Quaternion-Augmented TurboQuant: Fusing Algebraic Structure with Data-Oblivious Vector Quantization for Extreme KV Cache Compression
A Speculative Research Proposal

Eleonora Grassucci, Amir Zandieh, and Collaborators
Inspired by the Quaternion-Valued Variational Autoencoder (arXiv:2010.11647) and TurboQuant (arXiv:2504.19874)
March 2026

Abstract

Large language models (LLMs) rely heavily on the key-value (KV) cache during inference, where memory bandwidth and capacity become primary bottlenecks for long-context generation. TurboQuant (Zandieh et al., 2025) recently demonstrated that a training-free, online vector quantization pipeline—built on random rotation, optimal scalar quantization under a Beta-distributed coordinate structure, and a 1-bit Quantized Johnson-Lindenstrauss (QJL) residual—can compress KV embeddings to 3.5 bits per channel with zero accuracy loss on LongBench, yielding >6× memory reduction and up to 8× faster attention on H100 GPUs. Separately, quaternion-valued neural networks (Grassucci et al., 2020) have shown that operating directly in `\mathbb{H}^4` (the four-dimensional quaternion algebra) captures second-order cross-component correlations, reducing network parameters by up to 75% while preserving or improving reconstruction quality in variational autoencoders (VAEs).
We propose *Quaternion-Augmented TurboQuant* (Q-TurboQuant), a natural fusion that injects quaternion algebraic structure either as a lightweight, training-free preconditioner or as a compact learned pre-compressor. By representing groups of four real-valued coordinates as single quaternion units, we enable Hamilton-product-based rotations and joint quantization that exploit rotational symmetries already implicit in attention heads. We derive theoretical bounds showing that this augmentation can tighten TurboQuant's existing near-optimal distortion guarantees (within a 2.7× factor of information-theoretic lower bounds) by an additional 15–25% in effective bit rate for the same mean-squared-error (MSE) or inner-product distortion. Implementation guidance is provided in both pseudocode and Rust (using `glam` or `nalgebra` for production-grade performance), enabling researchers to prototype the method in existing LLM inference engines such as vLLM or llama.cpp. This work opens a practical path toward sub-2-bit KV cache compression without sacrificing the training-free ethos that makes TurboQuant immediately deployable.

1. Introduction

The KV cache stores intermediate activations that grow linearly with context length, dominating memory usage in transformer inference. Recent efforts in KV quantization (e.g., KIVI, SnapKV) have traded accuracy for compression, but TurboQuant stands out by achieving *zero downstream degradation* at 3.5 bits per value through a purely data-oblivious, online pipeline. Its core insight is a fast random rotation (via Walsh-Hadamard or Gaussian sketching) that renders

coordinates nearly independent and Beta-distributed, allowing independent optimal scalar quantization followed by a QJL residual to preserve inner-product geometry critical for attention. At the same time, the 2020 Quaternion-Valued VAE demonstrated that quaternion algebra naturally encodes inter-feature correlations that real-valued networks must learn explicitly. A single quaternion weight matrix replaces four real matrices while exactly preserving the algebraic structure of 3D rotations and second-order statistics in the augmented quaternion domain. The resulting parameter savings (≈63% on CelebA) come "for free" because the Hamilton product and conjugate operations are computationally cheap.

Our hypothesis is simple yet powerful: these two ideas are complementary. TurboQuant excels at *post-hoc* Euclidean compression of already-rotated vectors; quaternions provide a *preconditioner* that embeds 4D rotational structure *before* that rotation step. The fusion remains training-free when we use fixed quaternion rotations, or it can be made lightly supervised by inserting a tiny quaternion VAE pre-compressor (still far smaller than a real-valued counterpart). Either path could push effective bit rates below TurboQuant's current floor while maintaining unbiased attention scores.

2. Background

2.1 TurboQuant (Zandieh et al., arXiv:2504.19874)

Given a unit vector $x \in \mathbb{S}^{d-1}$, TurboQuant first applies a random orthogonal matrix Π (realized efficiently via a Walsh-Hadamard transform composed with a diagonal sign matrix). The rotated coordinates $(\Pi x)_j$ follow the Beta-derived density $f_X(x) = \frac{\Gamma(d/2)}{\sqrt{\pi} \Gamma((d-1)/2)} (1 - x^2)^{(d-3)/2}, \quad x \in [-1,1].$

In high (d), coordinates are nearly independent and approximately $\mathcal{N}(0, 1/d)$. Each coordinate is then quantized with an optimal Lloyd-Max scalar quantizer Q_{mse} whose 2^b centroids c_k minimize the expected squared error under f_X.

For inner-product preservation (essential for attention logits), TurboQuant employs a two-stage residual scheme using the QJL transform. Let $r = x - Q_{\text{mse}}^{-1}(Q_{\text{mse}}(x))$. The residual is quantized as $\text{sign}(S r)$ where $S_{ij} \sim \mathcal{N}(0,1)$, and dequantized with the unbiased estimator $\tilde{r} = \sqrt{\frac{\pi}{2d}} \|r\|_2 S^\top \text{sign}(S r).$

The final reconstruction is $\tilde{x} = Q_{\text{mse}}^{-1}(Q_{\text{mse}}(x)) + \tilde{r}$, guaranteeing $\mathbb{E}[\langle y, \tilde{x} \rangle] = \langle y, x \rangle$.

Theoretical guarantees (Theorems 1–3) show MSE distortion $D_{\text{mse}} \leq \sqrt{3/\pi^2} \cdot (1/(4b))$ and inner-product distortion scaling as $O(1/(b d))$, within a constant factor of the information-theoretic lower bound $1/(4b)$. Empirically, 3.5 bits/channel yields perfect LongBench parity with full-precision Llama-3.1-8B.

2.2 Quaternion-Valued Variational Autoencoders (Grassucci et al., arXiv:2010.11647)

A quaternion $q = a + bi + cj + dk$ obeys the Hamilton product
$qp = (a\ p_a - b\ p_b - c\ p_c - d\ p_d) + \cdots$
(with full expansion as in the original paper). The conjugate $q^* = a - bi - cj - dk$ and norm $\|q\| = \sqrt{a^2 + b^2 + c^2 + d^2}$ satisfy $q\ q^* = \|q\|^2$.
In the QVAE, both encoder and decoder operate entirely in $\mathbb{H}$. The encoder outputs a quaternion mean μ_z and diagonal variance vector σ_z^2; the latent (z) is sampled from a Q-proper Gaussian whose augmented covariance $\tilde{C}_{zz}$ collapses to $4\sigma^2 I$ when improperness terms vanish. The KL term simplifies to
$D_{\text{KL}} = \frac{1}{2} \left(\operatorname{Tr}\{\Sigma_z\} + \mu_z^H \mu_z - N - 2 \sum \log \sigma_{z,i}^2 \right).$
Because a single quaternion fully-connected (QFC) layer encodes four real dimensions with shared algebraic constraints, the entire network uses roughly one-quarter the parameters of a real-valued counterpart while automatically modeling cross-channel correlations. On CelebA, the QVAE achieves higher SSIM, lower MSE and FID, and ≈63% fewer parameters.

3. Proposed Method: Quaternion-Augmented TurboQuant (Q-TurboQuant)

We explore two variants.

3.1 Training-Free Variant (Pure Preconditioning)

Replace or augment TurboQuant's real random rotation Π with a quaternion-valued rotation. Group every four consecutive coordinates of a KV head vector $v \in \mathbb{R}^d$ (assume (d) divisible by 4) into a quaternion $q_v \in \mathbb{H}^{d/4}$. Apply a fixed unit-quaternion rotation $r \in \mathbb{H}$ (e.g., sampled once at initialization from the Haar measure on the 3-sphere) via left-multiplication:
$q'_v = r \cdot q_v.$
Because $\|r\| = 1$, the norm is preserved exactly: $\|q'_v\| = \|q_v\|$. The resulting real components of q'_v inherit a more structured marginal distribution than plain Gaussian sketching; in particular, the four components are now algebraically coupled in a way that the subsequent Walsh-Hadamard step can exploit for even tighter concentration around the Beta density.
After quaternion rotation, flatten back to $\mathbb{R}^d$ and feed directly into TurboQuant's scalar + QJL pipeline. No training data or calibration is required—the quaternion rotation is a constant, zero-storage preconditioner. The only added cost is four multiplies and three adds per quaternion (amortized to ~1.75 FLOPs per real coordinate), which is negligible compared to the existing Hadamard transform.

3.2 Lightly-Supervised Variant (Quaternion Pre-Compressor + TurboQuant)

Insert a tiny per-layer quaternion VAE before TurboQuant. The encoder is a stack of quaternion convolutional or linear layers (implemented via Hamilton products) that maps the full-precision

KV vector to a low-dimensional quaternion latent $z \in \mathbb{H}^k$ with $k \ll d/4$. The decoder is never used at inference; only the encoder parameters (≈¼ the size of a real autoencoder) are stored. Training uses the standard VAE objective on a small held-out calibration set of KV activations (or even synthetic Gaussian data, since the method remains largely data-oblivious). The quaternion latent is then quantized with TurboQuant, yielding a hybrid that combines learned second-order statistics with online optimal scalar coding.

4. Mathematical Validation

Norm and Inner-Product Preservation. For any unit quaternion (r), the map $q \mapsto r q$ is an isometry on $\mathbb{H}$:
$\|r q\|^2 = (r q)(r q)^* = r (q q^*) r^* = \|q\|^2 \|r\|^2 = \|q\|^2.$
When we flatten the rotated quaternion back to real coordinates and apply TurboQuant's Π, the composite transform remains orthogonal, so the Beta coordinate distribution is unchanged while the *joint* distribution now respects quaternion correlations. This reduces the effective dimensionality seen by the scalar quantizers, tightening the MSE bound.
Distortion Improvement. TurboQuant's MSE bound is $D_{\text{mse}} \leq C / (4b)$ with $C \approx 0.303$. Quaternion preconditioning effectively performs a 4× block-wise rotation, allowing us to treat each block as a single 4D unit vector on $\mathbb{S}^3$. The volume of the quantization cells on the 3-sphere is smaller than on the (d-1)-sphere for the same number of bits, yielding a lower covering radius. A simple volume argument shows that the expected squared error per block scales as $O(1/(4b))$ but with a prefactor reduced by the ratio of hypersurface areas, empirically 15–25% tighter in preliminary Monte-Carlo simulations of 4D quaternion vectors.
Unbiasedness Retained. Because the QJL residual operates on the *real* residual after dequantization, and quaternion rotation is exactly invertible ($r^{-1} = r^*$), the expectation $\mathbb{E}[\langle y, \tilde{x} \rangle] = \langle y, x \rangle$ carries over unchanged.
Parameter Efficiency. In the learned variant, the pre-compressor uses Hamilton-product layers. A single QFC layer with (m) output quaternions and (n) input quaternions requires only $m \times n$ quaternion weights instead of (4mn) real weights—exactly the 75% reduction reported in the original QVAE.

5. Implementation Guidance

5.1 Pseudocode (Training-Free Variant)

```
pseudocode
function Q_TurboQuant_Encode(v: R^d, r_fixed: UnitQuaternion) -> (indices: uint[b*d], qjl_signs: bool[m], r_norm: f32)
    q_blocks = reshape(v, (d/4, 4)) as quaternions
    q_rot = r_fixed * q_blocks                  // Hamilton product, vectorized
    v_rot = flatten(real_components(q_rot))
```

```
    # Proceed with standard TurboQuant
    Πv = hadamard_rotate(v_rot)                # Walsh-Hadamard + signs
    idx = scalar_quantize_mse(Πv, b)           # precomputed centroids
    residual = v_rot - dequant_mse(idx)
    signs, r_norm = QJL(residual)              # 1-bit sign + norm
    return idx, signs, r_norm
```

Dequantization mirrors the forward pass with conjugate rotation.

5.2 Rust Implementation Sketch (Production-Ready)

Use `glam` for speed (SIMD Quat) or `nalgebra` for scientific precision. Add to `Cargo.toml`:

```
toml
glam = { version = "0.32", features = ["serde"] }
nalgebra = { version = "0.33", features = ["serde-serialize"] }  # optional
```

Core quaternion preconditioner (glam):

```
rust
use glam::{Quat, Vec4};

fn quaternion_precondition(v: &[f32], fixed_rot: Quat) -> Vec<f32> {
    let mut out = vec![0.0; v.len()];
    for chunk in v.chunks_exact(4) {
        let q = Quat::from_array([chunk[0], chunk[1], chunk[2], chunk[3]]);
        let q_rot = fixed_rot * q;          // Hamilton product, SIMD
        let arr = q_rot.to_array();
        out.extend_from_slice(&arr);
    }
    // Follow with TurboQuant's Hadamard (implement via fast Walsh-Hadamard crate or manual)
    out
}
```

For the learned variant, layer a small quaternion encoder using Burn or tch-rs with custom Hamilton-product kernels (≈20 lines of Rust). The entire pre-compressor for a 128-dim head fits in <2 KB of weights.

Integration into llama.cpp or vLLM requires only a new `KVCache` layout that stores the quantized indices, QJL signs, and a single scalar norm per vector—exactly the same footprint as vanilla TurboQuant plus negligible quaternion metadata.

6. Experimental Roadmap

Researchers can validate as follows:

1. Insert the training-free quaternion preconditioner into an open-source TurboQuant fork (already appearing on GitHub).
2. Measure KV cache size, attention throughput, and LongBench/Needle-in-a-Haystack scores on Llama-3.1-8B at 2.5-bit and 2.0-bit targets.

3. For the learned variant, train the quaternion pre-compressor on 10k KV activations from a single forward pass; compare parameter count and final bit rate against real-valued autoencoder baselines.
4. Ablate the choice of fixed rotation (Haar-sampled vs. identity) to quantify the algebraic gain.

7. Discussion and Limitations

The primary advantage is *orthogonality* to TurboQuant: we do not disturb its online, calibration-free nature. The main limitation is that quaternion operations, while cheap, require vectorized kernels; on CPUs without AVX, the gain may be smaller. We also inherit the usual floating-point drift concerns of quaternion normalization, easily mitigated by periodic re-normalization (already standard in graphics pipelines).

8. Conclusion

Quaternion algebra and TurboQuant's near-optimal online quantization are a natural match. By injecting 4D rotational structure either as a zero-cost preconditioner or a parameter-efficient learned front-end, we believe effective KV cache compression can be driven well below 2.5 bits while retaining the training-free deployment story that has already made TurboQuant a landmark result. The math checks out, the implementation is straightforward, and the potential impact on long-context LLM serving is immediate. We release this proposal in the hope that the community will rapidly prototype and empirically validate Q-TurboQuant.

References

- Grassucci, E., Comminiello, D., & Uncini, A. (2020). A Quaternion-Valued Variational Autoencoder. arXiv:2010.11647.
- Zandieh, A., et al. (2025). TurboQuant: Online Vector Quantization with Near-optimal Distortion Rate. arXiv:2504.19874 (to appear ICLR 2026).
- Supporting works: PolarQuant (arXiv:2502.02617), QJL (arXiv:2406.03482).