NovaKV

Low-rank compression of KV cache

Charles De Clercq

Abstract

A language model stores, for every token it has read, the key and value projections of every layer. This cache grows linearly with context length and quickly dominates the memory cost of inference. NovaKV compresses it by a low-rank factorisation applied after training, on a frozen model. On Llama-3.1-8B-Instruct the cache occupies 44.5% less memory, measured as a real allocation rather than estimated from ranks, with perplexity moving from 7.22 to 7.57 and long-range retrieval losing 0.73 points. The same recipe transfers to Mistral-7B-Instruct-v0.3 without modification. This document states the approach and the measurements.

1 The problem

Attention requires, at every decoding step, the keys and values of every preceding token. These keys and values are stored in memory as the *KV cache*, whose size grows linearly with the context. At long context this cache becomes a major botleneck on how many requests a machine can serve.

Two recent methods compress it by low-rank approximation. STAR-KV [1] decomposes each attention head separately, choosing the rank of each by a trained differentiable soft threshold. ReCalKV [2] instead calibrates a basis shared across heads, in closed form, weighted by the Gram matrix of real activations.

2 Approach

The observation NovaKV is built on is that these two methods are the same operation at two different granularities: a low-rank approximation over a group of heads, the only free parameter being the size of the group. They are not competing proposals but two points on a single axis. Once seen this way, the question is how these two methods behave and in which context each is best-suited. We propose the following.

Keys. One basis per layer, shared across all key heads with its rank selected by a trained soft threshold following STAR-KV construction. The method is upgraded with an initialization from a whitened decomposition, that is, one computed in the inner product induced by the Gram matrix of real activations rather than in the raw weight space.

Values. A basis computed in closed form at a fixed rank, calibrated once again from real activations, and then frozen from the first step.

This asymetry between keys and values has contantly shown to give better results than any configuration which applies the same mechanism to both projections (a learned threshold on both, or a closed-form calibration on both).

Three decisive layers are left uncompressed to ensure better results. The compressed model is then physically truncated, so that the memory save is effectively measured in practice, rather than extrapolated from a mask over full-rank weights.

3 Results

All figures below are measured on the truncated checkpoint. The dense column is the unmodified model measured under the same protocol. We avoid here any complementary quantization trick, we only deal here with activations.

	Llama-3.1-8B-Instruct			Mistral-7B-Instruct-v0.3		
	dense	NovaKV	Δ	dense	NovaKV	Δ
Perplexity (WikiText-2)	7.22	7.57	+0.35	5.50	5.76	+0.26
RULER	97.60	96.87	−0.73	97.12	96.35	−0.77
LongBench	40.12	37.91	−2.21	43.44	39.72	−3.72
MMLU	63.23	59.00	−4.23	59.98	57.44	−2.54
HumanEval	69.51	60.37	−9.14	42.07	32.93	−9.14
MBPP	60.40	50.00	−10.40	42.60	35.20	−7.40
Cache (KB/token)	131.1	72.7	−44.5%	131.1	71.5	−45.4%

Perplexity is at context 2048; the rest are percentages. RULER measures retrieval of a specific fact from a long context, LongBench comprehension of whole documents, MMLU academic knowledge, HumanEval and MBPP code generation.

Language modelling and long-range retrieval are nearly untouched: the two models lose 0.73 and 0.77 points on RULER, and a few hundredths of perplexity. Knowledge and document comprehension cost between two and four points. Code generation is where compression is expensive, and it is the same on both models.

Cache memory. The last row is a measured allocation: the difference in occupied GPU memory before and after a real prefill, with the cache kept alive and everything else released. The weight file itself shrinks by only about 6%, which is expected since the key and value projections share a small part of the full eight-billion-parameter model.

Latency. We add a dedicated module meant to deal with latency. Single-stream decoding, with a fixed-shape cache and a captured CUDA graph, against the standard Hugging Face path:

Context	Dense	NovaKV	
256	27.00	10.51	2.57×
2048	27.19	12.79	2.13×
8192	27.39	18.77	1.46×

in milliseconds per token, all measured in one session, with an argmax match against the dense model checked at every decoding step. The dense baseline is flat across context because at batch one it is bound by CPU dispatch rather than by the cache (this is why the advantage narrows as context grows). That baseline is not itself CUDA-graph captured.

4 Scope

Compression is applied to a frozen model and requires no retraining of it. The published checkpoints cover Llama-3.1-8B-Instruct and Mistral-7B-Instruct-v0.3.

Code generation carries the cost: around ten points of pass@1 on both models and both benchmarks, against one to four points elsewhere. Training on a code-oriented corpus recovers a large part of it, at the expense of long-document comprehension; which trade to make depends on the intended use. Note that we chose a final number of 45% of memory reduction, but this percent can be aither pushed further, or lower (for instance to enhance code generation). Another leverage for code is the possibility to adapt the dedicated dataset used during training.

5 Perspective: batched inference through geometry

Beyond a batch size of one the compressed path is slower than dense. Rebuilding the keys costs arithmetic that grows with the batch, while the dispatch overheadthe CUDA graph removes does not. Single-stream and local inference is therefore the regime this currently serves.

The next step is to reduce that reconstruction cost. When a single basis is shared across all key heads of a layer, each head must traverse the entire shared latent to rebuild its own keys. An intermediate option is to add a clever way to group heads instead, for instance using chordal similarity on the Grassmannian, keeping unchanged cache size but enhancing token flow. The gain is currently masked by memory traffic, which dominates both configurations and is indifferent to group size. A way to make it visible requires a fused kernel for the grouped case, which is the work in progress.

6 Credits and availability

The rank selection by trained soft threshold is from STAR-KV [1] (ICML 2026). The closed-form offline value calibration is from ReCalKV [2]. What is specific here is the reading of the two as one axis, the whitened shared basis for the keys, the asymmetric pairing, and the evaluation.

Inference and evaluation code, together with the compressed weights, are available on github : `https://github.com/declercqcharles/NovaKV-public`.

References

[1] *STAR-KV: Low-Rank KV Cache Compression via Soft Thresholding for Adaptive Rank Control*, ICML 2026, arXiv:2606.08382.

[2] *ReCalKV: Low-Rank KV Cache Compression with Offline Value Calibration.* ICLR 2025 proceedings, arXiv:2505.24357