IRJCS: International Research Journal of Computer Science
Volume 13, Issue 06, June 2026
ISSN: 2393-9842
https://www.irjcs.com/archives
https://doi.org/10.26562/irjcs.2025.v1306.03

A Comparative Study of KV-Cache Quantisation Methods for Large Language Model Inference

Chinmay C S
Department of Information Science and Engineering,
RV College of Engineering, Bengaluru, India
chinmaycs.is22@rvce.edu.in
https://orcid.org/0009-0007-2418-603X
https://ror.org/05t4ema23

Rekha B S
Department of Information Science and Engineering,
RV College of Engineering, Bengaluru, India
rekhabs@rvce.edu.in
https://ror.org/05t4ema23

Publication History:
Manuscript Reference No: IRJCS/RS/Vol.13/Issue05/JNCS10081
ResearchArticle | Open Access | Double-Blind Peer-Reviewed|Article ID:IRJCS/RS/Vol.13/Issue 06/CSJN26.JNCS10081
Received: 22, April 2025, Revised: 12, May 2026, Accepted: 18, June 2026, Published Online: 06, July 2026.
https://www.irjcs.com/volumes/Vol13/iss-06/03.JNCS10081.pdf
Article Citation:Chinmay,Rekha(2026),A Comparative Study of KV-Cache Quantisation Methods for Large Language Model Inference, IRJCS: International Research Journal of Computer Science, Volume 13, Issue 06 of 2026 pages 607-612
Doi:>https://doi.org/10.26562/irjcs.2026.v1306.03
https://ror.org/05t4ema23 **BibTeX Key:Chinmay@KV-Cache2026.**
IRJCS papers should be cited as IRJCS (International Research Journal of Computer Science, AM Publications, India 2026, ISSN 2393-9842, https://doi.org/10.26562/irjcs.2026.v1306.03. The journal's official abbreviation is IRJCS.
ORCiD: https://orcid.org/0009-0004-9398-7488 About the License: Copyright©2026 copyright by the authors. This article is an open access and license under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

Abstract: A four-billion-parameter language model accumulates several gigabytes of key–value (KV) cache at a context of 32K tokens, often exceeding the size of the model weights. Quantising this cache reduces the memory footprint, but five published methods take different routes to that goal, with non-obvious trade-offs between them. This paper compares six KV-cache representations an uncompressed bf16 reference, KIVI, KVQuant, QuaRot, QJL, and Turbo Quant along five criteria including bit budget, calibration requirement, and decode-time key reconstruction. All numerical figures are taken from the corresponding source publications. Two findings follow. Reported compression ratios range from 2.6× (KIVI at 2-bit) to 6.4× (KVQuant at 2-bit non-uniform), and the gap between nominal bit budget and reported ratio reflects the cost of per-block metadata. QJL and TurboQuant share two properties absent from KIVI, KVQuant, and QuaRot: they require no calibration data, and they compute attention without reconstructing keys. This combination is most valuable in deployments where calibration prompts are unstable or unavailable, or where the same compressed cache spans multiple model variants.

Keywords: Large language models, KV-cache quantisation, model compression, post-training quantisation, attention mechanism, comparative survey, data-oblivious quantisation, random projections

I. INTRODUCTION

Long-context inference in large language models is bound principally by the memory cost of the key–value (KV) cache. Every generated token forces a re-read of the entire stored attention state [6] across all layers and all heads of the network; the cache therefore sits on the critical path of every decode step and grows monotonically as new tokens are appended. The size of the cache scales with the product of model depth, head count, head dimension, sequence length, and per-element precision. Concretely, for a transformer of depth L, with H KV heads, head dimension d, and context length T, full-precision storage occupies $2LHdT$ scalars at width b bytes each. At a context of 32K tokens, a four-billion-parameter network with the typical thirty-two-layer, eight-KV-head, 128-dimensional configuration accumulates several gigabytes of cache state a footprint comparable to, or exceeding, the size of the model weights themselves. Architectural mitigations like multi-query [7] and grouped-query [8] attention reduce this storage by a fixed factor without altering its scaling behaviour. Quantisation of the KV cache has therefore emerged as an active research direction. The KIVI proposal [3] demonstrates that two-bit precision is achievable when the asymmetric per-channel range of post-rotation key activations is exploited through axis-aware grouping. KVQuant [4] couples a calibrated non-uniform datatype, pre-rotation channel statistics for keys, and a sparse outlier path, and the work reports operating contexts in the multi-million-token regime on a single accelerator. QuaRot [5] steps outside the cache-only framing and instead modifies the model itself, embedding orthogonal rotations into the residual stream so that downstream linear quantisation encounters a less outlier-dominated distribution. QJL [2] retains nothing but the sign of a Johnson–Linden strauss-style random projection of every cached vector, with a closed-form constant correcting the resulting estimator's bias. The most recent of the family, TurboQuant [1], composes a Haar-distributed orthogonal transform, mixed-precision distortion-optimal codebooks, and a QJL residual to deliver low-distortion compression at sub-four-bit precision without requiring calibration data.

Beyond language modelling, compression and efficient deployment of neural networks has been explored across many computer science domains, including object detection [12], surveillance and tracking [13], autonomous driving [14], medical image analysis [15], and general information systems [16]. These five methods cover a wide range of design choices. Some require offline calibration on a held-out dataset; others compute statistics online from the live cache or avoid calibration altogether by routing the input through a randomised transform. Some preserve a sparse set of outlier values at full precision; others use mixed-precision codebooks to handle outlier-prone coordinates. Some materialise full-precision keys at decode time; others estimate inner products directly from packed integer representations. These choices interact, and the consequences of any single choice depend on the others, which makes a clear comparative view across these methods valuable as a guide to selection. Two contributions follow. First, an algorithmic comparison of six representative KV-cache quantisation schemes along five criteria that surface the differences between them. Second, a concrete observation: of the five quantising methods surveyed, three (KIVI, KVQuant, QuaRot) require either calibration data, online statistics, or offline weight modifications, while two (QJL and TurboQuant) route inputs through a randomised transform and need none of these. The choice between these two approaches has implications for metadata overhead, decode-time bandwidth, and resilience to distribution shift. All numerical figures used in the comparison are drawn from the cited source publications and are reported with explicit attribution; no new empirical measurements are introduced. The remainder of the paper is organised as follows. Section II reviews KV-cache mechanics and the structural families of quantisation primitives the surveyed methods occupy. Section III presents each method in turn. Section IV describes the methodology of the comparison and the scope of the claims it supports. Section V reports the comparative tables. Section VI discusses the implications. Section VII concludes.

II. BACKGROUND

A. Mechanics of the KV cache

During autoregressive decoding, each attention layer projects every previously emitted token into key and value vectors that subsequent decode steps must consult. The aggregate of this storage the KV cache is therefore a per-request data structure that lives for the duration of a single generation and is read in full on every step. For a token at position t, the attention output is computed as

$$o_t = \text{softmax}(q_t K^T / \sqrt{d}) V_{\leq t} \quad (1)$$

Where q_t denotes the projected query at the current step, and $K_{\leq t}$, $V_{\leq t}$ denote the cached key and value tensors over all preceding positions. Memory consumption, rather than arithmetic intensity, is the primary bottleneck in long-context inference.

B. Three families of quantisation primitives

Methods proposed for the KV cache fall structurally into three groups. *Uniform* schemes apply a linear partition of the value range and store an integer index per element; the naive INT8 baseline and KIVI's two-bit configuration both belong here. Non-uniform schemes precompute distortion-minimising centroids using an assumed input distribution, then quantise to indices into the resulting codebook; KVQuant adopts this strategy with a calibrated four-bit datatype. Data-oblivious schemes apply a randomised transform that pushes the per-coordinate marginal of the input toward a known distribution before quantising with a fixed codebook reused across every layer, every head, and every token. QJL and TurboQuant occupy this third family and inherit its operational advantage of carrying no per-block scale or zero-point alongside the compressed payload.

C. Why cache quantisation is not weight quantisation

Quantisation of model weights is a one-shot offline procedure operating on a fixed parameter tensor; well-established weight-quantisation methods all rest on that assumption. Cache quantisation, in contrast, is an inference-time operation against activations whose statistics drift across context, model depth, and prompt content. The implication runs in two directions. Methods requiring per-block calibration scale poorly with context length because calibration cost grows alongside T.Methods storing per-vector scale-and-zero-point pairs introduce metadata overhead that becomes a substantial fraction of the headline budget at low bit precisions a nominal two-bit scheme that also stores a sixteen-bit scale per sixteen-element block is, when totalled, closer to a three-bit scheme. Bit budget alone is therefore an incomplete summary; full overhead is accounted for in the comparison tables of Section V.

III. KV-CACHE QUANTISATION METHODS

A. Uncompressed bf16 baseline

The uncompressed reference retains every key and value tensor at bfloat16 precision, two bytes per scalar. No algorithmic mechanism is involved no calibration, no metadata, no reconstruction logic and the implementation reduces to whatever the host framework provides natively. Its role in the comparison is twofold: it fixes the quality ceiling against which lossy schemes are evaluated, and it represents the bandwidth-bound regime that subsequent compression schemes attempt to escape.

B. KIVI

Among two-bit cache compression methods, KIVI [3] has the highest citation count to date. Its founding observation is the structural asymmetry of post-rotation activations: keys exhibit pronounced per-channel magnitude variation while values do not. The design therefore groups keys along the channel axis for scale computation and groups values along the token axis, with each tensor quantised independently to two-bit indices accompanied by per-group affine constants. No held-out calibration prompt is needed quantisation statistics are accumulated online from the active cache but a small bfloat16 residual buffer must be retained for the most recent few tokens, whose per-channel statistics remain unstable until enough samples accumulate. The published implementation rests on a fused custom kernel for the asymmetric quantisation step.

Keys are dequantised to floating point during the attention computation, which limits the effective bandwidth saving to the size of the reconstructed working set.

C. KVQuant

KVQuant [4] aggregates several techniques that appeared individually in earlier weight-quantisation work and applies them to the cache. Keys are quantised before rather than after the rotary position transform so that channel-wise statistics remain stable as token position advances. Both key and value tensors are stored using a non-uniform four-bit datatype whose centroids are fitted to the empirical activation distribution observed on a small calibration set; a two-bit variant is also reported. A sparse path retains the heaviest roughly one percent of values at full precision, with per-vector normalisation applied prior to the lossy step. The authors of [4] report extended usable context running into the multi-million-token range on a single accelerator. The cost of these gains is twofold: the method depends on calibration data, and the sparse outlier matrix path required for inference depends on specialised gather primitives. As with KIVI, full-precision keys are materialised at decode time, which precludes purely compressed-domain attention.

D.QuaRot

QuaRot [5] sits in a different lineage and is included here for context. Rather than quantising the cache during inference, it edits the model itself weights, activations, and the cache together by inserting orthogonal (Hadamard) rotations into the residual stream so that no internal representation retains a strongly outlier-dominated distribution. After the rotation, every linear-quantisation scheme benefits. The reported configuration uses four-bit precision for weights, activations, and the cache; the rotation is applied offline to weights and online to activations. The technique is therefore structural rather than cache-only, and it requires write access to the model graph a constraint that excludes deployment paths consuming opaque checkpoints. It is included in the survey because the rotate-then-quantise pattern recurs in QJL and TurboQuant, though applied at a different layer of the stack and against a different invariant.

E.QJL

QJL [2] provides the inner mechanism on which TurboQuant later builds. Its construction starts from the Johnson Lindenstrauss lemma [9]: a randomised projection from $\square^d$ to $\square^m$ preserves inner products in expectation with controllable variance. QJL discards everything but the sign of each projected coordinate, leaving a one-bit-per-dimension representation, and corrects the bias of the resulting inner-product estimator using a closed-form multiplicative constant. The estimator is unbiased and concentrates well; with m on the order of a few hundred, the variance is small enough that downstream softmax behaviour is preserved. Considered as a standalone scheme, QJL is the most aggressive entry in the survey at one bit per coordinate plus a single fp32 norm per vector. The price is estimator variance, which is acceptable for retrieval-style workloads and attention-sink configurations but borderline for full generation. TurboQuant extends QJL by routing most of the bit budget through a distortion-optimal codebook and keeping the QJL projection only for the residual.

F.TurboQuant

TurboQuant [1] applies a Haar-distributed orthogonal transform to each key and value vector prior to quantisation. The rotation drives the per-coordinate marginal of the rotated vector toward an isotropic Gaussian of variance $1/d$ regardless of the input distribution, which means a single distortion-optimal codebook [10], [11] fitted once to that Gaussian target suffices globally across every layer, head, and token, with no calibration data required at deployment time. Keys are quantised by a mixed-precision scheme. A subset of coordinates identified as outlier-prone receives a four-bit codebook (sixteen centroids); the remainder receives a three-bit codebook (eight centroids); and a one-bit QJL residual sketch is retained to maintain unbiasedness of the QK^T estimator that the codebook alone would not deliver. The effective per-element budget for keys works out to roughly three-and-a-half bits. Values use a uniform three-bit codebook tuned for mean-squared distortion. A per-vector L_2 norm in fp32 is the only side information stored alongside the compressed payload. Attention is computed in the compressed domain: the query vector is projected once into the rotated codebook space and once into the projection space, and the inner product QK^T is estimated directly from packed integer representations. Full-precision keys are never reconstructed at decode. The codebook itself amounts to a few hundred bytes of static memory, shared globally, and is therefore effectively free in any deployment scenario.

IV. COMPARISON METHODOLOGY

A. Selection of methods

The six entries in the comparison were chosen to span the published range of approaches rather than to enumerate them exhaustively. The bf16 baseline anchors the quality ceiling. KIVI represents the dominant two-bit point. KVQuant represents calibrated non-uniform quantisation with outlier preservation. QuaRot represents the structural-rotation lineage as applied across the whole model. QJL represents the most aggressive sketch-based estimator. TurboQuant represents a recent composition combining rotation, mixed-precision codebooks, and a residual sketch.

B. Comparison Criteria

Five criteria guide the comparison, each chosen to make the differences between methods concrete:

- Bit budget (K and V separately): per-element precision after accounting for stored quantisation constants where applicable.
- Calibration requirement: whether the method requires a held-out calibration set or a runtime statistics buffer.
- Key reconstruction at decode: whether the method computes the inner product from compressed representations or first reconstructs full-precision keys.

- Per-block constants overhead: the structure and size of any auxiliary metadata stored alongside the compressed payload.
- Central design innovation: the architectural choice that most distinguishes the method from its peers.

C. Scope of numerical claims

Every numerical value reported in the comparative tables is taken directly from the source publication of the method to which it pertains and is cited inline at the table cell. No new measurements are introduced. Direct head-to-head numerical comparison across methods is limited by differences in the model family, hardware platform, and benchmark protocol of each underlying study; the comparison is therefore primarily *algorithmic*, with quantitative figures included for design-point context rather than for ranking.

V. COMPARATIVE ANALYSIS

A. Algorithmic comparison along five criteria

Table I arranges the six surveyed methods against the five criteria set out in Section IV-B.

Table I - Algorithmic Comparison of KV-Cache Quantisation Methods

Method	K bits	V bits	Calibration	K reconstr.	Per-block constants
bf16 baseline	16	16	none	n/a (no quant.)	none
KIVI [3]	2	2	none (online stats)	required	scale + zero-point per group
KVQuant [4]	4 (or 2)*	4 (or 2)*	required (offline NUF fit)	required	NUF datatype+sparse outliers
QuaRot [5]	4	4	offline weight rotation	required	per-tensor scale
QJL [2]	1	1	none (data-oblivious)	not required	fp32 norm per vector
TurboQuant [1]	3.5 (mixed)	3	none (data-oblivious)	not required	fp32 norm per vector

*KVQuant reports both 4-bit and 2-bit non-uniform variants [4]. All method properties shown in this table are derived from the respective cited source publications [1]–[5]. Three observations follow. First, only two entries QJL and TurboQuant avoid reconstructing full-precision keys at decode; every other method materialises keys back to floating point for the QK^T matrix multiply, capping the bandwidth saving extractable at long context. Second, only those same two combine calibration-free operation with a data-oblivious codebook structure; the other entries each impose at least one of a held-out calibration set, an online statistics buffer, or an offline weight rotation. Third, TurboQuant sits at a Pareto point between QJL (extreme compression, high estimator variance) and the KIVI/KVQuant pair (moderate compression, lower variance): the mixed-precision codebook paired with a QJL residual delivers both an unbiased estimator and a moderate effective bit budget.

B. Compression and quality from the source publications

Table II collects headline numbers reported by each method's authors. The figures are not co-measured; each entry was produced on different hardware, against a different model family, and under a different benchmark protocol. Their value here is in characterising the operating point each method was designed for, not in supporting a head-to-head numerical ranking.

Table II - Reported Compression and Quality Figures Across Surveyed Methods

Method	Setting (as reported)	Compression	Quality metric (as reported)
KIVI [3]	Llama-2-7B, 2-bit	~2.6× [3]	Near-baseline WikiText-2 perplexity; minor gap on LM-Eval benchmarks [3]
KVQuant [4], 4-bit NUF	Llama-7B, A100, 4-bit non-uniform	~3.7× [4]	<0.1 perplexity gap on WikiText-2 with calibrated NUF datatype [4]
KVQuant [4], 2-bit NUF	Llama-7B, A100, 2-bit non-uniform	~6.4× [4]	Modest perplexity gap; multi-million-token context preserved [4]
QuaRot [5]	Llama-2-7B, 4-bit weights + KV	~3.6× [5]	<1% accuracy drop on common zero-shot reasoning benchmarks [5]
QJL [2]	Llama-2-7B, 1-bit JL sketch	~5× [2]	Analytically bounded inner-product MSE; downstream quality preserved on reported tasks [2]
TurboQuant [1]	Mixed 3-/3.5-bit, codebook-based	~3.9× [1]	Near-optimal distortion at sub-four-bit precision; quality on par with baseline on reported tasks [1]

Note: All compression ratios and quality figures shown above are referenced from cited sources [1]–[5]. Each row corresponds to a different model family, hardware platform, and benchmark protocol and the rows are therefore not directly comparable to one another. Compression ratios include metadata overhead where reported by the original authors. The qualitative reading of Table II is that compression ratio is not a sufficient single-number summary. KIVI reports a smaller nominal ratio than its two-bit budget alone would suggest because per-group affine constants erode the headline budget. KVQuant's higher ratios depend on the joint availability of a calibration set and a specialised path for the sparse outlier matmul, neither of which is automatic. QJL records the most aggressive nominal compression but pays for it in estimator variance. TurboQuant's reported figure is competitive with KVQuant's four-bit operating point while preserving the data-oblivious structure that other rows give up.

C. Deployment properties

Table III reprojects the same six methods through a deployment-engineer lens, focusing on properties that determine the cost of integration: whether calibration data is required, whether keys are reconstructed at decode, and whether attention can be computed directly in the compressed domain.

IRJCS: International Research Journal of Computer Science
Volume 13, Issue 06, June 2026
ISSN: 2393-9842
https://www.irjcs.com/archives
https://doi.org/10.26562/irjcs.2025.v1306.03

Table III - Deployment Properties of the Surveyed Methods

Method	No calib.	No K recon.	Compr. attn.
bf16 baseline	yes	yes (no quant.)	n/a
KIVI [3]	yes	no	no
KVQuant [4]	no	no	no
QuaRot [5]	partial	no	no
QJL [2]	yes	yes	yes
TurboQuant [1]	yes	yes	yes

All entries derived from the cited source publications [1]–[5]. Two rows present favourably across all three columns: QJL and TurboQuant. QJL pays for this with higher estimator variance and is best suited to retrieval-style workloads or attention-sink configurations in which occasional inner-product errors are absorbed by the softmax. TurboQuant trades a fraction of QJL's nominal compression for bounded variance and is, in the surveyed set, the only entry simultaneously calibration-free, free of decode-time key reconstruction, and quality-preserving on full generation tasks.

VI. DISCUSSION

A. What QJL and TurboQuant share

QJL and TurboQuant share a single design decision: they push the cached vectors through a randomised transform whose output marginal is mathematically known in advance. Once the marginal is fixed, a single offline-computed codebook becomes globally applicable, and no calibration prompt, no online statistics buffer, and no per-vector or per-block affine pair needs to ride alongside the compressed payload. This property eliminates the dependency on specialised online quantisation operations that the per-channel, per-group designs of KIVI and KVQuant inherit. It is also what enables compressed-domain attention when the codebook is shared across all tensors, the query can be projected once into the codebook space and re-used for every inner-product estimate, removing the bandwidth cost of materialising keys back to floating point at every step.

B. Where each method trades off

The data-oblivious schemes pay in two places. Estimator variance is one: QJL's standalone one-bit representation drives variance up to the limit of what attention can tolerate, and TurboQuant's residual sketch is precisely a mechanism for absorbing the bias that the mixed-precision codebook alone would leave behind. Prefill cost is the other: the rotation and quantisation must be applied to every prompt token before compressed-domain decoding can begin, and on short prompts the per-token overhead is visible in wall-clock latency. The calibrated schemes pay in the opposite direction. KIVI's online per-channel statistics accumulate stably only after a warmup, KVQuant's outlier path depends on a specialised sparse-gather primitive, and QuaRot needs offline access to the model grapheach of which excludes a deployment pattern in which the orchestrator handles opaque checkpoints rather than mutable code.

C. Method selection by deployment scenario

Table IV maps each deployment scenario to the method best supported by the comparison.

Table IV - Recommended Method per Deployment Scenario

Scenario	Recommendation
Calibration budget available, maximum compression priority	KVQuant [4]
Simple deployment, two-bit budget tolerable	KIVI [3]
Structural model edits acceptable, four-bit across the stack	QuaRot [5]
Memory-constrained deployment, long context, quality-sensitive load	TurboQuant [1]
Retrieval / attention-sink workload, tolerant to estimator variance	QJL [2]
Resource-constrained, rough approximation acceptable	Naive INT8

The map is not a ranking. Each entry represents the best choice within its row and is the wrong choice elsewhere; the survey's contribution is to make that mapping explicit rather than to nominate a single winner.

D. Data-oblivious quantisation as a deployment property

A property that does not appear in any single compression number but emerges across Tables I–III is the operational value of being data-oblivious. A calibration-dependent method introduces three operational risks in deployment: the calibration prompt must be selected and stored, the calibration must be repeated whenever the underlying model or its fine-tune changes, and the calibration distribution may shift away from the production distribution in ways that are difficult to detect at inference time. A data-oblivious method has none of these failure modes by construction. Its codebook is fitted to a mathematical distribution the post-rotation Gaussian, who's marginal, is fixed by the random orthogonal transform irrespective of input and remains valid indefinitely. For deployment paths that expect the same compressed cache to be consumed by different model variants on different devices, that invariance is worth more in practice than a small additional fraction of compression.

VII. CONCLUSION

This paper has surveyed six representative methods for quantising the KV cache of large language models. A comparison along five criteria showed that the five quantising methods divide by their treatment of the input distribution. KIVI, KVQuant, and QuaRot rely on online statistics, offline calibration, or offline weight modifications respectively. QJL and TurboQuant compose a randomised transform with a fixed codebook, requiring no calibration data. Each method remains the right choice in its own deployment scenario;

the survey's contribution is to make those mappings explicit and to ground them in figures sourced directly from the method papers themselves. The main takeaway is concrete. Methods that route inputs through a random transform with a known target distribution can use one fixed codebook for all layers, heads, and tokens. This eliminates the calibration step, the per-block metadata, and the key reconstruction at decode that the other methods require. QJL and TurboQuant share this design choice at different bit budgets, with different variance trade-offs.

DECLARATION ON GENERATIVE AI

Generative AI tools were used to support drafting and language refinement during the preparation of this manuscript. All scientific content, the comparative analysis, the selection of methods, and the conclusions are the authors' own. The authors take full responsibility for the accuracy, originality, and integrity of the work.

REFERENCES

1. A.Zandieh, M.Daliri, A.Hadian, and V.Mirrokni, "TurboQuant: Online Vector Quantization with Near-optimal Distortion Rate," arXiv preprint arXiv:2504.19874, 2025.
2. A.Zandieh, M.Daliri, and I.Han, "QJL: 1-Bit Quantized JL Transform for KV Cache Quantization with Zero Overhead," arXiv preprint arXiv:2406.03482, 2024. https://doi.org/10.1609/aaai.v39i24.34773
3. Z.Liu, J.Yuan, H.Jin, S.Zhong, Z.Xu, V.Braverman, B.Chen, and X.Hu, "KIVI: A Tuning-Free Asymmetric 2bit Quantization for KV Cache," in Proc. 41st Int. Conf. Machine Learning (ICML), 2024.
4. C.Hooper, S.Kim, H.Mohammadzadeh, M.W.Mahoney, Y.S.Shao, K.Keutzer, and A.Gholami, "KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization," in Advances in Neural Information Processing Systems 37, 2024. https://doi.org/10.52202/079017-0040
5. S.Ashkboos, A.Mohtashami, M.L.Croci, B.Li,P.Cameron, M.Jaggi, D.Alistarh, T.Hoefler, and J.Hensman, "QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs," in Advances in Neural Information Processing Systems 37, 2024. https://doi.org/10.52202/079017-3180
6. A.Vaswani, N.Shazeer, N.Parmar, J.Uszkoreit, L.Jones, A.N.Gomez, L.Kaiser, and I.Polosukhin, "Attention Is All You Need," in Advances in Neural Information Processing Systems 30, 2017, pp. 5998–6008.
7. N.Shazeer, "Fast Transformer Decoding: One Write-Head is All You Need," arXiv preprint arXiv:1911.02150, 2019.
8. J.Ainslie, J.Lee-Thorp, M.de Jong, Y.Zemlyanskiy, F.Lebrón, and S.Sanghai, "GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints," in Proc. 2023 Conf. Empirical Methods in NLP (EMNLP), 2023. https://doi.org/10.18653/v1/2023.emnlp-main.298
9. W.B.Johnson and J.Lindenstrauss, "Extensions of Lipschitz Mappings into a Hilbert Space," Contemporary Mathematics, vol. 26, pp. 189–206, 1984. https://doi.org/10.1090/conm/026/737400
10. S.P.Lloyd, "Least Squares Quantization in PCM," IEEE Trans. Information Theory, vol. 28, no. 2, pp. 129–137, 1982. https://doi.org/10.1109/TIT.1982.1056489
11. J.Max, "Quantizing for Minimum Distortion," IRE Trans. Information Theory, vol. 6, no. 1, pp. 7–12, 1960. https://doi.org/10.1109/TIT.1960.1057548
12. "Object Detection Using Machine Learning," IRJCS:: International Research Journal of Computer Science, AM Publications, India, 2020. DOI: https://doi.org/10.26562/IRJCS.2020.APCS10083
13. "Criminal Recognition and Tracking System," IRJCS:: International Research Journal of Computer Science, AM Publications, India, 2019. DOI: https://doi.org/10.26562/IRJCS.2019.JNCS10095
14. "Self-Driving Car Simulation," IRJCS:: International Research Journal of Computer Science, AM Publications, India, vol. 7, no. 5, 2020. DOI: https://doi.org/10.26562/IRJCS.2020.V0705.002
15. G.E. and M. Sathikraja, "Diabetic Retinopathy Detection Using Convolutional Neural Networks," IRJCS:: International Research Journal of Computer Science, vol. 11, no. 4, 2024. DOI: https://doi.org/10.26562/irjcs.2024.v1104.47
16. A.Raj, K.Roy, N.Kumar, and M.Sakthivel, "College Management System," IRJCS:: International Research Journal of Computer Science, vol. 11, no. 4, 2024. DOI: https://doi.org/10.26562/irjcs.2024.v1104.08