QJL: 1-Bit Quantized JL transform for KV Cache Quantization

Amir Zandieh [1] Majid Daliri [2] Insu Han [3]

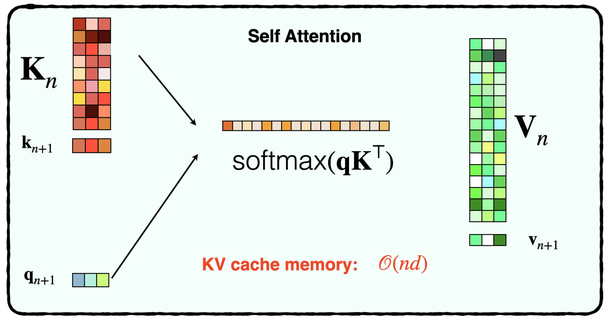
[1]Google Research [2]New York University [3]KAIST

Introduction to KV Cache

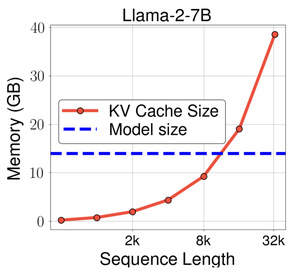
Large Language Models (LLMs) generate text token by token. To avoid recomputing attention for past tokens, we store **keys** (K) and **values** (V) in a **KV cache**.

$$\texttt{Att} = \texttt{Softmax}\left(\frac{\mathbf{QK}^T}{\sqrt{d}}\right)\mathbf{V}, \quad Q, K, V \in \mathbb{R}^{n\times d} \quad \text{(Prefill)}$$

$$\texttt{Att} = \texttt{Softmax}\left(\frac{\mathbf{qK}^T}{\sqrt{d}}\right)\mathbf{V}, \quad q \in \mathbb{R}^{1\times d} \quad \text{(Generation)}$$



The KV cache grows **linearly** with sequence length, increasing memory usage. For **reasoning, retrieval, and long-context tasks**, this limits scalability.
Compression is needed to reduce memory while preserving performance. As shown, the KV cache can surpass model size, making efficient storage essential.



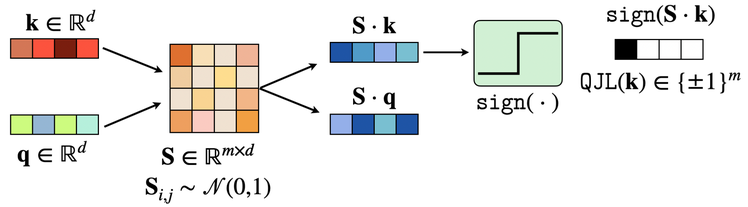
Previous Works

Methods for reducing KV cache size in autoregressive transformers fall into three categories:

- **Quantization** reduces memory by storing KV pairs with lower precision (*GEAR* [1], *KIVI* [2], *KVQuant* [3]).
- **Eviction** removes selected tokens to maintain a fixed cache size (*H2O* [4], *Sink*[5], *SubGen* [6]).
- **Offloading** shifts KV storage to external memory (*InstInfer* [7], *InfiniGen* [8], *ShadowKV* [9]).

Scan the QR code to access the full implementation. Alternatively, visit: github.com/amirzandieh/QJL

Quantized Johnson-Lindenstrauss (QJL)

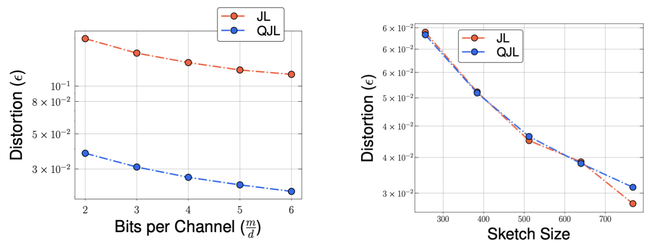


QJL Theorem. Let $S \in \mathbb{R}^{m\times d}$ be a random projection matrix, where $m = \frac{\log\left(\frac{1}{\delta}\right)}{\epsilon^2}$ with probability $1-\delta$. For any unit-norm vectors $\mathbf{q}, \mathbf{k} \in \mathbb{R}^d$ (i.e., $\|\mathbf{q}\| = \|\mathbf{k}\| = 1$), the following holds:

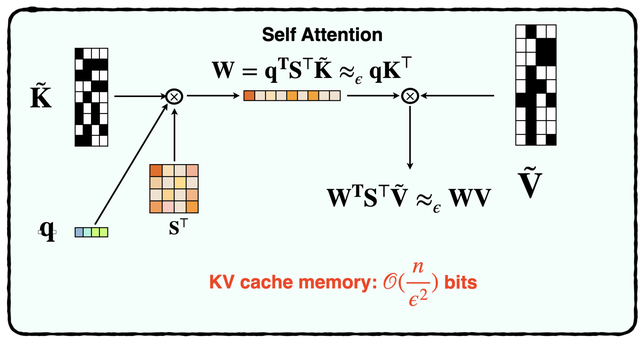
$$\left|\langle \mathbf{q}, S^T \operatorname{sign}(S\mathbf{k})\rangle - \langle \mathbf{q}, \mathbf{k}\rangle\right| \le \epsilon$$

Thus, the QJL projection preserves inner products with an additive error at most ϵ, enabling extreme compression while maintaining theoretical guarantees.

Compared to JL, QJL achieves the same theoretical distortion bound and performs similarly in practice for a given sketch size. However, QJL requires significantly fewer bits—only 1 bit per entry—leading to extreme compression while preserving accuracy.

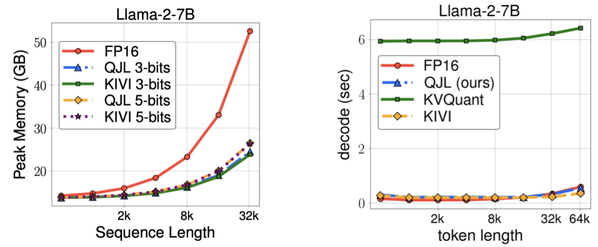


KV Cache Quantization with QJL



Memory and Runtime Evaluation

QJL drastically reduces **peak memory** compared to FP16 while matching KIVI at 3–5 bits. It maintains **fast decoding speed** with minimal memory use, unlike KVQuant, which suffers from higher runtime overhead.



Comprehensive Performance Evaluation of QJL on LongBench

QJL achieves superior performance compared to other quantization methods while significantly reducing memory usage. Despite using only **3, 4, or 5 bits**, it maintains accuracy on par with FP16, demonstrating that extreme quantization can be achieved without sacrificing performance.

Methods	Bits	Datasets from LongBench					
		NarrativeQA	Qasper	MultiQA-en	MultifQA-zh	HotpotQA	2WikiMultiQA
FP16 (**Longchat-7b-v1.5-32k**)	16	20.79	29.42	42.83	34.33	33.05	24.14
KIVI [2]	3	**20.96**	**29.01**	40.93	**34.75**	32.79	23.01
QJL (ours)	3	20.67	28.48	**40.94**	29.71	**35.62**	**23.60**
KVQuant [3]	4.3	20.14	28.77	**44.22**	**34.44**	34.06	**23.05**
QJL (ours)	4.3	**20.72**	**30.02**	41.18	31.73	**34.22**	22.63
KIVI [2]	5	20.49	28.90	**43.24**	**34.66**	33.07	**24.86**
QJL (ours)	5	**21.09**	**29.11**	41.58	31.86	**35.65**	24.61

Methods	Bits	Datasets from LongBench								Avg.
		Qmsum	Qasper	MultiNews	TREC	TriviaQA	SAMSum	LCC	RepoBench-P	
BP16 (**Llama-3.1-8B-Instruct**)	16	25.22	44.98	26.99	73.5	91.79	43.87	62.99	56.39	53.47
QJL (ours)	3	23.68	42.76	26.82	72.5	88.55	43.43	63.98	56.09	52.48
QJL (ours)	4	23.40	43.09	27.02	72.5	89.69	43.30	64.75	56.79	52.82

Performance on Shorter-Context Datasets

QJL maintains strong accuracy even on datasets with shorter context lengths. Despite extreme quantization, it performs comparably to FP16, demonstrating its versatility across different sequence lengths.

Models	Methods	Bits	Datasets from LM-eval				
			Lambada-OpenAI	HellaSwag	PIQA	MathQA	MMLU
Llama-2-7B	FP16 (baseline)	16	73.90	57.18	78.07	28.11	41.85
	KIVI[2]	3	73.88	57.13	78.07	28.11	41.81
	QJL (ours)	3	73.88	57.14	78.07	28.17	41.78
Llama-3-8B	BF16 (baseline)	16	75.59	60.17	79.65	40.64	62.09
	QJL (ours)	3	75.61	60.13	79.87	40.60	62.12

References

[1] GEAR (Kang et al., 2024)
[2] KIVI (Liu et al., 2024)
[3] KVQuant (Hooper et al., 2024)
[4] H2O (Zhang et al., 2023)
[5] Sink (Xian et al., 2024)
[6] SubGen (Zandieh, 2024)
[7] InstInfer (Pan et al., 2024)
[8] InfiniGen (Lee et al., 2024)
[9] ShadowKV (Sun et al., 2021)