.31
↑X-EcoMLA + DPO	Fixed ($r_{kv} = 160$)	50%	28.24	45.33	40.26	26.20	33.60	65.89	55.60	32.63	50.36	**42.01**
↑X-EcoMLA	Dynamic ($\delta_{kv} = 0.85$)	49.5%	26.88	43.52	39.84	24.50	31.80	67.08	55.00	31.87	51.62	41.35
↑X-EcoMLA + DPO	Dynamic ($\delta_{kv} = 0.85$)	49.5%	27.99	45.29	40.46	25.84	33.40	65.51	55.20	32.73	50.28	41.86
SmolLM360M-Inst	Base Model	100%	32.76	56.10	52.74	24.37	37.00	70.51	55.40	33.59	54.06	**46.28**
↑X-EcoMLA	Random ($r_{kv} = 288$)	50%	22.10	36.36	26.72	23.22	25.00	55.60	55.20	23.25	48.38	35.09
↑X-EcoMLA + DPO	Random ($r_{kv} = 288$)	50%	22.61	35.61	27.12	23.19	26.60	56.26	54.80	23.64	49.57	35.49
↑X-EcoMLA	Fixed ($r_{kv} = 288$)	50%	33.11	54.55	50.77	23.26	36.60	71.06	55.40	31.96	54.14	45.65
↑X-EcoMLA + DPO	Fixed ($r_{kv} = 288$)	50%	34.39	55.77	51.93	25.68	38.60	69.37	55.20	33.88	53.35	46.46
↑X-EcoMLA	Dynamic ($\delta_{kv} = 0.88$)	49.3%	32.85	54.38	50.96	23.47	37.40	71.11	55.40	31.67	54.62	45.76
↑X-EcoMLA + DPO	Dynamic ($\delta_{kv} = 0.88$)	49.3%	34.64	54.97	51.75	25.45	38.80	69.59	55.20	33.88	54.06	**46.48**
Llama3.2-1B-Inst	Base Model	100%	37.97	63.51	60.77	46.09	35.00	74.37	60.20	38.09	59.67	**52.85**
↑X-EcoMLA	Random ($r_{kv} = 512$)	53.1%	35.32	60.48	54.03	27.77	35.20	71.98	55.80	33.88	55.01	47.72
↑X-EcoMLA + DPO	Random ($r_{kv} = 512$)	53.1%	38.99	62.71	56.20	28.04	36.80	73.39	56.40	36.27	56.20	49.44
↑X-EcoMLA	Fixed ($r_{kv} = 512$)	53.1%	36.95	63.89	58.88	43.40	36.00	74.16	58.20	37.32	60.30	52.12
↑X-EcoMLA + DPO	Fixed ($r_{kv} = 512$)	53.1%	39.93	63.89	60.73	42.39	37.80	74.92	58.80	40.77	60.54	**53.31**
↑X-EcoMLA	Dynamic ($\delta_{kv} = 0.95$)	54.7%	37.12	63.64	58.87	43.26	34.40	73.72	60.00	37.51	60.22	52.08
↑X-EcoMLA + DPO	Dynamic ($\delta_{kv} = 0.95$)	54.7%	41.21	64.86	60.96	42.86	37.60	74.43	58.60	39.23	58.33	53.12
Llama3.2-3B-Inst	Base Model	100%	45.90	67.76	70.36	60.46	36.20	75.57	69.60	40.77	67.17	**59.31**
↑X-EcoMLA	Random ($r_{kv} = 816$)	43%	41.64	67.34	65.11	46.97	36.40	75.24	61.6	39.04	63.69	55.23
↑X-EcoMLA + DPO	Random ($r_{kv} = 816$)	43%	44.80	69.78	67.01	47.37	38.80	76.01	62.60	40.00	64.80	56.80
↑X-EcoMLA	Fixed ($r_{kv} = 816$)	43%	43.09	67.76	69.54	56.96	37.00	75.84	66.00	41.34	67.48	58.33
↑X-EcoMLA + DPO	Fixed ($r_{kv} = 816$)	43%	48.21	70.45	72.24	57.42	38.40	76.55	66.80	46.22	68.59	**60.54**
↑X-EcoMLA	Dynamic ($\delta_{kv} = 0.95$)	43%	42.75	66.41	69.59	57.47	36.80	75.46	67.60	42.30	68.03	58.49
↑X-EcoMLA + DPO	Dynamic ($\delta_{kv} = 0.95$)	43%	48.46	69.99	72.26	57.73	39.40	75.79	68.40	46.32	65.90	60.47

Table 1: Zero-shot evaluation of self-distilled X-EcoMLA with different initialization methods (random, SVD with fixed/dynamic rank selection) and base models on the LM Harness Eval benchmark across nine tasks: ARC-Challenge (ARC), ARC-Easy (ARE), HellaSwag (HS), MMLU, OpenBookQA (OBQA), PIQA, PubMedQA (PBMD), RACE (RA), and WinoGrande (WG). (↑ denotes upcycling the "Base Model".)

et al. (2020), PubMedQA (PBMD) Jin et al. (2019), and RACE (RA) Lai et al. (2017), WinoGrande (WG) Sakaguchi et al. (2021).

Self-distillation Evaluation Table 1 shows the benchmark performance of our proposed X-EcoMLA when we use the base models themselves as the teacher model, which we refer to as self-distillation. We evaluate three different initialization settings: (i) Fixed rank selection with random initialization, (ii) Fixed rank selection with SVD initialization, and (iii) Dynamic rank selection with SVD initialization. For all experiments, we set $d_{qk} = d_r = 32$ and adjust r_q, r_{kv}, δ_q, and δ_{kv} accordingly to achieve around 50% KV cache compression while ensuring the total number of parameters after the MLA upcycling remain roughly the same. For each scenario, we evaluate training with the full dataset (6.8B tokens for SFT and 0.2B tokens for DPO) and it is observed that DPO significant boosts the distillation performance. For fixed rank selection schemes, it is evident that SVD initialization significantly enhances distillation performance compared to random initialization, yielding 22.8% and 30.91% improvements for SmolLM models and 8.1% and 6.5% improvements for Llama 3.2 models. Such observation demonstrates the necessity of applying our SVD initialization method to inherit the knowledge from the pre-trained targe models. For dynamic rank selection schemes, we observe similar performance as the fix rank selection in most experiments, which demonstrates its effectiveness. Note that our X-EcoMLA + DPO could always achieve better performance than the pre-trained base models with only $43\% - 54.7\%$ KV cache sizes.

Extreme KV Cache Compression with Larger Teacher Table 2 presents the impact of KV Cache compression on model accuracy across various benchmarks. For more details, please refer to Appendix A.4.4 and Table 11. We adopt Llama3.2-1B-Instruct as the base model and progressively reduce the KV cache size of our X-EcoMLA from 53.1% to 7.81%. With the same base model as our teacher, we observe a consistent accuracy drop across most evaluation tasks, which demonstrates the trade-off between reducing memory consumption and maintaining model accuracy.

However, our results reveal that such performance degradation from extreme KV cache compression can be mitigate if we utilize larger teacher models such as Llama3.2-3B-Instruct and Llama3.1-8B-Instruct. For instance, when reducing the KV cache to 15.6%, using Llama3.1-8B-Inst as the teacher recovers 1.56 of the average score (52.94 vs. 51.38) when trained with half of the dataset

Model and Setting	Teacher	Param	Tokens	ARC	ARE	HS	MMLU	OBQA	PIQA	PBMD	RA	WG	Avg.
Llama3.2-1B-Inst	-	1.24B	-	37.97	63.51	60.77	46.09	35.00	74.37	60.20	38.09	59.67	52.85
100% MLA Layers ($r_{kv} = 512$, $r_q = 864$, $d_{qk} = 32$) - KV Size: **53.1%**													
↑X-EcoMLA + DPO	Llama3.2-1B-Inst	1.23B	3.6B	39.93	63.51	60.52	41.58	37.20	73.99	59.80	40.48	60.38	53.04
↑X-EcoMLA + DPO	Llama3.2-3B-Inst	1.23B	3.6B	42.75	64.81	62.04	43.88	37.40	73.72	59.20	41.44	61.48	54.08
↑X-EcoMLA + DPO	Llama3.1-8B-Inst	1.23B	3.6B	44.03	68.86	63.49	43.81	37.40	73.94	61.40	41.82	61.40	**55.13**
100% MLA Layers ($r_{kv} = 256$, $r_q = 1184$, $d_{qk} = 32$) - KV Size: **28.1%**													
↑X-EcoMLA + DPO	Llama3.2-1B-Inst	1.23B	3.6B	40.02	63.26	58.74	39.79	36.40	72.80	55.60	40.19	60.38	51.91
↑X-EcoMLA + DPO	Llama3.2-3B-Inst	1.23B	3.6B	40.70	64.35	60.10	41.77	37.20	73.83	57.80	39.23	61.17	52.91
↑X-EcoMLA + DPO	Llama3.1-8B-Inst	1.23B	3.6B	41.98	66.46	61.33	41.78	37.20	74.27	59.00	40.00	60.69	**53.63**
100% MLA Layers ($r_{kv} = 128$, $r_q = 1344$, $d_{qk} = 32$) - KV Size: **15.6%**													
↑X-EcoMLA + DPO	Llama3.2-1B-Inst	1.23B	3.6B	39.16	61.83	57.27	37.85	36.20	73.45	56.40	40.19	60.06	51.38
↑X-EcoMLA + DPO	Llama3.2-3B-Inst	1.23B	3.6B	39.42	62.88	58.41	39.45	37.20	73.39	58.00	39.71	59.75	52.02
↑X-EcoMLA + DPO	Llama3.1-8B-Inst	1.23B	3.6B	41.30	65.61	59.64	39.47	37.60	74.27	59.20	39.52	59.83	**52.94**
↑X-EcoMLA + DPO	Llama3.2-1B-Inst	1.23B	7B	40.10	62.88	58.17	39.70	37.80	73.50	56.60	39.33	59.67	51.97
↑X-EcoMLA + DPO	Llama3.2-3B-Inst	1.23B	7B	39.33	64.86	58.92	41.86	37.40	73.83	58.80	39.71	59.59	52.70
↑X-EcoMLA + DPO	Llama3.1-8B-Inst	1.23B	7B	42.49	67.13	60.58	42.51	36.60	73.99	59.40	40.38	59.43	**53.61**
100% MLA Layers ($r_{kv} = 64$, $r_q = 1424$, $d_{qk} = 32$) - KV Size: **9.4%**													
↑X-EcoMLA + DPO	Llama3.2-1B-Inst	1.23B	7B	39.16	62.63	56.04	34.90	36.40	72.85	56.40	37.70	58.33	50.49
↑X-EcoMLA + DPO	Llama3.2-3B-Inst	1.23B	7B	37.97	63.55	56.95	37.54	35.40	72.74	57.00	38.66	59.27	51.01
↑X-EcoMLA + DPO	Llama3.1-8B-Inst	1.23B	7B	39.85	67.13	58.45	38.51	37.40	73.83	58.00	39.81	59.27	**52.47**
100% MLA Layers ($r_{kv} = 48$, $r_q = 1440$, $d_{qk} = 32$) - KV Size: **7.81%**													
↑X-EcoMLA + DPO	Llama3.2-1B-Inst	1.23B	7B	38.48	61.66	55.32	30.62	35.20	72.36	56.60	37.99	59.43	49.74
↑X-EcoMLA + DPO	Llama3.2-3B-Inst	1.23B	7B	36.18	62.21	55.82	36.41	35.60	72.03	57.00	38.09	60.06	50.38
↑X-EcoMLA + DPO	Llama3.1-8B-Inst	1.23B	7B	37.71	65.32	57.32	36.27	36.80	72.96	58.20	38.76	58.80	**51.35**

Table 2: Impact of KV-cache compression and teacher model size on performance. Reducing the KV-cache size lowers accuracy, but larger teacher models help recover performance. DPO further improves alignment and accuracy. (↑ denotes upcycling the base model.)

(3.6B tokens). With the larger teacher, our X-EcoMLA achieves even better performance than the pre-trained base model with only 15.6% KV cache size and 3.6B training tokens. As we increase the training tokens to 7B, we could even push the KV cache compression ratio to 9.4% without significant performance degradation compared to the pre-trained base model (52.47 vs. 52.85). These results highlight the effectiveness of leveraging larger teachers and preference tuning to resolve the adverse effects of extreme KV cache compression while maintaining strong accuracy across multiple NLP benchmarks.

Scalability to Larger Models We have extended X-EcoMLA to two 8B-parameter models (Llama3-8B and Llama3.1-8B). The results in Table 3 demonstrate that even under aggressive KV cache compression, X-EcoMLA maintains performance very close to the full-scale baseline. Notably, at KV-size 10.94%, performance remains nearly on par with the full model.

Model	KV-size	Avg.	ARC	ARE	HS	MM	OBQA	PIQ	PM	RA	WG
Llama3-8B-Inst (Base)	100%	65.78	56.66	81.61	75.81	63.82	42.60	78.62	75.00	46.03	71.90
↑**X-EcoMLA (Ours; $r_{kv} = 256$)**	15.63%	65.16	54.69	81.02	75.69	59.20	44.40	77.91	74.80	48.04	70.72
Llama3.1-8B-Inst (Base)	100%	66.63	54.86	79.55	79.23	68.13	43.00	80.90	75.40	44.69	73.88
↑**X-EcoMLA (Ours; $r_{kv} = 160$)**	10.94%	65.85	57.17	80.35	77.57	60.13	43.00	79.16	76.20	47.85	71.19

Table 3: Performance of X-EcoMLA with 8B models under aggressive KV cache compression

System-Level Inference Metrics We evaluate system-level performance in terms of throughput (sequences/sec) and peak GPU memory usage (GB) for both the baseline model (Llama3.1-8B) and our proposed model, X-EcoMLA-8B ($r_{kv} = 128$, $10.67\times$ KV compression), across a range of batch sizes. All experiments are conducted on identical hardware (single AMD MI300 GPU) under consistent settings. As shown in Fig. 2, X-EcoMLA-8B achieves approximately $1.7\times$ to $2\times$ higher throughput than the baseline across all batch sizes. In terms of memory efficiency, X-EcoMLA-8B substantially reduces peak memory consumption. For instance, at batch size 128, Llama3.1-8B consumes 143 GB of memory and fails to run larger batches, whereas X-EcoMLA-8B requires only 28 GB—representing a $5\times$ reduction. These results demonstrate that X-EcoMLA-8B not only maintains strong model accuracy but also delivers significant system-level improvements—achieving higher throughput and dramatically lower memory usage. This makes it well-suited for latency-sensitive and memory-constrained deployment scenarios.
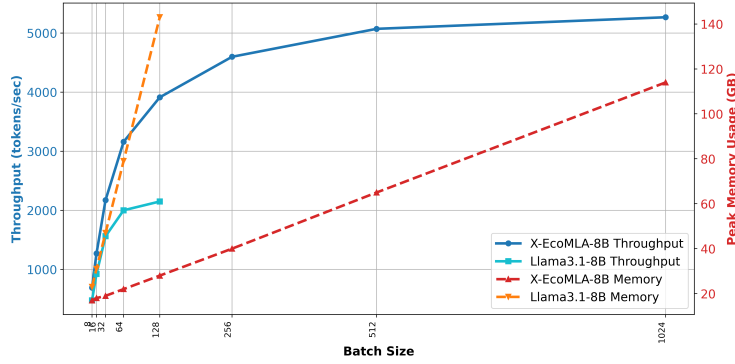
Figure 2: System-level inference performance for Llama3.1-8B and X-EcoMLA-8B on 8×AMD MI300 GPUs. X-EcoMLA enables higher throughput and drastically reduced memory usage across batch sizes. Llama3.1-8B runs out-of-memory (OOM) beyond batch size 128, while X-EcoMLA scales smoothly up to batch size 1024.

Comparison with Existing Post-Training Low-Rank Methods   We conducted experiments to compare our solution with two of the most recent SOTA methods — MHA2MLA Ji et al. (2025) and PALU Chang et al. (2025) — and to clarify how X-EcoMLA improves upon these approaches.

1. Comparison with MHA2MLA Baseline Setup   To more comprehensively evaluate the effectiveness of X-EcoMLA, we compare it against MHA2MLA Ji et al. (2025) under both continual pretraining and supervised fine-tuning (SFT) settings using the SmolLM 1.7B model. As SmolLM is originally built with MHA-based attention, this evaluation also demonstrates that X-EcoMLA can be seamlessly integrated into existing MHA-based architectures.

In the continual pretraining setting, we evaluate X-EcoMLA on the SmolLM 1.7B *base* model using a 12.5% KV cache size, following the same 6B-token training budget as the released MHA2MLA checkpoint[4]. We compare four configurations: (1) the full-attention baseline (100% KV cache), (2) MHA2MLA continually pretrained, (3) X-EcoMLA pretrained without a teacher, and (4) X-EcoMLA with self-distillation. Table 4 (top) summarizes the results where X-EcoMLA without a teacher achieves an average score of 51.94, slightly outperforming MHA2MLA (51.69) by +0.25. When using self-distillation, the accuracy improves to 52.87, reducing the gap to the full-attention baseline.

In the supervised fine-tuning setting, we use identical SFT data and compare performance under two KV compression ratios: 12.5% and 50%. We use SmolLM 1.7B-Instruct as both student and teacher for X-EcoMLA, and follow the joint-SVD + S_{higher} variant for MHA2MLA. As shown in Table 4 (bottom), X-EcoMLA outperforms MHA2MLA across both compression levels. At 12.5% KV size, X-EcoMLA achieves 49.34 vs. 48.19 (+1.15), and at 50%, it reaches 50.15 vs. 49.79 (+0.36).

We attribute this performance gain to a key architectural difference: the RoPE (Rotary Position Embedding) design. MHA2MLA stores separate Key-RoPE vectors per head. Under a fixed budget (e.g., 32 dimensions per token), each head receives only $\frac{32}{n_h}$ RoPE dimensions. In contrast, X-EcoMLA adopts the unified RoPE approach used in DeepSeek MLA, where all heads share a single Key-RoPE vector. This allows each head to fully leverage all 32 dimensions, offering 8× more positional encoding capacity in an 8-head setup—crucial for preserving performance under aggressive compression.

2. Comparison with PALU   PALU Chang et al. (2025) represents a recent state-of-the-art approach for low-rank KV-cache compression. It decomposes the linear projection layers into low-rank matrices, caches the compressed intermediate states, and reconstructs the full keys and values on the fly during inference. To evaluate the effectiveness of X-EcoMLA, we compare X-EcoMLA against PALU using the Llama3-8B-Instruct model.

Following the experimental protocol described in Section 4.2 of the PALU paper, we evaluate zero-shot performance on the used subset of the LM Harness Eval benchmark using PALU's best-

---

[4]https://huggingface.co/fnlp/SmolLM-1B7-MLA-d_kv_8

Method	KV Size	Avg. Acc.	ARC	ARE	HS	MM	OBQA	PIQA	PM	RA	WG
SmolLM 1.7B (Base)	100%	54.67	46.42	73.48	65.74	27.73	42.00	76.06	62.60	37.03	60.93
MHA2MLA Ji et al. (2025)	12.5%	51.69	41.55	69.57	61.43	24.63	39.00	74.70	60.20	35.69	58.41
↑X-EcoMLA-pretrain ($r_{kv} = 480$)	12.5%	51.94	40.19	69.36	62.52	23.79	40.00	75.30	61.40	35.89	59.04
↑X-EcoMLA ($r_{kv} = 480$)	12.5%	52.87	42.41	72.14	62.84	25.55	41.40	75.19	61.40	36.27	58.64
SmolLM 1.7B-Ins (Base)	100%	50.49	37.71	62.96	60.81	25.86	39.80	73.50	60.80	36.17	56.83
MHA2MLA (SFT)	12.5%	48.19	35.07	60.94	56.18	23.36	36.40	72.36	57.20	34.83	57.38
↑X-EcoMLA (SFT, $r_{kv} = 480$)	12.5%	49.34	37.29	62.96	59.05	23.68	38.00	72.52	59.80	34.64	56.12
MHA2MLA (SFT)	50%	49.79	37.63	62.29	59.60	24.09	38.20	74.05	60.40	34.35	57.46
↑X-EcoMLA (SFT, $r_{kv} = 2016$)	50%	50.15	37.97	64.06	60.23	24.89	39.00	73.34	60.20	34.93	56.75

Table 4: Comparison between X-EcoMLA and MHA2MLA a under both continual pretraining (top) and supervised fine-tuning (bottom) settings on the SmolLM 1.7B model.

performing configurations: G-LRD and J-LRD, both operating at a 50% KV compression ratio. Reported PALU results are taken directly from their paper. The results, summarized in Table 5, show that X-EcoMLA achieves an average accuracy of 67.34 at only 15.63% KV cache size. This performance is nearly on par with the full model (67.87) and clearly outperforms both PALU-J-LRD (66.19) and PALU-G-LRD (64.45), with respective gains of +1.15 and +2.89 points—despite using approximately $3\times$ less KV storage.

Model	KV-Size	Avg Acc.	ARC	ARE	HS	OBQA	PIQA	WG
Llama3-8B-Inst (Base)	100%	67.87	56.66	81.61	75.81	42.60	78.62	71.90
PALU-J-LRD	50%	66.19	51.96	79.63	73.20	43.40	76.50	72.45
PALU-G-LRD	50%	64.45	48.99	76.30	70.36	42.60	76.06	72.38
↑X-EcoMLA (Ours)	15.63%	67.34	54.69	81.02	75.69	44.40	77.53	70.72

Table 5: Comparison between X-EcoMLA and PALU baselines on Llama3-8B. Despite operating at a significantly lower KV size (15.63%), X-EcoMLA matches or outperforms 50%-KV SOTA methods.

Expanded Discussion and Experimental Details   The Appendix provides additional context and supporting results that complement the main paper. In Section A.1, we present a more detailed discussion of related work on KV cache compression, training-based memory-efficient architectures, and upcycling techniques. Section A.2 outlines the pseudocode for our proposed SVD-based initialization strategy for MLA layers. Hyperparameter for both fixed and dynamic rank selection are reported in Section A.3, helping to guide practical configurations. Section A.4 includes supplementary evaluations: long-context benchmarks on LongBench (Section A.4.1); comparisons with H2O as an alternative KV cache compression method (Section A.4.2); hybrid MLA variants that combine attention and MLA layers (Section A.4.3); and detailed results from our extreme KV cache compression experiments (Section A.4.4). In Section A.5, we present ablation studies, analyzing the impact of distillation loss versus direct supervision (Section A.5.1), the role of LayerNorm (Section A.5.2), and the trade-off between larger teacher models and longer training (Section A.5.3). Collectively, these sections provide deeper insights into the design choices and empirical robustness of X-EcoMLA across diverse settings and compression budgets.

6   Conclusion

In this work, we introduced X-EcoMLA, a lightweight post-training adaptation approach that enables the upcycling of pre-trained Transformer attention into an efficient MLA or hybrid variant. By leveraging dark knowledge from a well-trained teacher model and employing SVD-based initialization, our method significantly reduces KV cache memory requirements without sacrificing model performance. Our results show that using an 8B teacher model allows us to compress the KV cache size of the Llama3.2-1B-Instruct baseline by 6.4× while preserving 100% of its average score across multiple tasks on the LM Harness Evaluation benchmark. This is achieved with only 3.6B training tokens and about 70 GPU hours on AMD MI300 GPUs. Alternatively, we can compress KV cache by 10.6× using 7B training tokens over approximately 140 GPU hours, while maintaining 99.8% of the average score. These findings highlight the potential of X-EcoMLA as a practical and scalable solution for integrating MLA into existing LLMs, paving the way for more memory-efficient and deployable models without the need for costly retraining from scratch.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. *arXiv preprint arXiv:2402.18668*, 2024.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Aviv Bick, Kevin Y Li, Eric P Xing, J Zico Kolter, and Albert Gu. Transformers to ssms: Distilling quadratic knowledge to subquadratic models. *arXiv preprint arXiv:2408.10189*, 2024.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S Abdelfattah, and Kai-Chiang Wu. Palu: Kv-cache compression with low-rank projection. In *The Thirteenth International Conference on Learning Representations*, 2025.

Jiuhai Chen, Rifaa Qadri, Yuxin Wen, Neel Jain, John Kirchenbauer, Tianyi Zhou, and Tom Goldstein. Genqa: Generating millions of instructions from a handful of prompts. *arXiv preprint arXiv:2406.10323*, 2024.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022.

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

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with high-quality feedback, 2023.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Xin Dong, Yonggan Fu, Shizhe Diao, Wonmin Byeon, Zijia Chen, Ameya Sunil Mahabaleshwarkar, Shih-Yang Liu, Matthijs Van Keirsbilck, Min-Hung Chen, Yoshi Suhara, Yingyan Lin, Jan Kautz, and Pavlo Molchanov. Hymba: A hybrid-head architecture for small language models, 2024. URL https://arxiv.org/abs/2411.13676.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation. 2023.

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

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Ethan He, Abhinav Khattar, Ryan Prenger, Vijay Korthikanti, Zijie Yan, Tong Liu, Shiqing Fan, Ashwath Aithal, Mohammad Shoeybi, and Bryan Catanzaro. Upcycling large language models into mixture of experts. *arXiv preprint arXiv:2410.07524*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Tao Ji, Bin Guo, Yuanbin Wu, Qipeng Guo, Lixing Shen, Zhan Chen, Xipeng Qiu, Qi Zhang, and Tao Gui. Towards economical inference: Enabling deepseek's multi-head latent attention in any transformer-based llms. *arXiv preprint arXiv:2502.14837*, 2025.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. *arXiv preprint arXiv:1909.06146*, 2019.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. Gear: An efficient kv cache compression recipefor near-lossless generative inference of llm. *arXiv e-prints*, pp. arXiv–2403, 2024.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. Sparse upcycling: Training mixture-of-experts from dense checkpoints. *arXiv preprint arXiv:2212.05055*, 2022.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*, 2017.

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". Openorca: An open dataset of gpt augmented flan reasoning traces. https://https://huggingface.co/Open-Orca/OpenOrca, 2023.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avashalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A hybrid transformer-mamba language model, 2024.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024a.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024b.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

Beijing Academy of Artificial Intelligence (BAAI). Infinity instruct. https://huggingface.co/datasets/BAAI/Infinity-Instruct, 2024.

OpenAI. Gpt-4 technical report, 2023.

Michael Poli, Armin W Thomas, Eric Nguyen, Pragaash Ponnusamy, Björn Deiseroth, Kristian Kersting, Taiji Suzuki, Brian Hie, Stefano Ermon, Christopher Ré, et al. Mechanistic design and scaling of hybrid architectures. *arXiv preprint arXiv:2403.17844*, 2024.

Zhen Qin, Dong Li, Weigao Sun, Weixuan Sun, Xuyang Shen, Xiaodong Han, Yunshen Wei, Baohong Lv, Xiao Luo, Yu Qiao, and Yiran Zhong. Transnormerllm: A faster and better large language model with improved transnormer, 2024. URL https://arxiv.org/abs/2307.14995.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Luohe Shi, Hongyi Zhang, Yao Yao, Zuchao Li, and Hai Zhao. Keep the cost down: A review on methods to optimize llm's kv-cache consumption. *arXiv preprint arXiv:2407.18003*, 2024.

Yutao Sun, Li Dong, Yi Zhu, Shaohan Huang, Wenhui Wang, Shuming Ma, Quanlu Zhang, Jianyong Wang, and Furu Wei. You only cache once: Decoder-decoder architectures for language models. *Advances in Neural Information Processing Systems*, 37:7339–7361, 2024.

Teknium. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants, 2023. URL https://huggingface.co/datasets/teknium/OpenHermes-2.5.

Junxiong Wang. Llama3 ultrafeedback-armorm dataset. https://huggingface.co/datasets/JunxiongWang/llama3-ultrafeedback-armorm, 2024.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander M. Rush, and Tri Dao. The mamba in the llama: Distilling and accelerating hybrid models, 2025. URL https://arxiv.org/abs/2408.15237.

Zheng Wang, Boxiao Jin, Zhongzhi Yu, and Minjia Zhang. Model tells you where to merge: Adaptive kv cache merging for llms on long-context tasks. *arXiv preprint arXiv:2407.08454*, 2024.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. Pqcache: Product quantization-based kvcache for long context llm inference. *arXiv preprint arXiv:2407.12820*, 2024a.

Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Re. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=4g02l2N2Nx.

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

page

A Appendix

A.1 Related Work

KV Cache Management in Transformers Several approaches have been proposed to reduce or compress the KV cache size of Transformers, which can be broadly categorized into training-based and post-training solutions Shi et al. (2024). Training-based methods involve modifying the model architecture and pre-training it, typically yielding better performance, whereas post-training methods are easier to apply and do not require retraining.

A variety of post-training KV cache management solutions have been explored in the literature. One common strategy is KV cache eviction, such as Heavy Hitter (H_2O) Zhang et al. (2023), which defines an eviction policy based on the observation that only a few tokens contribute to the highest attention scores. This method retains the most recent and most significant tokens while discarding the others. Another approach is sliding window attention Arora et al. (2024); Beltagy et al. (2020), which restricts attention to a fixed number of recent tokens (or predefined patterns) to maintain a bounded KV cache size. Attention Sink Xiao et al. (2023) builds on this by retaining initial tokens in the KV cache to improve performance. Quantization-based KV cache compression Kang et al. (2024); Zhang et al. (2024a) reduces memory usage by storing KVs in a lower-precision format, while KV cache merging Wang et al. (2024) minimizes information loss by merging KV entries instead of discarding them. Although post-training solutions are computationally efficient, they often lead to performance degradation due to information loss. In contrast, training-based methods offer a better balance between memory efficiency and model accuracy. This paper focuses on training-based solutions, which we review in the following.

Training-based KV Cache Management Training-based solutions modify the attention mechanism or replace it with alternative architectures in Transformer models to reduce KV cache memory requirements. For instance, multi-query attention (MQA)Shazeer (2019) shares keys and values across all attention heads, reducing the KV cache size by a factor of n compared to a multi-head attention (MHA) model with n KV heads. However, sharing a single KV across n query heads can be too restrictive. To address this, grouped-query attention (GQA)Ainslie et al. (2023) divides query heads into groups, allowing each group to share a single set of key and value heads, making a balance between memory efficiency and performance. Another notable approach is YOCO Sun et al. (2024), a decode-decoder model that consists of a self-decoder and a cross-decoder module. Instead of storing KV vectors for each layer and token, the self-decoder module provides a shared global KV cache to the cross-decoder layers, significantly reducing memory overhead. Multi-head latent attention (MLA), introduced in DeepSeek-V2 Liu et al. (2024a), is another KV-cache efficient variation of MHA. MLA reduces KV cache size by projecting input hidden states into a compressed latent space through low-rank projection, leading to a substantial reduction in memory usage. DeepSeek-V2 demonstrated that MLA can outperform standard MHA while maintaining efficiency.

Motivated by MLA's strong performance and efficiency, we focus on adapting MLA for already pre-trained models. However, training-based solutions typically require full pre-training from scratch or extensive continual training. This raises a fundamental question: Can we upcycle pre-trained models to their MLA counterparts without costly retraining? In the following section, we review existing solutions for model upcycling that can be leveraged for this purpose.

Upcycling Attention In Komatsuzaki et al. (2022), model upcycling is defined as "upgrading an existing model with a relatively small additional computational budget." This term has primarily been used to describe the conversion of dense models into mixture-of-experts (MoE) models in an efficient manner Komatsuzaki et al. (2022); He et al. (2024). In this paper, we focus on the concept of attention upcycling, which involves adapting pre-trained attention blocks in a Transformer into more efficient forms, such as MLA, without requiring full re-training from scratch. There are several examples of attention upcycling in the literature. For instance, in GQA, Ainslie et al. (2023) propose replacing MHA blocks with GQA and performing light continual pre-training for adaptation. Similarly, HedgehogZhang et al. (2024b) introduces an upcycling method that converts pre-trained attention into linear attention using knowledge distillation.

A notable line of work focuses on leveraging the duality between Transformer self-attention and alternative architectures. MambaInLlama Wang et al. (2025) demonstrates this by replacing some attention

---

Algorithm 1 Python-like pseudocode of the proposed SVD initialization for MLA.

```
1  # MHA weights: W_Q, W_K, W_V
2  # MLA weights: W_DQ, W_UQ, W_QR, W_DKV, W_UK, W_KR, W_UV
3
4  # Initialization of W_DQ, W_UQ, and W_QR
5  U_q, sigma_q, V_q = svd(W_Q)
6  W_DQ = U_q
7  W_UQR_bar = (sigma_q @ V_q).view(r_q, n_h, d_h)
8  W_UQ = W_UQR_bar[:, :, :d_qk].view(r_q, n_h*d_qk)
9  W_QR = W_UQR_bar[:, :, -d_r:].view(r_q, n_h*d_r)
10
11 # Initialization of W_DKV, W_UK, W_KR, W_UV
12 U_kv, sigma_kv, V_kv = svd(torch.cat((W_K, W_V), -1))
13 W_DKV = U_kv
14 W_K_avg = W_K.view(d, n_h, d_h).mean(1)
15 W_KR = W_K_avg[:, -d_r:]
16
17 W_UKV = sigma_kv @ V_kv
18 W_UK_bar = W_UKV[:, :d_h*n_h].view(r_kv, n_h, d_h)
19 W_UK = W_UK_bar[:,:,:d_qk].view(r_kv, n_h*d_qk)
20 W_UV = W_UKV[:, d_h*n_h:]
```

---

layers in pre-trained models with Mamba layers, initializing them from their corresponding attention layers, and then fine-tuning using end-to-end knowledge distillation. Similarly, MOHAWK Bick et al. (2024) follows a knowledge distillation-based approach for training hybrid attention-Mamba models. However, MOHAWK differs from MambaInLlama in some aspects: (a) It does not initialize the student sub-quadratic model from the Transformer attention layers; (b) It incorporates intermediate layer distillation in addition to end-to-end distillation.

A.2 Algorithm

Our simple method for initializing the MLA weights using SVD approach applied to the pre-trained attention weights is summarized in the pseudocode in Algorithm 1.

A.3 Hyper-parameter Selection

In Table 6 and 7, we present the model performance with different hyperparameters for fixed rank selection and dynamic rank selection, respectively. In Table 6, we evaluate three KV ranks ($r_{kv} = 512, 256, 128$) and two head dimensions ($d_{qk} = 32, 64$). We adjust r_q accordingly to make sure all configurations have approximately the same number of parameters. The results indicate a significant performance loss as the KV rank r_{kv} decreases. With the same KV rank, $d_{qk} = 64$ generally provides better performance. However, such advantage is more obvious with $r_{kv} = 128, 256$ where r_q is relatively large. When $r_{kv} = 512$, both head dimensions provides similar performance. In Table 7, we explore two thresholds (90% and 95%) for r_q and r_{kv} and two head dimensions ($d_{qk} = 32, 64$) for dynamic rank selection. When training with a small portion of the dataset (1.6B), we notice that the performance is mainly influenced by the KV rank r_{kv}. Although setting $d_{qk} = 64$ leads to more parameters, it does not necessarily translate to performance improvement, even when trained with the full dataset.

A.4 Supplementary Results

A.4.1 Long Context Evaluations

We evaluated our MLA-optimized models on the LongBench benchmark, which covers a range of long-context understanding tasks such as LCC, Qasper, QMSum, Multi-News, and SamSum. Table 8 reports the results under various KV-cache size for both Llama3.2-1B and Llama3.2-3B models. Notably, X-EcoMLA 3B models achieves a score of 60.03 on LCC, significantly outperforming the full-sized Llama3.2-3B baseline (52.11), despite using only 43% of the KV cache. Across other tasks

Configuration	Param	r_q	r_{kv}	d_{qk}	KV Size	Tokens	Avg Score
Base Model: **Llama3.2-1B-Inst**							
↑X-EcoMLA +DPO	1.23B	864	512	32	53.1%	3.6B	53.04
↑X-EcoMLA +DPO	1.23B	480	512	64	56.3%	3.6B	53.14
↑X-EcoMLA +DPO	1.23B	1184	256	32	28.1%	3.6B	51.91
↑X-EcoMLA +DPO	1.23B	736	256	64	31.3%	3.6B	52.38
↑X-EcoMLA +DPO	1.23B	1344	128	32	15.6%	3.6B	51.38
↑X-EcoMLA +DPO	1.23B	864	128	64	18.8%	3.6B	51.60

Table 6: Hyperparameter selection for the internal dimensions of the X-EcoMLA block under a fixed setting with 100% MLA layers, without LayerNorm, and using an identical teacher model as the base.

Configuration	Param	r_q	r_{kv}	d_{qk}	KV Size	Tokens	Avg Score
Base Model: **Llama3.2-1B-Inst**							
↑X-EcoMLA +DPO	1.22B	90%	90%	32	42.7%	1.6B	51.26
↑X-EcoMLA +DPO	1.25B	90%	90%	64	45.9%	1.6B	51.31
↑X-EcoMLA +DPO	1.23B	95%	90%	32	42.7%	1.6B	51.36
↑X-EcoMLA +DPO	1.27B	95%	90%	64	45.9%	1.6B	51.21
↑X-EcoMLA +DPO	1.23B	90%	95%	32	54.7%	1.6B	52.18
↑X-EcoMLA +DPO	1.26B	90%	95%	64	57.9%	1.6B	51.51
↑X-EcoMLA +DPO	1.24B	95%	95%	32	54.7%	1.6B	52.40
↑X-EcoMLA +DPO	1.28B	95%	95%	64	57.9%	1.6B	52.16
↑X-EcoMLA +DPO	1.23B	90%	95%	32	54.7%	7.0B	53.22
↑X-EcoMLA +DPO	1.26B	90%	95%	64	57.9%	7.0B	53.23

Table 7: Hyperparameter selection for the internal dimensions of the X-EcoMLA block under a dynamic setting with 100% MLA layers, without LayerNorm, and using an identical teacher model (Llama-3.2-1B) as the base.

such as Qasper, Multi-News, and SamSum, our compressed models match or even slightly exceed the performance of their full-cache counterparts. These results indicate that our method scales well to long-sequence scenarios and is particularly effective in memory-constrained environments.

Model	KV-Size	Avg. Acc.	lcc	repobench-p	qasper	qmsum	multi_news	samsum
Llama3.2-1B-Inst (Base)	100.00%	30.805	35.47	40.12	22.92	21.65	25.68	38.99
↑X-EcoMLA (ours)	53.13%	30.77	38.73	40.36	21.13	20.50	25.76	38.11
↑X-EcoMLA (ours)	28.13%	30.66	38.74	40.54	21.21	20.61	25.62	37.26
Llama3.2-3B-Inst (Base)	100.00%	40.01	52.11	54.16	40.42	23.63	26.51	43.21
↑X-EcoMLA (ours)	42.91%	39.29	60.03	56.24	29.94	21.08	27.54	40.93
↑X-EcoMLA (ours)	25.00%	39.11	59.59	53.94	31.75	20.93	27.19	41.26

Table 8: Long-context evaluation on the LongBench benchmark across varying KV cache sizes. All the X-EcoMLA models are trained with Llama3.1-8B-Inst as the teacher model

A.4.2 *Comparison with other KV Cache Compression Techniques*

We compare X-EcoMLA with the widely used H2O method Zhang et al. (2023), using the same base model (Llama3.2-1B-Instruct) and identical KV cache sizes. The evaluation is conducted on the **lm-eval-hardness** benchmark to assess performance under increasingly aggressive memory constraints. As shown in Table 9, X-EcoMLA consistently outperforms H2O across all compression levels—both in terms of average accuracy and on most individual tasks. Notably, at a KV size of 9.4%, X-EcoMLA achieves an average accuracy of **50.49%**, compared to **45.05%** for H2O, with particularly large gains on ARC, ARE, and PIQA. Even at 6.25% KV size, X-EcoMLA maintains a strong lead, indicating its robustness under extreme compression. These results demonstrate that X-EcoMLA achieves significantly better accuracy under the same memory budget, making it a strong candidate for memory-efficient inference.

Model	KV-size	Avg. Acc.	ARC	ARE	HS	MM	OBQA	PIQ	PM	RA	WG
H2O	15.6%	50.30	37.71	57.41	59.91	40.83	31.60	71.11	60.40	37.99	55.80
↑X-EcoMLA (ours)	15.6%	51.97	40.10	62.88	58.17	39.70	37.80	73.50	56.60	39.33	59.67
H2O	9.4%	45.05	30.03	43.01	57.79	33.25	29.60	64.96	58.80	36.08	51.93
↑X-EcoMLA (ours)	9.4%	50.49	39.16	62.63	56.04	34.90	36.40	72.85	56.40	37.70	58.33
H2O	6.25%	41.30	26.54	34.68	52.75	26.95	28.60	59.03	58.60	34.26	50.28
↑X-EcoMLA (ours)	6.25%	49.74	38.48	61.66	55.32	30.62	35.20	72.36	56.60	37.99	59.43

Table 9: Comparison of X-EcoMLA and H2O Zhang et al. (2023) across various KV cache sizes. X-EcoMLA consistently outperforms H2O, especially under aggressive compression.

A.4.3 Hybrid MLA Models

In this section, we include some supplementary results. Table 10 shows the benchmark performance of our X-EcoMLA method on Llama3.2-1B-Inst model when we use the same model as teacher. We evaluate three different initialization settings: (i) Fixed rank selection with random initialization, (ii) Fixed rank selection with SVD initialization, and (iii) Dynamic rank selection with SVD initialization. For the fixed rank selection scenario, we set $r_q = 854$, $r_{kv} = 512$, and $d_{qk} = d_r = 32$ such that the total number of parameters after the MLA upcycling remain roughly the same. For the dynamic rank selection case, we apply a threshold of 0.95 for both r_q and r_{kv} so that the number of parameters aligns with other setups. We investigate two MLA layer upcycling strategies: upcycling 100% of layers to MLA and upcycling 50% of layers to MLA. For the 100% upcycling strategy, we replace all GQA modules in the base model with MLA. In this scenario, the proposed X-EcoMLA model uses only 53.1% of the KV cache size for fixed rank selection and 54.7% for dynamic rank selection. For the 50% upcycling strategy, we replace GQA modules in layers 1, 3, 5, 7, 8, 10, 12, and 14. This brings us 78.1% KV cache size for the fixed rank selection and 78% for dynamic rank selection.

For each strategy, we evaluate training with the full dataset (6.8B tokens) and half dataset (3.4B tokens). It is evident that for fixed rank selection schemes, SVD initialization significantly enhances distillation performance compared to random initialization, yielding an 8% improvement for 100% MLA and 3% improvement for 50% MLA.

Model and Setting	Init. Method	KV-Size	Tokens	ARC	ARE	HS	MMLU	OBQA	PIQA	PBMD	RA	WG	Avg.
Llama3.2-1B-Inst	Base	100%	-	37.97	63.30	60.65	46.05	34.80	74.32	60.00	38.18	59.67	**52.77**
100% MLA Layers- Teacher: Identical to the Base Model													
↑X-EcoMLA	Random (512)	53.1%	6.8B	35.32	60.48	54.03	27.77	35.20	71.98	55.80	33.88	55.01	47.72
↑X-EcoMLA + DPO	Random (512)	53.1%	7.0B	38.99	62.71	56.20	28.04	36.8	73.39	56.40	36.27	56.20	49.44
↑X-EcoMLA	Fixed (512)	53.1%	6.8B	36.95	63.89	58.88	**43.40**	36.00	74.16	58.20	37.32	60.30	52.12
↑X-EcoMLA + DPO	Fixed (512)	53.1%	7.0B	40.19	63.93	60.67	42.31	**37.60**	**75.03**	59.20	40.86	61.01	53.42
↑X-EcoMLA	Dynamic (95%)	54.7%	6.8B	37.12	63.64	58.87	42.54	34.40	73.72	60.00	37.51	60.22	52.08
↑X-EcoMLA + DPO	Dynamic (95%)	54.7%	7.0B	**40.36**	**64.31**	60.88	42.54	36.80	74.16	**61.40**	40.77	60.69	**53.54**
↑X-EcoMLA	Fixed (512)	53.1%	3.4B	37.37	**64.35**	58.36	42.03	35.00	73.61	57.40	37.03	59.51	51.63
↑X-EcoMLA + DPO	Fixed (512)	53.1%	3.6B	39.93	63.51	60.52	41.58	**37.20**	73.99	**59.80**	40.48	60.38	**53.04**
↑X-EcoMLA	Dynamic (95%)	54.7%	3.4B	37.12	63.64	58.44	**42.14**	34.40	73.61	57.00	37.22	59.98	51.50
↑X-EcoMLA + DPO	Dynamic (95%)	54.7%	3.6B	**40.27**	62.71	**60.55**	41.21	36.40	**74.16**	**59.80**	39.90	60.14	52.79
50% MLA Layers, Teacher: Identical to the Base Model													
↑X-EcoMLA	Random (512)	78.1%	6.8B	36.86	62.79	57.23	38.19	36.00	73.78	56.20	36.75	58.72	50.72
↑X-EcoMLA + DPO	Random (512)	78.1%	7.0B	38.99	63.64	59.00	37.46	**37.60**	74.59	57.00	39.14	60.46	51.99
↑X-EcoMLA	Fixed (512)	78.1%	6.8B	37.97	63.01	59.71	**44.37**	35.80	**74.86**	**60.60**	38.37	59.98	52.74
↑X-EcoMLA + DPO	Fixed (512)	78.1%	7.0B	40.87	63.93	**61.95**	43.39	37.20	74.48	59.80	40.48	60.85	53.66
↑X-EcoMLA	Dynamic (95%)	78%	6.8B	38.48	63.85	59.78	44.27	35.60	74.54	60.40	38.18	60.85	52.88
↑X-EcoMLA + DPO	Dynamic (95%)	78%	7.0B	**41.64**	**64.44**	61.78	43.58	36.40	74.21	60.00	40.48	60.93	**53.71**
↑X-EcoMLA	Fixed (512)	78.1%	3.4B	37.12	63.55	59.36	**44.16**	35.20	73.94	57.60	37.70	60.77	52.16
↑X-EcoMLA + DPO	Fixed (512)	78.1%	3.6B	40.61	**64.73**	**62.06**	43.51	**37.40**	73.78	59.40	40.29	**61.25**	**53.67**
↑X-EcoMLA	Dynamic (95%)	78%	3.4B	38.05	63.09	59.37	43.60	35.00	**74.27**	59.80	36.94	60.93	52.33
↑X-EcoMLA + DPO	Dynamic (95%)	78%	3.6B	39.93	63.64	61.76	43.33	36.40	73.83	**61.40**	40.57	60.30	53.46

Table 10: Zero-shot evaluation of MLA variants with different initialization methods (random, SVD with fixed rank selection, and SVD with dynamic rank selection) on the LM Harness Eval benchmark across nine tasks: ARC-Challenge (ARC), ARC-Easy (ARE), HellaSwag (HS), MMLU, OpenBookQA (OBQA), PIQA, PubMedQA (PBMD), RACE (RA), and WinoGrande (WG). (↑ denotes upcycling the base model.)

A.4.4 More Details on Extreme KV Cache Compression Experiments

In this section, we include more details for Table 2 results. For each row, we also show the results of SFT training.

Model and Setting	Teacher	Param	Tokens	ARC	ARE	HS	MMLU	OBQA	PIQA	PBMD	RA	WG	Avg.
Llama3.2-1B-Inst	-	1.24B	-	37.97	63.30	60.65	46.05	34.80	74.32	60.00	38.18	59.67	52.77
100% MLA Layers ($r_{kv} = 512$, $r_q = 864$, $d_{qk} = 32$) - KV Size: **53.1%**													
↑X-EcoMLA	Llama3.2-1B-Inst	1.23B	3.4B	37.37	64.35	58.36	42.03	35.00	73.61	57.40	37.03	59.51	51.63
↑X-EcoMLA + DPO	Llama3.2-1B-Inst	1.23B	3.6B	39.93	63.51	60.52	41.58	37.20	73.99	59.80	40.48	60.38	53.04
↑X-EcoMLA	Llama3.2-3B-Inst	1.23B	3.4B	37.71	65.19	58.84	43.13	36.20	73.45	58.20	37.89	59.67	52.25
↑X-EcoMLA + DPO	Llama3.2-3B-Inst	1.23B	3.6B	42.75	64.81	62.04	43.88	37.40	73.72	59.20	41.44	61.48	54.08
↑X-EcoMLA	Llama3.2-8B-Inst	1.23B	3.4B	39.51	67.38	60.41	43.18	38.40	73.94	60.40	38.28	61.72	53.69
↑X-EcoMLA + DPO	Llama3.2-8B-Inst	1.23B	3.6B	44.03	68.86	63.49	43.81	37.40	73.94	61.40	41.82	61.40	**55.13**
100% MLA Layers ($r_{kv} = 256$, $r_q = 1184$, $d_{qk} = 32$) - KV Size: **28.1%**													
↑X-EcoMLA	Llama3.2-1B-Inst	1.23B	3.4B	37.54	62.84	56.89	41.22	33.6	73.12	55.4	36.46	59.19	50.70
↑X-EcoMLA + DPO	Llama3.2-1B-Inst	1.23B	3.6B	40.02	63.26	58.74	39.79	36.40	72.80	55.60	40.19	60.38	51.91
↑X-EcoMLA	Llama3.2-3B-Inst	1.23B	3.4B	36.35	63.51	57.09	41.30	35.00	73.07	56.80	36.46	60.14	51.08
↑X-EcoMLA + DPO	Llama3.2-3B-Inst	1.23B	3.6B	40.70	64.35	60.10	41.77	37.20	73.83	57.80	39.23	61.17	52.91
↑X-EcoMLA	Llama3.2-8B-Inst	1.23B	3.4B	38.14	65.45	58.70	41.15	36.20	73.67	59.00	36.17	60.62	52.12
↑X-EcoMLA + DPO	Llama3.2-8B-Inst	1.23B	3.6B	41.98	66.46	61.33	41.78	37.20	74.27	59.00	40.00	60.69	**53.63**
100% MLA Layers ($r_{kv} = 128$, $r_q = 1344$, $d_{qk} = 32$) - KV Size: **15.6%**													
↑X-EcoMLA	Llama3.2-1B-Inst	1.23B	3.4B	36.52	61.41	55.37	38.02	34.60	72.52	56.00	35.60	58.56	49.84
↑X-EcoMLA + DPO	Llama3.2-1B-Inst	1.23B	3.6B	39.16	61.83	57.27	37.85	36.20	73.45	56.40	40.19	60.06	51.38
↑X-EcoMLA	Llama3.2-3B-Inst	1.23B	3.4B	36.26	61.95	55.84	39.28	35.40	71.76	57.60	35.89	59.27	50.36
↑X-EcoMLA + DPO	Llama3.2-3B-Inst	1.23B	3.6B	39.42	62.88	58.41	39.45	37.20	73.39	58.00	39.71	59.75	52.02
↑X-EcoMLA	Llama3.2-8B-Inst	1.23B	3.4B	36.35	64.60	57.32	38.25	37.00	73.45	60.40	35.22	58.25	51.20
↑X-EcoMLA + DPO	Llama3.2-8B-Inst	1.23B	3.6B	41.30	65.61	59.64	39.47	37.60	74.27	59.20	39.52	59.83	**52.94**
↑X-EcoMLA	Llama3.2-1B-Inst	1.23B	6.8B	37.54	62.21	56.36	39.67	35.40	73.23	55.60	35.31	58.33	50.41
↑X-EcoMLA + DPO	Llama3.2-1B-Inst	1.23B	7B	40.10	62.88	58.17	39.70	37.80	73.50	56.60	39.33	59.67	51.97
↑X-EcoMLA	Llama3.2-3B-Inst	1.23B	6.8B	35.58	63.51	56.71	41.38	35.80	72.80	57.20	35.89	58.56	50.83
↑X-EcoMLA + DPO	Llama3.2-3B-Inst	1.23B	7B	39.33	64.86	58.92	41.86	37.40	73.83	58.80	39.71	59.59	52.70
↑X-EcoMLA	Llama3.2-8B-Inst	1.23B	6.8B	38.65	66.88	58.46	42.01	34.80	73.67	60.00	36.46	59.12	52.23
↑X-EcoMLA + DPO	Llama3.2-8B-Inst	1.23B	7B	42.49	67.13	60.58	42.51	36.60	73.99	59.40	40.38	59.43	**53.61**
100% MLA Layers ($r_{kv} = 64$, $r_q = 1424$, $d_{qk} = 32$) - KV Size: **9.4%**													
↑X-EcoMLA	Llama3.2-1B-Inst	1.23B	6.8B	37.12	61.32	54.46	34.89	35.60	72.36	56.80	35.22	57.30	49.45
↑X-EcoMLA + DPO	Llama3.2-1B-Inst	1.23B	7B	39.16	62.63	56.04	34.90	36.40	72.85	56.40	37.70	58.33	50.49
↑X-EcoMLA	Llama3.2-3B-Inst	1.23B	6.8B	35.07	61.95	54.95	38.61	35.20	72.09	57.40	35.98	58.25	49.94
↑X-EcoMLA + DPO	Llama3.2-3B-Inst	1.23B	7B	37.97	63.55	56.95	37.54	35.40	72.74	57.00	38.66	59.27	51.01
↑X-EcoMLA	Llama3.2-8B-Inst	1.23B	6.8B	36.09	65.07	57.01	38.60	35.80	72.96	58.00	35.98	59.98	51.05
↑X-EcoMLA + DPO	Llama3.2-8B-Inst	1.23B	7B	40.02	67.17	58.40	38.53	37.80	73.83	58.00	39.43	60.93	**52.68**
100% MLA Layers ($r_{kv} = 48$, $r_q = 1440$, $d_{qk} = 32$) - KV Size: **6.25%**													
↑X-EcoMLA	Llama3.2-1B-Inst	1.23B	6.8B	36.77	60.61	53.51	32.44	33.40	72.20	56.60	34.55	58.33	48.71
↑X-EcoMLA + DPO	Llama3.2-1B-Inst	1.23B	7B	38.48	61.66	55.32	30.62	35.20	72.36	56.60	37.99	59.43	49.74
↑X-EcoMLA	Llama3.2-3B-Inst	1.23B	6.8B	33.70	61.32	54.11	35.96	34.60	71.27	56.00	35.22	58.48	48.96
↑X-EcoMLA + DPO	Llama3.2-3B-Inst	1.23B	7B	36.18	62.21	55.82	36.41	35.60	72.03	57.00	38.09	60.06	50.38
↑X-EcoMLA	Llama3.2-8B-Inst	1.23B	6.8B	36.35	64.60	55.50	36.65	34.60	72.31	57.80	35.79	58.25	50.21
↑X-EcoMLA + DPO	Llama3.2-8B-Inst	1.23B	7B	37.71	65.32	57.32	36.27	36.80	72.96	58.20	38.76	58.80	**51.35**

Table 11: Impact of KV-cache compression and teacher model size on performance. Reducing the KV-cache size lowers accuracy, but larger teacher models help recover performance. DPO further improves alignment and accuracy. (↑ denotes upcycling the base model.)

A.5 Ablation Studies

A.5.1 Distillation vs. Cross-Entropy

In Table 12, we examine the trade-off between learning from the teacher knowledge (via KL divergence) and direct supervision from the dataset (via cross-entropy loss w.r.t. ground truth) during the SFT distillation stage. We adopt Llama3.2-1B-Instruct for both our base model and student model, and we use CE and KL to denote the weights for the cross-entropy loss and KL divergence loss. The results show that relying solely on direct supervision (CE = 1, KL = 0) significantly degrades model performance (48.54 vs. 52.77), underscoring the importance of leveraging teacher knowledge for effective learning.

In contrast, incorporating teacher knowledge—either exclusively or in combination with direct supervision—yields the best results, indicating the importance of teacher-guided learning in maintaining accuracy. Given these insights, we primarily adopt teacher-based learning in our configurations to minimize hyperparameter tuning efforts unless stated otherwise.

Configuration	CE	KL	Avg Score
Llama3.2-1B-Inst	-	-	52.77
↑X-EcoMLA	0	1	50.84
↑X-EcoMLA	1	0.01	50.93
↑X-EcoMLA	1	0.05	50.71
↑X-EcoMLA	1	0.1	50.98
↑X-EcoMLA	1	0	48.54

Table 12: Comparison of different CE and KL loss weightings in the SFT knowledge distillation phase. The experiment utilizes the same teacher as the base model and applies dynamic SVD compression with $\delta_q = \delta_{kv} = 0.95$, trained on 20% of the dataset.

A.5.2 Impact of LayerNorm

The original MLA module incorporates additional LayerNorm layers between the down- and up-projection operations. However, we observe that it is beneficial to omit those LayerNorm layers in our proposed X-EcoMLA, as evidenced in Table 13 and the loss curves in Figure 3 in the Appendix. By removing the intermediate LayerNorm layers, our proposed X-EcoMLA demonstrates an improved loss convergence. Besides, across various setups (fixed, dynamic) and different training dataset sizes (3.4B, 6.8B) in Table 13, the removal of LayerNorm layers consistently leads to performance gains.
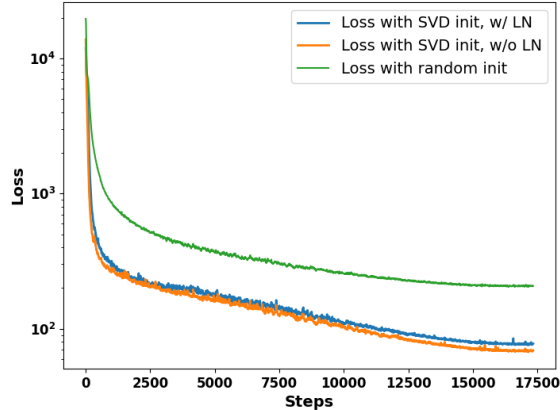

Figure 3: Loss curve comparison between random initialization and SVD initialization w/ and w/o LayerNorm layers. All schemes are trained with fixed KV rank selection and 100% MLA layers upcycling.

Figure 3 shows the loss curves of our SVD initialization with and without layer normalization, as well as with random initialization. The results demonstrate that removing layer normalization leads to lower loss values.

A.5.3 Larger Teacher or more Training Data?

Table 14 highlights the impact of increasing training data (tokens) versus using a larger teacher model on both accuracy score and training time. When using the same teacher model (Llama3.2-1B-Inst), increasing the number of training tokens (from 3.4B to 6.8B) improves performance but comes at the cost of significantly higher training time (from 4.82 to 9.64 hours).

On the other hand, switching to a larger teacher (e.g., Llama3.2-3B-Inst or Llama3.2-8B-Inst) provides notable accuracy improvements with less reliance on additional training data. For instance, using the

Configuration	Init. Method	LayerNorm	KV Size	Tokens	**Avg Score**
	Base Model: **Llama3.2-1B-Inst**				
↑X-EcoMLA	Dynamic ($\delta_{kv} = 0.95$)	✓	54.7%	6.8B	51.68
↑X-EcoMLA	Dynamic ($\delta_{kv} = 0.95$)	✗	54.7%	6.8B	**52.08**
↑X-EcoMLA	Fixed ($r_{kv} = 512$)	✓	53.1%	6.8B	51.68
↑X-EcoMLA	Fixed ($r_{kv} = 512$)	✗	53.1%	6.8B	**52.12**
↑X-EcoMLA	Dynamic ($\delta_{kv} = 0.95$)	✓	54.7%	3.4B	51.18
↑X-EcoMLA	Dynamic ($\delta_{kv} = 0.95$)	✗	54.7%	3.4B	**51.50**
↑X-EcoMLA	Fixed ($r_{kv} = 512$)	✓	53.1%	3.4B	50.89
↑X-EcoMLA	Fixed ($r_{kv} = 512$)	✗	53.1%	3.4B	**51.63**

Table 13: Comparison of MLA with LayerNorm vs. without LayerNorm

Model and Setting	Teacher	Tokens	Training time	Avg Score
	Base Model: **Llama3.2-1B-Inst**; 100% MLA Layers ($r_{kv} = 512, r_q = 864$)			
↑X-EcoMLA	Llama3.2-1B-Inst	6.8B	9.64 hours	52.12
↑X-EcoMLA + DPO	Llama3.2-1B-Inst	7.0B	10.06 hours	53.42
↑X-EcoMLA	Llama3.2-1B-Inst	3.4B	4.82 hours	51.63
↑X-EcoMLA + DPO	Llama3.2-1B-Inst	3.6B	5.24 hours	53.04
↑X-EcoMLA	Llama3.2-3B-Inst	3.4B	6.24 hours	52.25
↑X-EcoMLA + DPO	Llama3.2-3B-Inst	3.6B	6.65 hours	54.08
↑X-EcoMLA	Llama3.2-8B-Inst	3.4B	8.54 hours	53.69
↑X-EcoMLA + DPO	Llama3.2-8B-Inst	3.6B	8.96 hours	55.13

Table 14: Comparison of training efficiency and accuracy when increasing training data (tokens) versus using a larger teacher model. Larger teachers yield better accuracy with moderate time increases. The time cost is measured on a 8 MI300 GPUs.

8B teacher with DPO achieves the highest score, outperforming training with double tokens under the smaller 1B teacher (55.13 vs. 53.42), even with less training time (8.96 vs. 10.06 hours). However, this comes with a moderate increase in training time.

These results suggest that leveraging a stronger teacher model is generally more efficient for improving accuracy than simply increasing training data. While additional tokens help, the benefit diminishes compared to the gains from using a larger teacher, making training with a larger teacher a more effective strategy when computational resources allow.