

KV Cache Quantization

From Quantization Fundamentals to Extreme Low-Bit Compression

Sri Surya Varma

Apoorv Goyal

Rudrapratap Chinhara

Rishit Aggarwal

September 4, 2026

- **Why KV Cache Memory Explodes**

How long context and larger batches turn the cache into the serving bottleneck.

- **Quantization Fundamentals**

Scalar quantization, vector quantization, and the distortion objective.

- **Vector Quantization and TurboQuant**

Using higher-dimensional structure to compress activations more effectively.

- **KV Cache Transform Coding (KVTC)**

PCA-based decorrelation, bit allocation, and lossless coding.

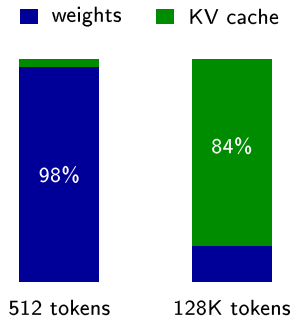
- **Low-Bit KV Quantization: KIVI and KVQuant**

Granularity, outliers, and preserving quality at 2–3 bits.

- **Coupled Quantization**

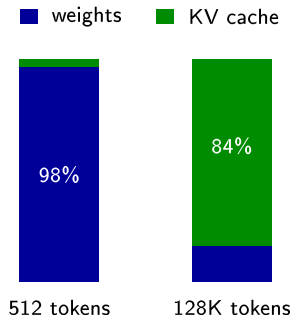
Exploiting inter-channel dependence to push toward the 1-bit regime.

Why KV cache memory explodes



$$\text{KV size} = 2 n h d e \cdot b \cdot l$$

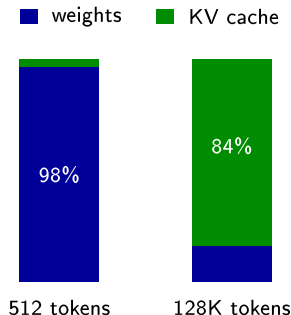
Why KV cache memory explodes



$$\text{KV size} = 2 n h d e \cdot b \cdot l$$

- Grows **linearly** in batch size b and sequence length l — double either one, double the cache.
- At short context, weights dominate. Push l out far enough, or b up far enough, and the **KV cache takes over**.

Why KV cache memory explodes



$$\text{KV size} = 2 n h d e \cdot b \cdot l$$

- Grows **linearly** in batch size b and sequence length l — double either one, double the cache.
- At short context, weights dominate. Push l out far enough, or b up far enough, and the **KV cache takes over**.

Every sequence in a batch has its own past, so there's no way to share that memory across requests — it just keeps stacking up.

Why quantize it

- Fewer heads, evicting tokens, offloading to CPU — all reduce the cache, but each gives something up: retraining, lost context, or latency.

Why quantize it

- Fewer heads, evicting tokens, offloading to CPU — all reduce the cache, but each gives something up: retraining, lost context, or latency.
- **Quantization** takes a different angle: keep every token, keep the architecture, just store each number in fewer bits.

Why quantize it

- Fewer heads, evicting tokens, offloading to CPU — all reduce the cache, but each gives something up: retraining, lost context, or latency.
- **Quantization** takes a different angle: keep every token, keep the architecture, just store each number in fewer bits.

$$Q(X) = \left\lfloor \frac{X - z}{s} \right\rfloor, \quad X' = Q(X) \cdot s + z$$

Why quantize it

- Fewer heads, evicting tokens, offloading to CPU — all reduce the cache, but each gives something up: retraining, lost context, or latency.
- **Quantization** takes a different angle: keep every token, keep the architecture, just store each number in fewer bits.

$$Q(X) = \left\lfloor \frac{X - z}{s} \right\rfloor, \quad X' = Q(X) \cdot s + z$$

It's also the most direct lever on that formula: shrink *bytes/element* and the cache shrinks with it, at any batch size or context length.

Quantising a vector

September 3, 2026

Objective: A Quantization Example

Suppose you are given a value $x \in [-1, 1]$ represented in fp32 (4 bytes).

Objective: A Quantization Example

Suppose you are given a value $x \in [-1, 1]$ represented in fp32 (4 bytes).

Target: Represent x using only **2 bits**.

Objective: A Quantization Example

Suppose you are given a value $x \in [-1, 1]$ represented in fp32 (4 bytes).

Target: Represent x using only **2 bits**.

2 bits $\implies$ 4 possible levels.

Objective: A Quantization Example

Suppose you are given a value $x \in [-1, 1]$ represented in fp32 (4 bytes).

Target: Represent x using only **2 bits**.

2 bits $\implies$ 4 possible levels.

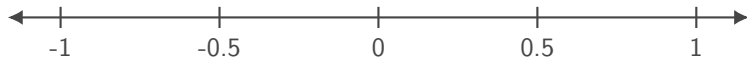


Objective: A Quantization Example

Suppose you are given a value $x \in [-1, 1]$ represented in fp32 (4 bytes).

Target: Represent x using only **2 bits**.

2 bits $\implies$ 4 possible levels.



Objective: A Quantization Example

Suppose you are given a value $x \in [-1, 1]$ represented in fp32 (4 bytes).

Target: Represent x using only **2 bits**.

2 bits $\implies$ 4 possible levels.



Objective: A Quantization Example

Suppose you are given a value $x \in [-1, 1]$ represented in fp32 (4 bytes).

Target: Represent x using only **2 bits**.

2 bits $\implies$ 4 possible levels.



Original x	Quantized $Q(x)$	Reconstructed $Q^{-1}(Q(x))$	Squared Error (SE)
0.8	11	0.75	$(0.05)^2$
-0.1	01	-0.25	$(0.15)^2$
0.25	10	0.25	$(0.00)^2$
-0.8	00	-0.75	$(0.05)^2$

The Formal Problem

Scalar quantization consists of two mappings:

$$\text{Quantizer: } Q : \mathbb{R} \rightarrow \{0, 1\}^b$$

$$\text{DeQuantizer: } Q^{-1} : \{0, 1\}^b \rightarrow \mathbb{R}$$

The Formal Problem

Scalar quantization consists of two mappings:

$$\text{Quantizer: } Q : \mathbb{R} \rightarrow \{0, 1\}^b$$

$$\text{DeQuantizer: } Q^{-1} : \{0, 1\}^b \rightarrow \mathbb{R}$$

Goal: Minimize the “Error” of this representation.

The Formal Problem

Scalar quantization consists of two mappings:

$$\text{Quantizer: } Q : \mathbb{R} \rightarrow \{0, 1\}^b$$

$$\text{DeQuantizer: } Q^{-1} : \{0, 1\}^b \rightarrow \mathbb{R}$$

Goal: Minimize the “Error” of this representation.

Mean Squared Error (MSE)

The Formal Problem

Scalar quantization consists of two mappings:

$$\text{Quantizer: } Q : \mathbb{R} \rightarrow \{0, 1\}^b$$

$$\text{DeQuantizer: } Q^{-1} : \{0, 1\}^b \rightarrow \mathbb{R}$$

Goal: Minimize the “Error” of this representation.

Mean Squared Error (MSE)

Assuming $x \sim f_X(x)$:

Objective Function

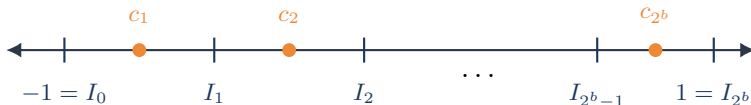
$$\mathbb{E} \left[|x - Q^{-1}(Q(x))|^2 \right]$$

What Can We Control?

To build a b -bit quantizer, we need 2^b intervals and 2^b reconstruction levels.

What Can We Control?

To build a b -bit quantizer, we need 2^b intervals and 2^b reconstruction levels.

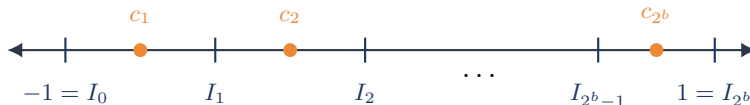


■ **Boundaries** $\{I_i\}$: Defines the quantizer $Q(x)$.

If $x \in [I_{i-1}, I_i) \implies Q(x) = i$ (in binary)

What Can We Control?

To build a b -bit quantizer, we need 2^b intervals and 2^b reconstruction levels.



- **Boundaries** $\{I_i\}$: Defines the quantizer $Q(x)$.
If $x \in [I_{i-1}, I_i) \implies Q(x) = i$ (in binary)
- **Reconstruction Levels** $\{c_i\}$: Defines the dequantizer $Q^{-1}(\hat{x})$.
 $Q^{-1}(i) = c_i$

Defining the Cost Function

By breaking the expected value into the sum over our defined intervals, we can rewrite the Mean Squared Error:

Defining the Cost Function

By breaking the expected value into the sum over our defined intervals, we can rewrite the Mean Squared Error:

$$\begin{aligned} C &= \mathbb{E} \left[|x - Q^{-1}(Q(x))|^2 \right] \\ &= \sum_{i=1}^{2^b} \int_{I_{i-1}}^{I_i} |x - c_i|^2 f_X(x) dx \end{aligned}$$

Defining the Cost Function

By breaking the expected value into the sum over our defined intervals, we can rewrite the Mean Squared Error:

$$\begin{aligned} C &= \mathbb{E} \left[|x - Q^{-1}(Q(x))|^2 \right] \\ &= \sum_{i=1}^{2^b} \int_{I_{i-1}}^{I_i} |x - c_i|^2 f_X(x) dx \end{aligned}$$

The Continuous Quantization Cost

$$C(f_X, \{c_i\}, \{I_i\}, b) = \sum_{i=1}^{2^b} \int_{I_{i-1}}^{I_i} |x - c_i|^2 f_X(x) dx$$

1. Optimal Levels $\{c_i\}$ Given Boundaries $\{I_i\}$

Objective: If the boundaries $\{I_i\}$ are fixed, what are the best $\{c_i\}$?

1. Optimal Levels $\{c_i\}$ Given Boundaries $\{I_i\}$

Objective: If the boundaries $\{I_i\}$ are fixed, what are the best $\{c_i\}$?

Notice that in the total cost C , only *one* integral depends on a specific c_i :

$$C_i := \int_{I_{i-1}}^{I_i} |x - c_i|^2 f_X(x) dx$$

1. Optimal Levels $\{c_i\}$ Given Boundaries $\{I_i\}$

Objective: If the boundaries $\{I_i\}$ are fixed, what are the best $\{c_i\}$?

Notice that in the total cost C , only *one* integral depends on a specific c_i :

$$C_i := \int_{I_{i-1}}^{I_i} |x - c_i|^2 f_X(x) dx$$

Taking the derivative with respect to c_i and setting it to zero:

1. Optimal Levels $\{c_i\}$ Given Boundaries $\{I_i\}$

Objective: If the boundaries $\{I_i\}$ are fixed, what are the best $\{c_i\}$?

Notice that in the total cost C , only *one* integral depends on a specific c_i :

$$C_i := \int_{I_{i-1}}^{I_i} |x - c_i|^2 f_X(x) dx$$

Taking the derivative with respect to c_i and setting it to zero:

$$\frac{\partial C_i}{\partial c_i} = -2 \int_{I_{i-1}}^{I_i} (x - c_i) f_X(x) dx = 0$$

1. Optimal Levels $\{c_i\}$ Given Boundaries $\{I_i\}$

Objective: If the boundaries $\{I_i\}$ are fixed, what are the best $\{c_i\}$?

Notice that in the total cost C , only *one* integral depends on a specific c_i :

$$C_i := \int_{I_{i-1}}^{I_i} |x - c_i|^2 f_X(x) dx$$

Taking the derivative with respect to c_i and setting it to zero:

$$\frac{\partial C_i}{\partial c_i} = -2 \int_{I_{i-1}}^{I_i} (x - c_i) f_X(x) dx = 0$$

$$\implies \int_{I_{i-1}}^{I_i} x f_X(x) dx = c_i \int_{I_{i-1}}^{I_i} f_X(x) dx$$

1. Optimal Levels $\{c_i\}$ Given Boundaries $\{I_i\}$

Objective: If the boundaries $\{I_i\}$ are fixed, what are the best $\{c_i\}$?

Result: The Centroid Condition

$$c_i^* = \frac{\int_{I_{i-1}}^{I_i} x f_X(x) dx}{\int_{I_{i-1}}^{I_i} f_X(x) dx} = \mathbb{E}[x \mid x \in [I_{i-1}, I_i]]$$

2. Optimal Boundaries $\{I_i\}$ Given Levels $\{c_i\}$

Objective: If the levels $\{c_i\}$ are fixed, what are the best $\{I_i\}$?

2. Optimal Boundaries $\{I_i\}$ Given Levels $\{c_i\}$

Objective: If the levels $\{c_i\}$ are fixed, what are the best $\{I_i\}$? A specific boundary I_i only affects the two adjacent integrals in the sum:

$$C_{I_i} = \int_{I_{i-1}}^{I_i} |x - c_i|^2 f_X(x) dx + \int_{I_i}^{I_{i+1}} |x - c_{i+1}|^2 f_X(x) dx$$

2. Optimal Boundaries $\{I_i\}$ Given Levels $\{c_i\}$

Objective: If the levels $\{c_i\}$ are fixed, what are the best $\{I_i\}$? A specific boundary I_i only affects the two adjacent integrals in the sum:

$$C_{I_i} = \int_{I_{i-1}}^{I_i} |x - c_i|^2 f_X(x) dx + \int_{I_i}^{I_{i+1}} |x - c_{i+1}|^2 f_X(x) dx$$

Taking the derivative with respect to I_i (using Leibniz rule) and setting to zero:

2. Optimal Boundaries $\{I_i\}$ Given Levels $\{c_i\}$

Objective: If the levels $\{c_i\}$ are fixed, what are the best $\{I_i\}$? A specific boundary I_i only affects the two adjacent integrals in the sum:

$$C_{I_i} = \int_{I_{i-1}}^{I_i} |x - c_i|^2 f_X(x) dx + \int_{I_i}^{I_{i+1}} |x - c_{i+1}|^2 f_X(x) dx$$

Taking the derivative with respect to I_i (using Leibniz rule) and setting to zero:

$$|I_i - c_i|^2 f_X(I_i) - |I_i - c_{i+1}|^2 f_X(I_i) = 0$$

2. Optimal Boundaries $\{I_i\}$ Given Levels $\{c_i\}$

Objective: If the levels $\{c_i\}$ are fixed, what are the best $\{I_i\}$? A specific boundary I_i only affects the two adjacent integrals in the sum:

$$C_{I_i} = \int_{I_{i-1}}^{I_i} |x - c_i|^2 f_X(x) dx + \int_{I_i}^{I_{i+1}} |x - c_{i+1}|^2 f_X(x) dx$$

Taking the derivative with respect to I_i (using Leibniz rule) and setting to zero:

$$|I_i - c_i|^2 f_X(I_i) - |I_i - c_{i+1}|^2 f_X(I_i) = 0$$

$$\implies (I_i - c_i)^2 = (I_i - c_{i+1})^2$$

2. Optimal Boundaries $\{I_i\}$ Given Levels $\{c_i\}$

Objective: If the levels $\{c_i\}$ are fixed, what are the best $\{I_i\}$? A specific boundary I_i only affects the two adjacent integrals in the sum:

$$C_{I_i} = \int_{I_{i-1}}^{I_i} |x - c_i|^2 f_X(x) dx + \int_{I_i}^{I_{i+1}} |x - c_{i+1}|^2 f_X(x) dx$$

Taking the derivative with respect to I_i (using Leibniz rule) and setting to zero:

$$|I_i - c_i|^2 f_X(I_i) - |I_i - c_{i+1}|^2 f_X(I_i) = 0$$

$$\implies (I_i - c_i)^2 = (I_i - c_{i+1})^2$$

Result: The Nearest Neighbor Condition

$$I_i^* = \frac{c_i + c_{i+1}}{2}$$

The Lloyd–Max Algorithm

Because c_i^* depends on I_i , and I_i^* depends on c_i , we can solve this iteratively.

The Lloyd–Max Algorithm

Because c_i^* depends on I_i , and I_i^* depends on c_i , we can solve this iteratively.

Algorithm Steps

- 1 **Initialize** boundaries $\{I_i\}_{i=0}^{2^b}$ (e.g., uniformly over the range of x).

The Lloyd–Max Algorithm

Because c_i^* depends on I_i , and I_i^* depends on c_i , we can solve this iteratively.

Algorithm Steps

- 1 **Initialize** boundaries $\{I_i\}_{i=0}^{2^b}$ (e.g., uniformly over the range of x).
- 2 **Update levels** (Centroids):

$$c_i \leftarrow \mathbb{E}[x \mid x \in [I_{i-1}, I_i]], \quad \forall i$$

The Lloyd–Max Algorithm

Because c_i^* depends on I_i , and I_i^* depends on c_i , we can solve this iteratively.

Algorithm Steps

1 Initialize boundaries $\{I_i\}_{i=0}^{2^b}$ (e.g., uniformly over the range of x).

2 Update levels (Centroids):

$$c_i \leftarrow \mathbb{E}[x \mid x \in [I_{i-1}, I_i]], \quad \forall i$$

3 Update boundaries (Midpoints):

$$I_i \leftarrow \frac{c_i + c_{i+1}}{2}, \quad \forall i \in \{1, \dots, 2^b - 1\}$$

The Lloyd–Max Algorithm

Because c_i^* depends on I_i , and I_i^* depends on c_i , we can solve this iteratively.

Algorithm Steps

1 Initialize boundaries $\{I_i\}_{i=0}^{2^b}$ (e.g., uniformly over the range of x).

2 Update levels (Centroids):

$$c_i \leftarrow \mathbb{E}[x \mid x \in [I_{i-1}, I_i]], \quad \forall i$$

3 Update boundaries (Midpoints):

$$I_i \leftarrow \frac{c_i + c_{i+1}}{2}, \quad \forall i \in \{1, \dots, 2^b - 1\}$$

4 Repeat steps 2 and 3 until the decrease in cost C is below a threshold.

The Lloyd–Max Algorithm

Convergence Properties:

- It is a descent procedure; cost decreases at each iteration.

The Lloyd–Max Algorithm

Convergence Properties:

- It is a descent procedure; cost decreases at each iteration.
- Guaranteed to converge to a **local optimum**.

The Lloyd–Max Algorithm

Convergence Properties:

- It is a descent procedure; cost decreases at each iteration.
- Guaranteed to converge to a **local optimum**.
- Global optimality depends heavily on initialization and the shape of $f_X(x)$.

Moving Towards Vector Quantization

Moving Towards Vector Quantization

Objective: given $x \in \mathbb{R}^d$, design

$$Q : \mathbb{R}^d \rightarrow \{0, 1\}^{b \cdot d}, \quad Q^{-1} : \{0, 1\}^{b \cdot d} \rightarrow \mathbb{R}^d$$

$$\min \mathbb{E} \left[\|x - Q^{-1}(Q(x))\|_2^2 \right], \quad \tilde{x} := Q^{-1}(Q(x))$$

Moving Towards Vector Quantization

Objective: given $x \in \mathbb{R}^d$, design

$$Q : \mathbb{R}^d \rightarrow \{0, 1\}^{b \cdot d}, \quad Q^{-1} : \{0, 1\}^{b \cdot d} \rightarrow \mathbb{R}^d$$

$$\min \mathbb{E} \left[\|x - Q^{-1}(Q(x))\|_2^2 \right], \quad \tilde{x} := Q^{-1}(Q(x))$$

$$C := \mathbb{E} \left[\|x - Q^{-1}(Q(x))\|_2^2 \right] = \sum_{j=1}^d \mathbb{E} \left[|x_j - Q^{-1}(Q(x))_j|^2 \right]$$

Moving Towards Vector Quantization

Objective: given $x \in \mathbb{R}^d$, design

$$Q : \mathbb{R}^d \rightarrow \{0, 1\}^{b \cdot d}, \quad Q^{-1} : \{0, 1\}^{b \cdot d} \rightarrow \mathbb{R}^d$$

$$\min \mathbb{E} \left[\|x - Q^{-1}(Q(x))\|_2^2 \right], \quad \tilde{x} := Q^{-1}(Q(x))$$

$$C := \mathbb{E} \left[\|x - Q^{-1}(Q(x))\|_2^2 \right] = \sum_{j=1}^d \mathbb{E} \left[|x_j - Q^{-1}(Q(x))_j|^2 \right]$$

Assume: $x_j \stackrel{iid}{\sim} f_X$.

$$C = d \cdot \mathbb{E} \left[|x_j - Q^{-1}(Q(x))_j|^2 \right] = d \cdot \left(\sum_{i=1}^{2^b} \int_{I_{j-1}}^{I_j} |x_j - c_i|^2 f_X(x_j) dx_j \right)$$

scalar quantization, done d times

Our Aim

Our Aim: Quantize a *fixed* vector $x \in \mathbb{R}^d$.

Our Aim

Our Aim: Quantize a *fixed* vector $x \in \mathbb{R}^d$.

One caveat:

Our Aim

Our Aim: Quantize a *fixed* vector $x \in \mathbb{R}^d$.

One caveat:

- How to ensure random iid components?

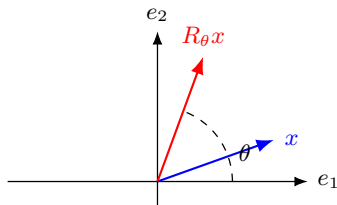
Rotation Matrices

Rotation Matrices

- Example: rotate a fixed $x \in \mathbb{R}^2$ by an angle θ .

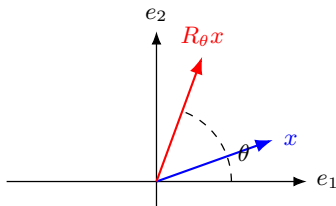
Rotation Matrices

- Example: rotate a fixed $x \in \mathbb{R}^2$ by an angle θ .



Rotation Matrices

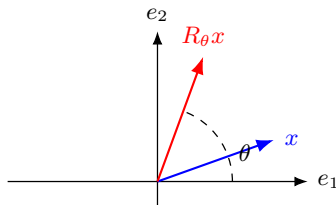
- Example: rotate a fixed $x \in \mathbb{R}^2$ by an angle θ .



$$R_\theta = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

Rotation Matrices

- Example: rotate a fixed $x \in \mathbb{R}^2$ by an angle θ .



$$R_\theta = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

Claim

For any such rotation, there exists a rotation matrix Π such that the rotation is given by $x \mapsto \Pi x$.

Orthogonal Matrices and $SO(d)$

Orthogonal Matrices and $SO(d)$

Define orthogonal matrices:

$$A \text{ is orthogonal} \iff AA^{\top} = A^{\top}A = I$$

$$\implies |\det A| = \pm 1$$

Orthogonal Matrices and $SO(d)$

Define orthogonal matrices:

$$A \text{ is orthogonal} \iff AA^\top = A^\top A = I$$

$$\implies |\det A| = \pm 1$$

Orthogonal group

$$O(d) = \{R \in \mathbb{R}^{d \times d} \mid R^\top R = I\}$$

Orthogonal Matrices and $SO(d)$

Define orthogonal matrices:

$$A \text{ is orthogonal} \iff AA^\top = A^\top A = I$$

$$\implies |\det A| = \pm 1$$

Orthogonal group

$$O(d) = \{R \in \mathbb{R}^{d \times d} \mid R^\top R = I\}$$

Special orthogonal group

$SO(d) \subset O(d)$ is the group of all **rotations** (excluding reflections, $\det R = +1$) in $\mathbb{R}^{d \times d}$ centred at the origin.

Lemma 1: Coordinate Distribution on the Hypersphere

Lemma 1 (coordinate distribution of a random point on the hypersphere)

Lemma 1: Coordinate Distribution on the Hypersphere

Lemma 1 (coordinate distribution of a random point on the hypersphere)

For any positive integer d , if $x \in \mathbb{S}^{d-1}$ is a random variable uniformly distributed over the unit hypersphere,

Lemma 1: Coordinate Distribution on the Hypersphere

Lemma 1 (coordinate distribution of a random point on the hypersphere)

For any positive integer d , if $x \in \mathbb{S}^{d-1}$ is a random variable uniformly distributed over the unit hypersphere, then for any $j \in [d]$ the coordinate x_j follows the following (scaled/shifted) Beta distribution:

$$x_j \sim f_X(x) := \frac{\Gamma(d/2)}{\sqrt{\pi} \cdot \Gamma((d-1)/2)} (1 - x^2)^{(d-3)/2}.$$

Lemma 1: Coordinate Distribution on the Hypersphere

Lemma 1 (coordinate distribution of a random point on the hypersphere)

For any positive integer d , if $x \in \mathbb{S}^{d-1}$ is a random variable uniformly distributed over the unit hypersphere, then for any $j \in [d]$ the coordinate x_j follows the following (scaled/shifted) Beta distribution:

$$x_j \sim f_X(x) := \frac{\Gamma(d/2)}{\sqrt{\pi} \cdot \Gamma((d-1)/2)} (1 - x^2)^{(d-3)/2}.$$

In high dimensions this Beta distribution converges to the normal distribution

$$f_X(\cdot) \rightarrow \mathcal{N}(0, 1/d).$$

Lemma 1: Coordinate Distribution on the Hypersphere

Lemma 1 (coordinate distribution of a random point on the hypersphere)

For any positive integer d , if $x \in \mathbb{S}^{d-1}$ is a random variable uniformly distributed over the unit hypersphere, then for any $j \in [d]$ the coordinate x_j follows the following (scaled/shifted) Beta distribution:

$$x_j \sim f_X(x) := \frac{\Gamma(d/2)}{\sqrt{\pi} \cdot \Gamma((d-1)/2)} (1 - x^2)^{(d-3)/2}.$$

In high dimensions this Beta distribution converges to the normal distribution

$$f_X(\cdot) \rightarrow \mathcal{N}(0, 1/d).$$

Also for sufficiently large d , x_j 's become "approximately" independent.

How to Get $x \sim \text{Unif}(\mathbb{S}^{d-1})$

How to Get $x \sim \text{Unif}(\mathbb{S}^{d-1})$

Let $\Pi \sim \text{Unif}(SO(d))$.

How to Get $x \sim \text{Unif}(\mathbb{S}^{d-1})$

Let $\Pi \sim \text{Unif}(SO(d))$.

Then, for any fixed $x \in \mathbb{R}^d$ with $\|x\| = 1$:

How to Get $x \sim \text{Unif}(\mathbb{S}^{d-1})$

Let $\Pi \sim \text{Unif}(SO(d))$.

Then, for any fixed $x \in \mathbb{R}^d$ with $\|x\| = 1$:

$$Y = \Pi x \sim \text{Unif}(\mathbb{S}^{d-1})$$

How to Get $x \sim \text{Unif}(\mathbb{S}^{d-1})$

Let $\Pi \sim \text{Unif}(SO(d))$.

Then, for any fixed $x \in \mathbb{R}^d$ with $\|x\| = 1$:

$$Y = \Pi x \sim \text{Unif}(\mathbb{S}^{d-1})$$

Intuition

How to Get $x \sim \text{Unif}(\mathbb{S}^{d-1})$

Let $\Pi \sim \text{Unif}(SO(d))$.

Then, for any fixed $x \in \mathbb{R}^d$ with $\|x\| = 1$:

$$Y = \Pi x \sim \text{Unif}(\mathbb{S}^{d-1})$$

Intuition

You rotate x by a uniformly random angle, so the resulting vector has no reason to be biased for a direction.

Generating $\Pi \sim \text{Unif}(SO(d))$

How to get $\Pi \sim \text{Unif}(SO(d))$:

Generating $\Pi \sim \text{Unif}(SO(d))$

How to get $\Pi \sim \text{Unif}(SO(d))$:

- 1 Generate a random matrix $A = [a_{ij}]$ such that $a_{ij} \stackrel{iid}{\sim} \mathcal{N}(0, 1)$.

Generating $\Pi \sim \text{Unif}(SO(d))$

How to get $\Pi \sim \text{Unif}(SO(d))$:

- 1 Generate a random matrix $A = [a_{ij}]$ such that $a_{ij} \stackrel{iid}{\sim} \mathcal{N}(0, 1)$.
- 2 Compute the **QR decomposition** of A :

$$A = \Pi R$$

where Π is orthogonal and R is upper triangular.

Generating $\Pi \sim \text{Unif}(SO(d))$

How to get $\Pi \sim \text{Unif}(SO(d))$:

- 1 Generate a random matrix $A = [a_{ij}]$ such that $a_{ij} \stackrel{iid}{\sim} \mathcal{N}(0, 1)$.
- 2 Compute the **QR decomposition** of A :

$$A = \Pi R$$

where Π is orthogonal and R is upper triangular.

- 3 Then it can be proven that $\Pi \sim \text{Unif}(SO(d))$.

Generating $\Pi \sim \text{Unif}(SO(d))$

How to get $\Pi \sim \text{Unif}(SO(d))$:

- 1 Generate a random matrix $A = [a_{ij}]$ such that $a_{ij} \stackrel{iid}{\sim} \mathcal{N}(0, 1)$.
- 2 Compute the **QR decomposition** of A :

$$A = \Pi R$$

where Π is orthogonal and R is upper triangular.

- 3 Then it can be proven that $\Pi \sim \text{Unif}(SO(d))$.

Look up Haar-measure if you are interested in a rigorous proof!

Algorithm 1 TURBOQUANT_{mse}: Optimized for MSE

- 1: **input:** dimension d and bit-width b
 - 2: Generate a random rotation matrix $\Pi \in \mathbb{R}^{d \times d}$
 - 3: Construct codebook with centroids $c_1, \dots, c_{2^b} \in [-1, 1]$ minimizing MSE
-

Algorithm 2 TURBOQUANT_{mse}: Optimized for MSE

- 1: **input:** dimension d and bit-width b
- 2: Generate a **random rotation matrix** $\Pi \in \mathbb{R}^{d \times d}$
- 3: Construct **codebook** with centroids $c_1, \dots, c_{2^b} \in [-1, 1]$ minimizing MSE
- 4: **procedure** QUANT_{MSE}(x)
- 5: $y \leftarrow \Pi \cdot x$
- 6: $\text{idx}_j \leftarrow \arg \min_k |y_j - c_k|$ for $j \in [d]$
- 7: **output:** idx
- 8: **end procedure**

TurboQuant_{mse} Algorithm

Algorithm 3 TURBOQUANT_{mse}: Optimized for MSE

- 1: **input:** dimension d and bit-width b
 - 2: Generate a **random rotation matrix** $\Pi \in \mathbb{R}^{d \times d}$
 - 3: Construct **codebook** with centroids $c_1, \dots, c_{2^b} \in [-1, 1]$ minimizing MSE
 - 4: **procedure** QUANT_{MSE}(x)
 - 5: $y \leftarrow \Pi \cdot x$
 - 6: $\text{idx}_j \leftarrow \arg \min_k |y_j - c_k|$ for $j \in [d]$
 - 7: **output:** idx
 - 8: **end procedure**
 - 9: **procedure** DEQUANT_{MSE}(idx)
 - 10: $\tilde{y}_j \leftarrow c_{\text{idx}_j}$ for every $j \in [d]$
 - 11: $\tilde{x} \leftarrow \Pi^\top \cdot \tilde{y}$
 - 12: **output:** $\tilde{x}$
 - 13: **end procedure**
-

Performance guarantee

Lemma (performance guarantee: $\text{TURBOQUANT}_{\text{mse}}$)

For any bit-width $b \geq 1$ and any vector $x \in \mathbb{S}^{d-1}$, the procedure $\text{Quant}_{\text{mse}}(x)$ in Algorithm 1 returns an index vector $\text{idx} \in [2^b]^d$. When this index vector is provided to the primitive $\text{DeQuant}_{\text{mse}}(\text{idx})$, a reconstructed vector $\tilde{x} \in \mathbb{R}^d$ is obtained that satisfies the following distortion guarantee. Specifically, the mean-squared error (MSE), defined as

$$D_{\text{mse}} := \mathbb{E}_{\tilde{x}} [\|x - \tilde{x}\|_2^2],$$

obeys the bound

$$D_{\text{mse}} \leq \frac{\sqrt{3}\pi}{2} \cdot \frac{1}{4^b},$$

Theorem: Lower bound on best achievable compression distortion

Lower bound

For any randomized quantization algorithm $Q : \mathbb{S}^{d-1} \rightarrow \{0, 1\}^{b \cdot d}$ with bit-width b and any reconstruction map $Q^{-1} : \{0, 1\}^{b \cdot d} \rightarrow \mathbb{R}^d$, there exists a hard input instance $x \in \mathbb{S}^{d-1}$ such that:

$$D_{\text{ms}}(Q) := \mathbb{E} \left[\|x - Q^{-1}(Q(x))\|_2^2 \right] \geq \frac{1}{4^b}.$$

TurboQuant_{mse}: MSE Upper Bound

Goal: bound $D_{\text{mse}} := \mathbb{E}[\|x - \tilde{x}\|_2^2]$.

Proof.

Let $\tilde{y}$ be the dequantized value.

Since Π is a rotation matrix, $\|x - \tilde{x}\|_2 = \|\Pi x - \Pi \tilde{x}\|_2$. Using $y = \Pi x$:

TurboQuant_{mse}: MSE Upper Bound

Goal: bound $D_{\text{mse}} := \mathbb{E}[\|x - \tilde{x}\|_2^2]$.

Proof.

Let $\tilde{y}$ be the dequantized value.

Since Π is a rotation matrix, $\|x - \tilde{x}\|_2 = \|\Pi x - \Pi \tilde{x}\|_2$. Using $y = \Pi x$:

$$D_{\text{mse}} = \mathbb{E}[\|y - \tilde{y}\|_2^2] = \sum_{j \in [d]} \mathbb{E}[|y_j - \tilde{y}_j|^2] = \sum_{j \in [d]} \mathbb{E}[|y_j - c_{\text{idx}_j}|^2]$$

TurboQuant_{mse}: MSE Upper Bound

Goal: bound $D_{\text{mse}} := \mathbb{E}[\|x - \tilde{x}\|_2^2]$.

Proof.

Let $\tilde{y}$ be the dequantized value.

Since Π is a rotation matrix, $\|x - \tilde{x}\|_2 = \|\Pi x - \Pi \tilde{x}\|_2$. Using $y = \Pi x$:

$$\begin{aligned} D_{\text{mse}} &= \mathbb{E}[\|y - \tilde{y}\|_2^2] = \sum_{j \in [d]} \mathbb{E}[|y_j - \tilde{y}_j|^2] = \sum_{j \in [d]} \mathbb{E}[|y_j - c_{\text{id}x_j}|^2] \\ &= d \cdot \mathbb{E}[|y_1 - c_{\text{id}x_1}|^2] = d \cdot \min_{-1 \leq c_1 \leq \dots \leq c_{2^b} \leq 1} \sum_{i=1}^{2^b} \int_{\frac{c_{i-1} + c_i}{2}}^{\frac{c_i + c_{i+1}}{2}} |x - c_i|^2 f_X(x) dx \end{aligned}$$

TurboQuant_{mse}: MSE Upper Bound

Goal: bound $D_{\text{mse}} := \mathbb{E}[\|x - \tilde{x}\|_2^2]$.

Proof.

Let $\tilde{y}$ be the dequantized value.

Since Π is a rotation matrix, $\|x - \tilde{x}\|_2 = \|\Pi x - \Pi \tilde{x}\|_2$. Using $y = \Pi x$:

$$\begin{aligned} D_{\text{mse}} &= \mathbb{E}[\|y - \tilde{y}\|_2^2] = \sum_{j \in [d]} \mathbb{E}[|y_j - \tilde{y}_j|^2] = \sum_{j \in [d]} \mathbb{E}[|y_j - c_{\text{id}x_j}|^2] \\ &= d \cdot \mathbb{E}[|y_1 - c_{\text{id}x_1}|^2] = d \cdot \min_{-1 \leq c_1 \leq \dots \leq c_{2^b} \leq 1} \sum_{i=1}^{2^b} \int_{\frac{c_{i-1} + c_i}{2}}^{\frac{c_i + c_{i+1}}{2}} |x - c_i|^2 f_X(x) dx \\ &= d \cdot C(f_X, b) \end{aligned}$$

TurboQuant_{mse}: MSE Upper Bound

Goal: bound $D_{\text{mse}} := \mathbb{E}[\|x - \tilde{x}\|_2^2]$.

Proof.

Let $\tilde{y}$ be the dequantized value.

Since Π is a rotation matrix, $\|x - \tilde{x}\|_2 = \|\Pi x - \Pi \tilde{x}\|_2$. Using $y = \Pi x$:

$$\begin{aligned} D_{\text{mse}} &= \mathbb{E}[\|y - \tilde{y}\|_2^2] = \sum_{j \in [d]} \mathbb{E}[|y_j - \tilde{y}_j|^2] = \sum_{j \in [d]} \mathbb{E}[|y_j - c_{\text{id}_{X_j}}|^2] \\ &= d \cdot \mathbb{E}[|y_1 - c_{\text{id}_{X_1}}|^2] = d \cdot \min_{-1 \leq c_1 \leq \dots \leq c_{2^b} \leq 1} \sum_{i=1}^{2^b} \int_{\frac{c_{i-1} + c_i}{2}}^{\frac{c_i + c_{i+1}}{2}} |x - c_i|^2 f_X(x) dx \\ &= d \cdot C(f_X, b) \\ &= d \cdot C(f_X, b) \leq d \cdot \left(\frac{\sqrt{3}\pi}{2d} \cdot \frac{1}{4^b} \right) \end{aligned}$$

TurboQuant_{mse}: MSE Upper Bound — Closed Form

Now we bound the optimal cost $C(f_X, b)$.

For moderate d , $f_X \rightarrow \mathcal{N}(0, 1/d)$.

Numerically solving Eq. (4) for $b = 1, 2, 3, 4$:

$$C(f_X, b) \approx \frac{0.36}{d}, \frac{0.117}{d}, \frac{0.03}{d}, \frac{0.009}{d}, \quad \text{respectively.}$$

For larger bit-widths $b > 4$, apply the **Panter–Dite** high-resolution formula for the distortion of a fixed-rate scalar quantizer:

$$C(f_X, b) \leq \frac{1}{12} \cdot \left(\int f_X(x)^{1/3} dx \right)^3 \cdot \frac{1}{4^b} = \frac{\sqrt{3}\pi}{2d} \cdot \frac{1}{4^b}.$$



Is MSE enough?

Is MSE enough?

→ Which is the most important vector operation today?

Is MSE enough?

→ Which is the most important vector operation today?

DOT PRODUCT!

Is MSE enough?

→ Which is the most important vector operation today?

DOT PRODUCT!

→ What if we are able to preserve dot products?

Is MSE enough?

→ Which is the most important vector operation today?

DOT PRODUCT!

→ What if we are able to preserve dot products?

→ The following are two important metrics we care about:

Inner product distortion

$$D_{\text{inner product}} := \mathbb{E} \left[\left| \langle y, x \rangle - \langle y, \tilde{x} \rangle \right|^2 \right]$$

Is MSE enough?

→ Which is the most important vector operation today?

DOT PRODUCT!

→ What if we are able to preserve dot products?

→ The following are two important metrics we care about:

Inner product distortion

$$D_{\text{inner product}} := \mathbb{E} \left[\left| \langle y, x \rangle - \langle y, \tilde{x} \rangle \right|^2 \right]$$

Unbiasedness of Inner product estimate

$$\mathbb{E} [\langle y, \tilde{x} \rangle] = \langle y, x \rangle$$

What if We Use $\text{TurboQuant}_{\text{mse}}$?

Consider $b = 1$, large d :

$$\mathbb{E}[\langle y, \tilde{x}_{\text{mse}} \rangle] = \frac{2}{\pi} \langle y, x \rangle \quad \textbf{Biased!}$$

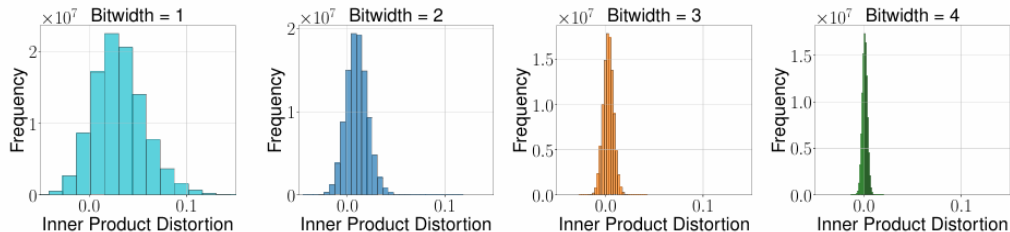


Figure: $\text{TurboQuant}_{\text{mse}}$ bias decreases with b for inner product estimate

What if We Use TurboQuant_{mse} ?

Consider $b = 1$, large d :

$$\mathbb{E}[\langle y, \tilde{x}_{mse} \rangle] = \frac{2}{\pi} \langle y, x \rangle \quad \text{Biased!}$$

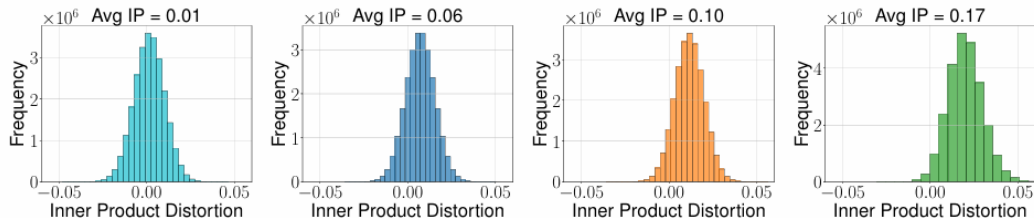


Figure: Inner product estimate bias depends on avg value in $TurbQuant_{mse}$, $b = 2$

Theorem: Lower bound on best achievable compression distortion

Lower bound

For any randomized quantization algorithm $Q : \mathbb{S}^{d-1} \rightarrow \{0, 1\}^{b \cdot d}$ with bit-width b and any reconstruction map $Q^{-1} : \{0, 1\}^{b \cdot d} \rightarrow \mathbb{R}^d$, there exists a hard input instance $x \in \mathbb{S}^{d-1}$ such that:

$$D_{\text{ms}}(Q) := \mathbb{E} \left[\|x - Q^{-1}(Q(x))\|_2^2 \right] \geq \frac{1}{4^b}.$$

Theorem: Lower bound on best achievable compression distortion

Lower bound

For any randomized quantization algorithm $Q : \mathbb{S}^{d-1} \rightarrow \{0, 1\}^{b \cdot d}$ with bit-width b and any reconstruction map $Q^{-1} : \{0, 1\}^{b \cdot d} \rightarrow \mathbb{R}^d$, there exists a hard input instance $x \in \mathbb{S}^{d-1}$ such that:

$$D_{\text{ms}}(Q) := \mathbb{E} \left[\|x - Q^{-1}(Q(x))\|_2^2 \right] \geq \frac{1}{4^b}.$$

Furthermore, there exists a $y \in \mathbb{S}^{d-1}$ such that:

$$D_{\text{prod}}(Q) = \mathbb{E} \left[|\langle y, x \rangle - \langle y, Q^{-1}(Q(x)) \rangle|^2 \right] \geq \frac{1}{d} \cdot \frac{1}{4^b}.$$

The 1-bit Quantizer: QJL

The 1-bit Quantizer: QJL

Definition 1 (QJL)

For any positive integer d the QJL map $Q_{\text{qjl}} : \mathbb{R}^d \rightarrow \{-1, +1\}^d$ is defined as:

The 1-bit Quantizer: QJL

Definition 1 (QJL)

For any positive integer d the QJL map $Q_{\text{qjl}} : \mathbb{R}^d \rightarrow \{-1, +1\}^d$ is defined as:

$$Q_{\text{qjl}}(\mathbf{x}) := \text{sgn}(\mathbf{S} \cdot \mathbf{x}) \quad \text{for any } \mathbf{x} \in \mathbb{R}^d,$$

The 1-bit Quantizer: QJL

Definition 1 (QJL)

For any positive integer d the QJL map $Q_{\text{qjl}} : \mathbb{R}^d \rightarrow \{-1, +1\}^d$ is defined as:

$$Q_{\text{qjl}}(\mathbf{x}) := \text{sgn}(\mathbf{S} \cdot \mathbf{x}) \quad \text{for any } \mathbf{x} \in \mathbb{R}^d,$$

where $\mathbf{S} \in \mathbb{R}^{d \times d}$ is a random matrix with i.i.d. entries sampled from the normal distribution $\mathcal{N}(0, 1)$ and the sgn function is applied entry-wise to its vector input.

The 1-bit Quantizer: QJL

Definition 1 (QJL)

For any positive integer d the QJL map $Q_{\text{qjl}} : \mathbb{R}^d \rightarrow \{-1, +1\}^d$ is defined as:

$$Q_{\text{qjl}}(\mathbf{x}) := \text{sgn}(\mathbf{S} \cdot \mathbf{x}) \quad \text{for any } \mathbf{x} \in \mathbb{R}^d,$$

where $\mathbf{S} \in \mathbb{R}^{d \times d}$ is a random matrix with i.i.d. entries sampled from the normal distribution $\mathcal{N}(0, 1)$ and the sgn function is applied entry-wise to its vector input.

The dequantization map $Q_{\text{qjl}}^{-1} : \{-1, +1\}^d \rightarrow \mathbb{R}^d$ is defined as:

The 1-bit Quantizer: QJL

Definition 1 (QJL)

For any positive integer d the QJL map $Q_{\text{qjl}} : \mathbb{R}^d \rightarrow \{-1, +1\}^d$ is defined as:

$$Q_{\text{qjl}}(\mathbf{x}) := \text{sgn}(\mathbf{S} \cdot \mathbf{x}) \quad \text{for any } \mathbf{x} \in \mathbb{R}^d,$$

where $\mathbf{S} \in \mathbb{R}^{d \times d}$ is a random matrix with i.i.d. entries sampled from the normal distribution $\mathcal{N}(0, 1)$ and the sgn function is applied entry-wise to its vector input.

The dequantization map $Q_{\text{qjl}}^{-1} : \{-1, +1\}^d \rightarrow \mathbb{R}^d$ is defined as:

$$Q_{\text{qjl}}^{-1}(\mathbf{z}) := \frac{\sqrt{\pi/2}}{d} \cdot \mathbf{S}^\top \cdot \mathbf{z} \quad \text{for any } \mathbf{z} \in \{-1, +1\}^d.$$

Performance Guarantee of QJL

Performance Guarantee of QJL

Lemma 4 (performance guarantee: QJL)

Let Q_{qjl} and Q_{qjl}^{-1} be defined as per Definition 1.

Performance Guarantee of QJL

Lemma 4 (performance guarantee: QJL)

Let Q_{qjl} and Q_{qjl}^{-1} be defined as per Definition 1. For any vector $\mathbf{x} \in \mathbb{S}^{d-1}$ and any $\mathbf{y} \in \mathbb{R}^d$ we have the following:

Performance Guarantee of QJL

Lemma 4 (performance guarantee: QJL)

Let Q_{qjl} and Q_{qjl}^{-1} be defined as per Definition 1. For any vector $\mathbf{x} \in \mathbb{S}^{d-1}$ and any $\mathbf{y} \in \mathbb{R}^d$ we have the following:

■ *Unbiased:*

$$\mathbb{E} \left[\langle \mathbf{y}, Q_{\text{qjl}}^{-1}(Q_{\text{qjl}}(\mathbf{x})) \rangle \right] = \langle \mathbf{y}, \mathbf{x} \rangle.$$

Performance Guarantee of QJL

Lemma 4 (performance guarantee: QJL)

Let Q_{qjl} and Q_{qjl}^{-1} be defined as per Definition 1. For any vector $\mathbf{x} \in \mathbb{S}^{d-1}$ and any $\mathbf{y} \in \mathbb{R}^d$ we have the following:

■ *Unbiased:*

$$\mathbb{E}\left[\langle \mathbf{y}, Q_{\text{qjl}}^{-1}(Q_{\text{qjl}}(\mathbf{x})) \rangle\right] = \langle \mathbf{y}, \mathbf{x} \rangle.$$

■ *Variance Bound:*

$$\text{Var}\left(\langle \mathbf{y}, Q_{\text{qjl}}^{-1}(Q_{\text{qjl}}(\mathbf{x})) \rangle\right) \leq \frac{\pi}{2d} \cdot \|\mathbf{y}\|_2^2$$

Combining the Two

Combining the Two

Let Q_{mse} be $\text{TURBOQUANT}_{\text{mse}}$ with $b - 1$ bits.

Let $\tilde{x}_{\text{mse}} = Q_{\text{mse}}^{-1}(Q_{\text{mse}}(x))$.

Combining the Two

Let Q_{mse} be $\text{TURBOQUANT}_{\text{mse}}$ with $b - 1$ bits.

Let $\tilde{x}_{\text{mse}} = Q_{\text{mse}}^{-1}(Q_{\text{mse}}(x))$. **Define** the residual vector

$$r = x - \tilde{x}_{\text{mse}}, \quad \|r\|_2 = \|x - \tilde{x}_{\text{mse}}\|_2.$$

Combining the Two

Let Q_{mse} be $\text{TURBOQUANT}_{\text{mse}}$ with $b - 1$ bits.

Let $\tilde{x}_{\text{mse}} = Q_{\text{mse}}^{-1}(Q_{\text{mse}}(x))$. **Define** the residual vector

$$r = x - \tilde{x}_{\text{mse}}, \quad \|r\|_2 = \|x - \tilde{x}_{\text{mse}}\|_2.$$

QJL-quantize the (unit-norm) residual direction $\frac{r}{\|r\|_2}$. Let $\tilde{r}_{qjl} = Q_{qjl}^{-1}(Q_{qjl}(\frac{r}{\|r\|_2}))$

Combining the Two

Let Q_{mse} be $\text{TURBOQUANT}_{\text{mse}}$ with $b - 1$ bits.

Let $\tilde{x}_{\text{mse}} = Q_{\text{mse}}^{-1}(Q_{\text{mse}}(x))$. **Define** the residual vector

$$r = x - \tilde{x}_{\text{mse}}, \quad \|r\|_2 = \|x - \tilde{x}_{\text{mse}}\|_2.$$

QJL-quantize the (unit-norm) residual direction $\frac{r}{\|r\|_2}$. Let $\tilde{r}_{qjl} = Q_{qjl}^{-1}(Q_{qjl}(\frac{r}{\|r\|_2}))$

Quantization

$$Q_{\text{prod}}(x) = \left[Q_{\text{mse}}(x), Q_{\text{qjl}}(x - \tilde{x}_{\text{mse}}), \|x - \tilde{x}_{\text{mse}}\|_2 \right]$$

Combining the Two

Let Q_{mse} be $\text{TURBOQUANT}_{\text{mse}}$ with $b - 1$ bits.

Let $\tilde{x}_{\text{mse}} = Q_{\text{mse}}^{-1}(Q_{\text{mse}}(x))$. **Define** the residual vector

$$r = x - \tilde{x}_{\text{mse}}, \quad \|r\|_2 = \|x - \tilde{x}_{\text{mse}}\|_2.$$

QJL-quantize the (unit-norm) residual direction $\frac{r}{\|r\|_2}$. Let $\tilde{r}_{\text{qjl}} = Q_{\text{qjl}}^{-1}(Q_{\text{qjl}}(\frac{r}{\|r\|_2}))$

Quantization

$$Q_{\text{prod}}(x) = \left[Q_{\text{mse}}(x), Q_{\text{qjl}}(x - \tilde{x}_{\text{mse}}), \|x - \tilde{x}_{\text{mse}}\|_2 \right]$$

Dequantization

$$\tilde{x}_{\text{prod}} = \tilde{x}_{\text{mse}} + \tilde{x}_{\text{qjl}} = \tilde{x}_{\text{mse}} + \|r\|_2 \cdot \tilde{r}_{\text{qjl}}$$

TurboQuant_{prod} Algorithm

Algorithm 2 TURBOQUANT_{prod}: optimized for inner product

- 1: **input:** dimension d and bit-width b
 - 2: Instantiate a TURBOQUANT_{mse} with bit-width $b - 1$ as per Algorithm 1
 - 3: Generate a **random projection matrix** $\mathbf{S} \in \mathbb{R}^{d \times d}$ with i.i.d. entries $S_{i,j} \sim \mathcal{N}(0, 1)$
-

TurboQuant_{prod} Algorithm

Algorithm 2 TURBOQUANT_{prod}: optimized for inner product

- 1: **input:** dimension d and bit-width b
- 2: Instantiate a TURBOQUANT_{mse} with bit-width $b - 1$ as per Algorithm 1
- 3: Generate a **random projection matrix** $\mathbf{S} \in \mathbb{R}^{d \times d}$ with i.i.d. entries $S_{i,j} \sim \mathcal{N}(0, 1)$
- 4: **procedure** QUANT_{PROD}($\mathbf{x}$)
- 5: $\text{idx} \leftarrow \text{QUANT}_{\text{mse}}(\mathbf{x})$
- 6: $\mathbf{r} \leftarrow \mathbf{x} - \text{DEQUANT}_{\text{mse}}(\text{idx})$ ▷ residual vector
- 7: $\text{qjl} \leftarrow \text{sgn}(\mathbf{S} \cdot \mathbf{r})$ ▷ QJL on residual vector
- 8: **output:** ($\text{idx}, \text{qjl}, \|\mathbf{r}\|_2$)
- 9: **end procedure**

TurboQuant_{prod} Algorithm

Algorithm 2 TURBOQUANT_{prod}: optimized for inner product

-
- 1: **input:** dimension d and bit-width b
 - 2: Instantiate a TURBOQUANT_{mse} with bit-width $b - 1$ as per Algorithm 1
 - 3: Generate a **random projection matrix** $\mathbf{S} \in \mathbb{R}^{d \times d}$ with i.i.d. entries $S_{i,j} \sim \mathcal{N}(0, 1)$
-
- 4: **procedure** QUANT_{PROD}($\mathbf{x}$)
 - 5: $\text{idx} \leftarrow \text{QUANT}_{\text{mse}}(\mathbf{x})$
 - 6: $\mathbf{r} \leftarrow \mathbf{x} - \text{DEQUANT}_{\text{mse}}(\text{idx})$ ▷ residual vector
 - 7: $\text{qjl} \leftarrow \text{sgn}(\mathbf{S} \cdot \mathbf{r})$ ▷ QJL on residual vector
 - 8: **output:** ($\text{idx}, \text{qjl}, \|\mathbf{r}\|_2$)
 - 9: **end procedure**
-
- 10: **procedure** DEQUANT_{PROD}($\text{idx}, \text{qjl}, \gamma$)
 - 11: $\tilde{\mathbf{x}}_{\text{mse}} \leftarrow \text{DEQUANT}_{\text{mse}}(\text{idx})$
 - 12: $\tilde{\mathbf{x}}_{\text{qjl}} \leftarrow \frac{\sqrt{\pi/2}}{d} \cdot \gamma \cdot \mathbf{S}^\top \cdot \text{qjl}$
 - 13: **output:** $\tilde{\mathbf{x}}_{\text{mse}} + \tilde{\mathbf{x}}_{\text{qjl}}$
 - 14: **end procedure**
-

Performance Guarantee of TurboQuant_{prod}

Performance Guarantee of TurboQuant_{prod}

Theorem 2 (performance guarantee: TURBOQUANT_{prod})

For any bit-width $b \geq 1$ and any vector $x \in \mathbb{S}^{d-1}$, the procedure QUANT_{prod}(x) in Algorithm 2 outputs an index vector $\text{idx} \in [2^{b-1}]^d$ along with a sign vector $\text{qjl} \in \{-1, 1\}^d$ and a positive number $\gamma \geq 0$. When passed to DEQUANT_{prod}($\text{idx}, \text{qjl}, \gamma$), it produces a reconstructed vector $\tilde{x} \in \mathbb{R}^d$ that for any vector $y \in \mathbb{R}^d$ satisfies:

- *Expected inner-product:* $\mathbb{E}_{\tilde{x}} [\langle y, \tilde{x} \rangle] = \langle y, x \rangle$

Performance Guarantee of TurboQuant_{prod}

Theorem 2 (performance guarantee: TURBOQUANT_{prod})

For any bit-width $b \geq 1$ and any vector $x \in \mathbb{S}^{d-1}$, the procedure QUANT_{prod}(x) in Algorithm 2 outputs an index vector $\text{idx} \in [2^{b-1}]^d$ along with a sign vector $\text{qjl} \in \{-1, 1\}^d$ and a positive number $\gamma \geq 0$. When passed to DEQUANT_{prod}($\text{idx}, \text{qjl}, \gamma$), it produces a reconstructed vector $\tilde{x} \in \mathbb{R}^d$ that for any vector $y \in \mathbb{R}^d$ satisfies:

- *Expected inner-product:* $\mathbb{E}_{\tilde{x}} [\langle y, \tilde{x} \rangle] = \langle y, x \rangle$
- *Inner-product distortion* defined as $D_{\text{prod}} := \mathbb{E}_{\tilde{x}} \left[|\langle y, x \rangle - \langle y, \tilde{x} \rangle|^2 \right]$ is bounded by

$$D_{\text{prod}} \leq \frac{\sqrt{3}\pi^2 \cdot \|y\|_2^2}{d} \cdot \frac{1}{4^b} \quad \text{for any } b \geq 0.$$

Proof of Unbiasedness

Proof of Unbiasedness

$$\mathbb{E}[\langle y, \tilde{x}_{prod} \rangle \mid \tilde{x}_{mse}] = \mathbb{E}_{\tilde{x}_{qjl}} [\langle y, \tilde{x}_{mse} + \tilde{x}_{qjl} \rangle \mid \tilde{x}_{mse}]$$

Proof of Unbiasedness

$$\begin{aligned}\mathbb{E}[\langle y, \tilde{x}_{prod} \rangle \mid \tilde{x}_{mse}] &= \mathbb{E}_{\tilde{x}_{qjl}} [\langle y, \tilde{x}_{mse} + \tilde{x}_{qjl} \rangle \mid \tilde{x}_{mse}] \\ &= \langle y, \tilde{x}_{mse} \rangle + \mathbb{E}_{\tilde{x}_{qjl}} [\langle y, \tilde{x}_{qjl} \rangle \mid \tilde{x}_{mse}]\end{aligned}$$

Proof of Unbiasedness

$$\begin{aligned}\mathbb{E}[\langle y, \tilde{x}_{prod} \rangle \mid \tilde{x}_{mse}] &= \mathbb{E}_{\tilde{x}_{qjl}} [\langle y, \tilde{x}_{mse} + \tilde{x}_{qjl} \rangle \mid \tilde{x}_{mse}] \\ &= \langle y, \tilde{x}_{mse} \rangle + \mathbb{E}_{\tilde{x}_{qjl}} [\langle y, \tilde{x}_{qjl} \rangle \mid \tilde{x}_{mse}] \\ &= \langle y, \tilde{x}_{mse} \rangle + \langle y, r \rangle\end{aligned}$$

Proof of Unbiasedness

$$\begin{aligned}\mathbb{E}[\langle y, \tilde{x}_{prod} \rangle \mid \tilde{x}_{mse}] &= \mathbb{E}_{\tilde{x}_{qjl}}[\langle y, \tilde{x}_{mse} + \tilde{x}_{qjl} \rangle \mid \tilde{x}_{mse}] \\&= \langle y, \tilde{x}_{mse} \rangle + \mathbb{E}_{\tilde{x}_{qjl}}[\langle y, \tilde{x}_{qjl} \rangle \mid \tilde{x}_{mse}] \\&= \langle y, \tilde{x}_{mse} \rangle + \langle y, r \rangle \\&= \langle y, x \rangle,\end{aligned}$$

Proof of Unbiasedness

$$\begin{aligned}\mathbb{E}[\langle y, \tilde{x}_{prod} \rangle \mid \tilde{x}_{mse}] &= \mathbb{E}_{\tilde{x}_{qjl}}[\langle y, \tilde{x}_{mse} + \tilde{x}_{qjl} \rangle \mid \tilde{x}_{mse}] \\&= \langle y, \tilde{x}_{mse} \rangle + \mathbb{E}_{\tilde{x}_{qjl}}[\langle y, \tilde{x}_{qjl} \rangle \mid \tilde{x}_{mse}] \\&= \langle y, \tilde{x}_{mse} \rangle + \langle y, r \rangle \\&= \langle y, x \rangle,\end{aligned}$$

Now, using the **law of total expectation**:

$$\mathbb{E}_{\tilde{x}}[\langle y, \tilde{x} \rangle] = \mathbb{E}_{\tilde{x}_{mse}}[\mathbb{E}[\langle y, \tilde{x} \rangle \mid \tilde{x}_{mse}]] = \mathbb{E}[\langle y, x \rangle] = \langle y, x \rangle,$$

which proves the first claim of the theorem.

Upper Bound Proof

Upper Bound Proof

We apply the same conditioning on $\tilde{x}_{\text{mse}}$ when computing the distortion:

Upper Bound Proof

We apply the same conditioning on $\tilde{x}_{\text{mse}}$ when computing the distortion:

$$\mathbb{E} \left[\left| \langle y, x \rangle - \langle y, \tilde{x}_{\text{prod}} \rangle \right|^2 \middle| \tilde{x}_{\text{mse}} \right] = \mathbb{E}_{\tilde{x}_{\text{qjl}}} \left[\left| \langle y, x \rangle - \langle y, \tilde{x}_{\text{mse}} + \tilde{x}_{\text{qjl}} \rangle \right|^2 \middle| \tilde{x}_{\text{mse}} \right]$$

Upper Bound Proof

We apply the same conditioning on $\tilde{x}_{\text{mse}}$ when computing the distortion:

$$\begin{aligned}\mathbb{E}\left[\left|\langle y, x \rangle - \langle y, \tilde{x}_{\text{prod}} \rangle\right|^2 \middle| \tilde{x}_{\text{mse}}\right] &= \mathbb{E}_{\tilde{x}_{\text{qjl}}}\left[\left|\langle y, x \rangle - \langle y, \tilde{x}_{\text{mse}} + \tilde{x}_{\text{qjl}} \rangle\right|^2 \middle| \tilde{x}_{\text{mse}}\right] \\ &= \mathbb{E}_{\tilde{x}_{\text{qjl}}}\left[\left|\langle y, r \rangle - \langle y, \tilde{x}_{\text{qjl}} \rangle\right|^2 \middle| \tilde{x}_{\text{mse}}\right]\end{aligned}$$

Upper Bound Proof

We apply the same conditioning on $\tilde{x}_{\text{mse}}$ when computing the distortion:

$$\begin{aligned}\mathbb{E}\left[\left|\langle y, x \rangle - \langle y, \tilde{x}_{\text{prod}} \rangle\right|^2 \middle| \tilde{x}_{\text{mse}}\right] &= \mathbb{E}_{\tilde{x}_{\text{qjl}}}\left[\left|\langle y, x \rangle - \langle y, \tilde{x}_{\text{mse}} + \tilde{x}_{\text{qjl}} \rangle\right|^2 \middle| \tilde{x}_{\text{mse}}\right] \\&= \mathbb{E}_{\tilde{x}_{\text{qjl}}}\left[\left|\langle y, r \rangle - \langle y, \tilde{x}_{\text{qjl}} \rangle\right|^2 \middle| \tilde{x}_{\text{mse}}\right] \\&= \text{Var}(\langle y, \tilde{x}_{\text{qjl}} \rangle \mid \tilde{x}_{\text{mse}}) \leq \frac{\pi}{2d} \cdot \|r\|_2^2 \|y\|_2^2,\end{aligned}$$

By the **law of total expectation**:

$$D_{\text{prod}} = \mathbb{E}_{\tilde{x}_{\text{mse}}}\left[\mathbb{E}\left[\left|\langle y, x \rangle - \langle y, \tilde{x} \rangle\right|^2 \middle| \tilde{x}_{\text{mse}}\right]\right]$$

Upper Bound Proof

We apply the same conditioning on $\tilde{x}_{\text{mse}}$ when computing the distortion:

$$\begin{aligned}\mathbb{E}\left[|\langle y, x \rangle - \langle y, \tilde{x}_{\text{prod}} \rangle|^2 \mid \tilde{x}_{\text{mse}}\right] &= \mathbb{E}_{\tilde{x}_{\text{qjl}}}\left[|\langle y, x \rangle - \langle y, \tilde{x}_{\text{mse}} + \tilde{x}_{\text{qjl}} \rangle|^2 \mid \tilde{x}_{\text{mse}}\right] \\&= \mathbb{E}_{\tilde{x}_{\text{qjl}}}\left[|\langle y, r \rangle - \langle y, \tilde{x}_{\text{qjl}} \rangle|^2 \mid \tilde{x}_{\text{mse}}\right] \\&= \text{Var}(\langle y, \tilde{x}_{\text{qjl}} \rangle \mid \tilde{x}_{\text{mse}}) \leq \frac{\pi}{2d} \cdot \|r\|_2^2 \|y\|_2^2,\end{aligned}$$

By the **law of total expectation**:

$$\begin{aligned}D_{\text{prod}} &= \mathbb{E}_{\tilde{x}_{\text{mse}}}\left[\mathbb{E}\left[|\langle y, x \rangle - \langle y, \tilde{x} \rangle|^2 \mid \tilde{x}_{\text{mse}}\right]\right] \\&\leq \frac{\pi}{2d} \cdot \|y\|_2^2 \cdot \mathbb{E}[\|x - \tilde{x}_{\text{mse}}\|_2^2] = \frac{\pi}{2d} \cdot \|y\|_2^2 \cdot D_{\text{mse}}.\end{aligned}$$

The theorem follows by invoking the MSE bounds from Theorem 1 with bit-width $b - 1$. ■

Does this work?

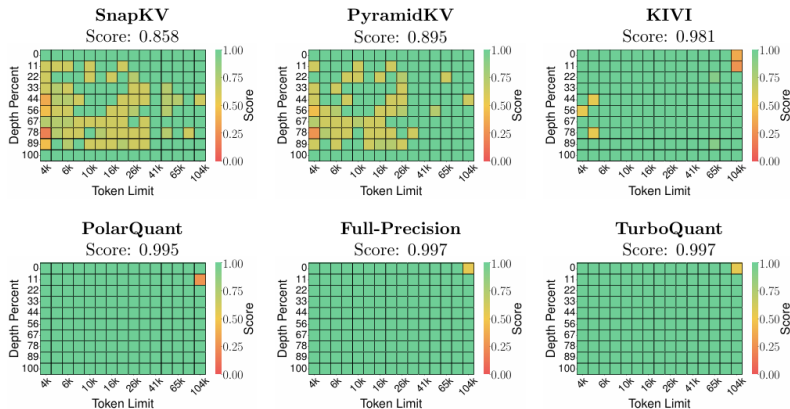


Figure 4: Evaluation of Llama-3.1-8B-Instruct on the “Needle-In-A-Haystack” test, where a model must retrieve a hidden sentence from long-context sequences. While some methods struggle with recall, TURBOQUANT, despite being more than $4\times$ quantized, achieves the same exact performance as the uncompressed baseline.

Does this work?

Method	KV Size	SingleQA	MultiQA	Summarization	Few shot	Synthetic	Code	Average
Llama-3.1-8B-Instruct								
Full Cache	16	45.29	45.16	26.55	68.38	59.54	46.28	50.06
KIVI	3	43.38	37.99	27.16	68.38	59.50	44.68	48.50
KIVI	5	45.04	45.70	26.47	68.57	59.55	46.41	50.16
PolarQuant	3.9	45.18	44.48	26.23	68.25	60.07	45.24	49.78
TURBOQUANT (ours)	2.5	44.16	44.96	24.80	68.01	59.65	45.76	49.44
TURBOQUANT (ours)	3.5	45.01	45.31	26.00	68.63	59.95	46.17	50.06
Ministral-7B-Instruct								
Full Cache	16	47.53	49.06	26.09	66.83	53.50	47.90	49.89
TURBOQUANT (ours)	2.5	48.38	49.22	24.91	66.69	53.17	46.83	49.62

Table 1: LongBench-V1 [10] results of various KV cache compression methods on Llama-3.1-8B-Instruct.

Figure: End-to-end Generation on LongBench

Does this work?

Approach	d=200	d=1536	d=3072
Product Quantization	37.04	239.75	494.42
RabitQ	597.25	2267.59	3957.19
TURBOQUANT	0.0007	0.0013	0.0021

Table 2: Quantization time (in seconds) for different approaches across various dimensions using 4-bit quantization.

Figure: Near Neighbor Search experiment

KV Cache Transform Coding for Compact Storage in LLM Inference

PCA-Based Compression, Bit Allocation, and Retrieval Latency

Rudrapratap Chinhara

What does KVTC do?

Core Mechanism

KVTC is primarily a storage/transfer compression method: during decoding, the model operates on the decompressed KV cache.

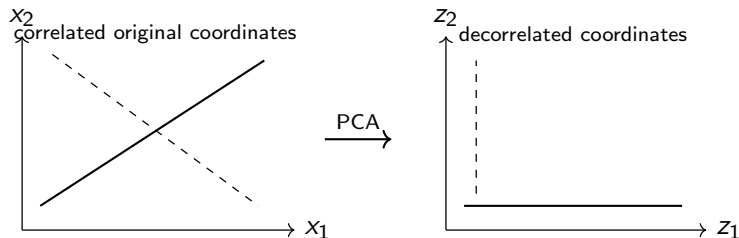
compress KV → store/transfer cheaply → decompress when reused

1. The intuition behind using PCA

Suppose a KV vector has many correlated coordinates:

$$x = [x_1, x_2, \dots, x_p].$$

If dimensions carry redundant information, uniform quantization wastes bits.



PCA rotates the coordinate system so that variance is concentrated:

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_r.$$

Key consequence: the first components deserve more precision; trailing components can tolerate aggressive quantization or even be discarded.

Why is PCA plausible for KV caches?

The paper first tests whether different attention heads occupy related latent spaces. For two heads i, j , solve an orthogonal Procrustes problem:

$$R^* = \arg \min_R \|K_i - K_j R\|_F \quad \text{s.t. } R^\top R = I.$$

Then compare token-wise cosine similarity before and after alignment.

Empirical observation

- Before alignment: inter-head cosine similarity is typically below 0.2.
- After an orthogonal alignment: similarity increases substantially for Keys and moderately for Values.

Interpretation

Different heads can use different coordinate systems for related information. A shared latent basis can therefore exploit redundancy across heads.

Cosine Similarity Heatmap for Keys and Values

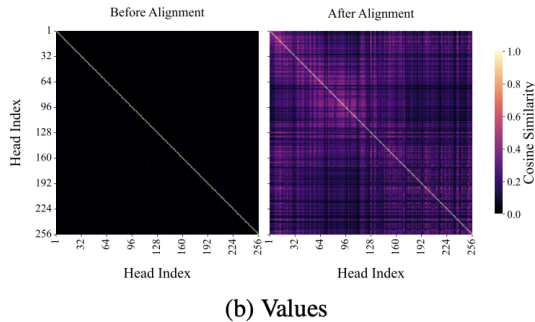
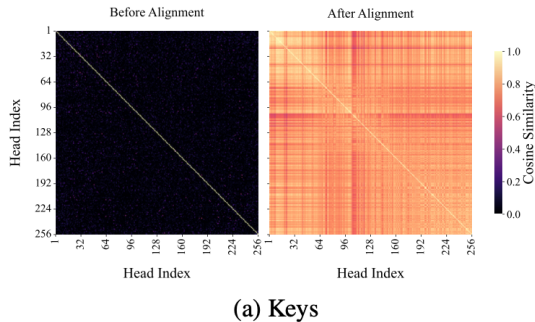


Figure: Cosine similarity before and after alignment between key(a) and value(b) heads calculated using Llama 3.1 8B on inputs from Qasper(Dasigi et al., 2021; Shaham et al., 2022).

How PCA is constructed in KVTC

For each sampled token position:

- ① collect KV vectors from selected layers and heads;
- ② undo positional rotations (RoPE);
- ③ concatenate them along the hidden dimension.

This gives a calibration matrix:

$$C \in \mathbb{R}^{n \times p}, \quad p = l h d_{\text{head}}.$$

Center it:

$$C - \mu = U \Sigma V^{\top}.$$

The right singular vectors V are the PCA directions.

How PCA is constructed in KVTC

For a new KV matrix X :

$$D = (X - \mu)V$$

and reconstruction is:

$$\hat{X} = DV^T + \mu.$$

If only $r < p$ components are kept:

$$\hat{X} \approx DV^T + \mu, \quad V \in \mathbb{R}^{p \times r}.$$

Why undo RoPE before PCA?

RoPE makes the observed Key representation depend on token position.

Conceptually:

$$K_{\text{observed}}(t) = R_t K_{\text{base}}.$$

If PCA sees the position-dependent vectors directly, it may spend capacity modeling positional rotations rather than the underlying low-dimensional structure.

KVTC therefore:

KV cache $\rightarrow$ undo RoPE $\rightarrow$ PCA

The paper explicitly identifies positional embeddings as distorting the apparent low-rank structure of Keys.

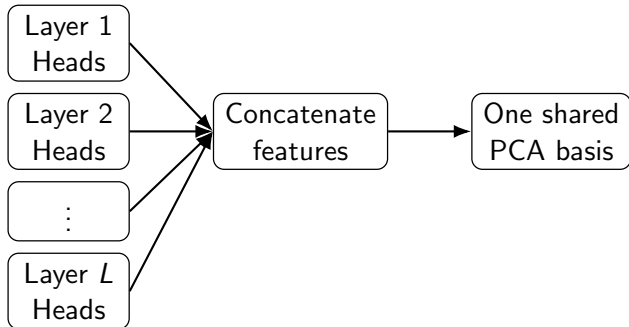
Why concatenate across layers and heads?

For Llama 3.1 8B:

$$32 \text{ layers} \times 8 \text{ KV heads} \times 128 = 32768$$

features per Key/Value position.

KVTC does not learn an independent PCA basis for every small tensor. It can concatenate multiple layers/heads and learn a larger shared basis.



Ablations show better performance as more layers participate in the concatenation.

Ablation: how many layers should share the PCA basis?

Experimental setup

DP quantization disabled to isolate the effect of PCA.

Sliding window: $w = 16$

Method	PCA layers	GSM8K	MMLU	QASPER	LITM	RULER-VT
kvtc _{64×} -DPQ	1	0.9 _{0.3}	0.3 _{0.0}	18.0	0.0 _{0.0}	0.1 _{0.1}
	2	1.4 _{0.3}	0.1 _{0.0}	17.7	0.0 _{0.0}	0.1 _{0.1}
	4	1.5 _{0.3}	0.1 _{0.0}	18.1	0.0 _{0.0}	0.1 _{0.1}
	8	2.4 _{0.4}	0.6 _{0.1}	19.4	0.0 _{0.0}	0.2 _{0.1}
	16	13.9 _{1.0}	1.9 _{0.1}	22.1	0.0 _{0.0}	10.0 _{0.9}
	32	21.8 _{1.1}	6.5 _{0.2}	26.7	5.1 _{0.3}	55.5 _{1.6}

Subscript denotes standard deviation across repeated runs.

Head concatenation: Keys and Values behave differently

Experimental setup

32× dimensionality reduction applied separately to Keys or Values kvtc_{32×}-DPQ

Cache	PCA layers	GSM8K	MMLU	QASPER	LITM	RULER-VT
Keys	1	3.2 _{0.5}	4.4 _{0.2}	21.7	0.0 _{0.0}	0.1 _{0.1}
	4	25.9 _{1.2}	27.5 _{0.4}	25.9	0.2 _{0.0}	2.5 _{0.5}
	8	49.9 _{1.4}	43.6 _{0.4}	35.2	44.5 _{0.6}	74.9 _{1.4}
	16	52.5 _{1.4}	52.3 _{0.4}	38.6	86.0 _{0.4}	90.1 _{1.9}
	32	53.2 _{1.4}	56.3 _{0.4}	38.8	96.7 _{0.2}	97.4 _{0.5}
Values	1	2.0 _{0.4}	0.9 _{0.1}	18.0	0.0 _{0.0}	2.8 _{0.5}
	4	28.9 _{1.2}	29.0 _{0.4}	30.0	15.6 _{0.5}	68.2 _{1.5}
	8	40.9 _{1.4}	41.8 _{0.4}	34.6	53.1 _{1.6}	76.4 _{1.3}
	16	49.2 _{1.4}	45.7 _{0.4}	37.1	82.6 _{0.5}	88.6 _{1.0}
	32	48.7 _{1.4}	48.6 _{0.4}	38.7	81.4 _{0.5}	89.0 _{1.0}

Subscript denotes standard deviation across repeated runs.

PCA is only half the story

After projection:

$$D = (X - \mu)V,$$

PCA has concentrated variance, but D may still contain many dimensions.

A naive approach is:

keep first k PCs; discard the rest.

KVTC instead asks:

How should a fixed number of bits be distributed across PCs?
--

Example:

$[8, 8, 8, 8, \dots]$ vs. $[8, 8, 6, 4, 2, 1, 0, 0, \dots]$.

The second allocation exploits the fact that PCA orders components by explained variance.

2. Bit allocation: the optimization problem

Let q_i be the number of bits assigned to principal component i .

KVTC wants to minimize reconstruction error under a global bit budget:

$$\min_{\{q_i\}} \left\| DV^\top - D_{q_1, \dots, q_k} V^\top \right\|_F^2$$

subject to:

$$\text{total storage} \leq B.$$

Because V is orthonormal:

$$\left\| (D - D_q) V^\top \right\|_F^2 = \|D - D_q\|_F^2.$$

Therefore:

optimize the quantization error directly in PCA space

This is what makes the DP formulation simple.

Dynamic programming: what exactly is being optimized?

Define:

$DP[i, b]$ = minimum reconstruction error using the first i PCs with b payload bits.

The DP also stores a backpointer:

$BP[i, b]$ = the local decision that achieved $DP[i, b]$.

At each step, the algorithm considers:

group size + bit width

and chooses the combination that gives the lowest error while respecting the remaining budget.

PCs $1-i$ + b bits $\longrightarrow$ minimum error
--

The DP is solved once during calibration and reused at inference.

KVTC does not assign bits independently to every scalar

The paper groups consecutive PCA coordinates.

Allowed group sizes:

$\{1, 16, 64, 256, 1024\}$

Each group shares:

16-bit shift + 16-bit scaling factor.

The DP therefore chooses:

group size + bit width

not merely a bit width for each scalar.

The total budget includes:

payload bits + per-group shift/scale overhead.

This matters because extremely small groups would waste storage on metadata.

Dynamic Programming for Bit Allocation

DP state

$E[i, b]$ = minimum reconstruction error for the first i PCA components using b bits.

For a candidate block of size s and precision type t :

$$E[i, b] = \min_{s, t} [E[i - s, b - c(s, t)] + \Delta e(s, t)]$$

where

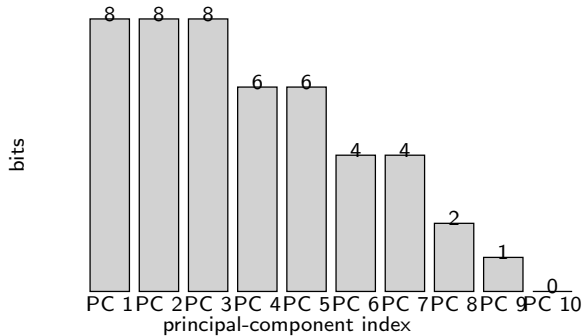
- $c(s, t)$ = bits required by block (s, t)
- $\Delta e(s, t)$ = change in reconstruction error
- $s \in \{1, 16, 64, 256, 1024\}$
- $t \in \{\text{None}, \text{INT2}, \text{INT4}, \text{FP8}\}$

After filling the DP table

Start from the target state $E[N, B]$ and **backtrack** to recover the block size and precision assigned to every PCA component group.

What does the learned allocation look like?

Because PCA sorts components by explained variance, the learned bit widths generally decrease with component index.



Important

The DP often assigns **zero bits to trailing PCs**. These dimensions are effectively removed.

Why DP matters: PCA truncation is not enough

The paper compares full KVTC against a variant that removes the dynamic-programming quantization and relies on pure PCA truncation.

At $64\times$ compression with a 16-token sliding window on Llama 3.1 8B:

	KVTC	PCA-only
GSM8K	53.0	21.8
MMLU	56.0	6.5
QASPER	38.1	26.7
LITM	80.0	5.1
RULER-VT	88.0	55.5

Attention sinks and recent tokens

KVTC does not compress every token equally.

It protects:

oldest attention-sink tokens

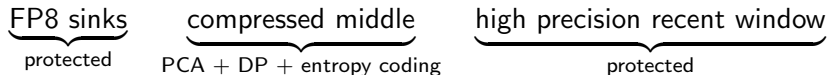
and

most recent tokens

Reason:

- attention sinks can receive disproportionate attention;
- recent tokens are especially important for next-token prediction;
- both regions show higher sensitivity to compression.

Default evaluations use a sliding window of recent tokens and protect the first few sink tokens.



Ablation Results: Preserving Attention Sink Tokens

Motivation

The first few tokens can act as **attention sinks** (Xiao et al., 2024). PCA compression causes disproportionately large information loss for these initial tokens.

Experiment: Compare two settings at high compression:

$$\text{kvtc}_{64\times}^{\blacktriangleright 0} \quad \text{vs.} \quad \text{kvtc}_{64\times}^{\blacktriangleright 4}$$

- $\blacktriangleright 0$: compress the first 4 tokens normally
- $\blacktriangleright 4$: **exclude the first 4 tokens from compression**

Ablation Results at 64× Compression

Model	Setting	GSM8K	MMLU	QASPER	LITM	RULER-VT
Llama 3.1 8B	16×, ▶ 0	56.8	60.3	40.2	99.2	98.2
	16×, ▶ 4	56.9	60.1	40.7	99.3	99.1
	64×, ▶ 0	1.6	9.9	27.6	0.0	61.8
	64×, ▶ 4	57.2	60.7	37.8	90.2	95.9
MN-Minitron 8B	64×, ▶ 0	59.5	61.9	37.9	55.1	91.5
	64×, ▶ 4	57.8	62.1	38.1	59.5	93.4
Mistral NeMo 12B	64×, ▶ 0	61.9	62.4	37.5	84.8	95.8
	64×, ▶ 4	61.9	61.4	38.0	95.3	98.0

Key takeaway

Excluding the first four tokens improves performance at high compression, especially for **Llama 3.1 8B** on long-context tasks.

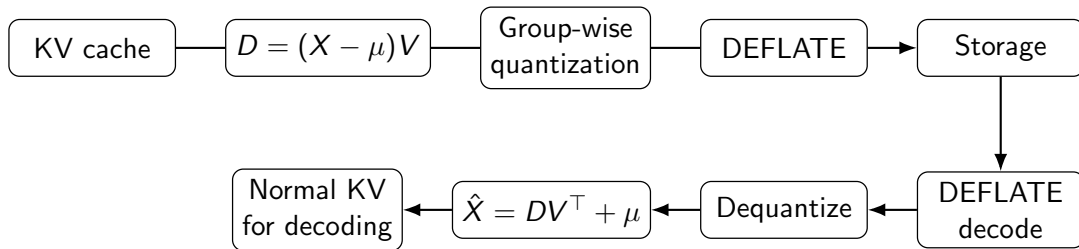
3. Implementation: calibration

Calibration is performed once per model and target compression ratio.

- 1 Run calibration sequences through the model.
- 2 Collect Key/Value caches.
- 3 Sample token positions into a global pool.
- 4 Undo positional rotations.
- 5 Concatenate selected layers/heads.
- 6 Compute mean μ and randomized SVD/PCA.
- 7 Run DP to determine bit allocation.
- 8 Store V , μ , quantization configuration.

The paper reports calibration of a 12B model within about 10 minutes on an H100 GPU.
No model weights are fine-tuned.

Implementation: compression and decompression



Critical implementation detail

During decoding, KVTC does **not** operate directly on the compressed representation. The cache is decompressed first.

Entropy coding: the final compression stage

After quantization, values are packed into a byte array.

KVTC applies:

DEFLATE

using NVIDIA's:

nvCOMP

for GPU-parallel execution.

This stage is **lossless**.

Its additional compression is content-dependent.

Conceptually:

PCA $\rightarrow$ variable-bit symbols $\rightarrow$ repeated/statistical structure $\rightarrow$ DEFLATE.

For Llama 3.1 8B, target compression ratios of $8\times$, $16\times$, $32\times$, and $64\times$ become approximately $9\text{--}10\times$, $18\text{--}22\times$, $34\text{--}44\times$, and $64\text{--}88\times$ respectively after entropy coding.

Latency Improvements with KVTC

The paper evaluates a simple implementation on an H100 with Mistral NeMo 12B. For an 8K context, restoring a compressed KV cache gives a much lower TTFT than recomputing the KV cache:

$$T_{\text{KVTC restore}} \ll T_{\text{recompute KV}}$$

Note

KVTC trades computation at eviction time for substantially cheaper cache restoration later. The reported $\sim 8\times$ improvement is specifically a comparison with **recomputing an evicted KV cache**; it is not an unconditional end-to-end speedup for every decoding step.

Latency: KVTC decompression vs. KV recomputation

Experimental setup

Mistral NeMo 12B — NVIDIA H100 — bf16

Simple Hugging Face implementation

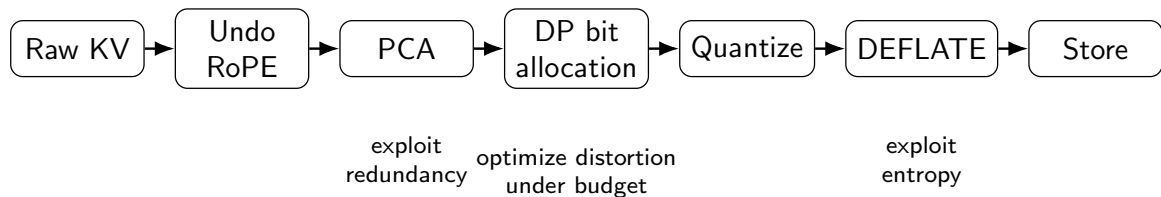
Module	BS=8, CTX=8K		BS=2, CTX=16K	
	Compress	Decompress	Compress	Decompress
Project	153 ms	156 ms	78 ms	75 ms
Quantize	67 ms	37 ms	39 ms	27 ms
Deflate	137 ms	64 ms	66 ms	36 ms
Total	379 ms	267 ms	194 ms	143 ms
<i>Time-to-first-token (TTFT)</i>				
Vanilla recompute	3098 ms		1780 ms	
KVTC decompression	380 ms		208 ms	

BS denotes batch size, CTX context length.

Limitations and engineering issues

- ➊ **Offline calibration cost:** a representative calibration set and SVD are required.
- ➋ **PCA storage:** projection matrices add non-trivial model-specific state.
- ➌ **Decompression overhead:** inverse projection is computationally expensive.
- ➍ **No direct compressed-space attention:** decoding currently uses reconstructed KV.
- ➎ **Aggressive compression hurts quality:** $64\times$ and beyond can degrade long-context tasks.
- ➏ **Compression ratio depends on protected tokens and entropy coding:** nominal CR is not always effective end-to-end CR.
- ➐ **Kernel implementation is still simple/research-grade:** the paper identifies kernel fusion and hierarchical PCA as optimization opportunities.
- ➑ **Calibration generalization:** the PCA basis must represent the cache distributions encountered in deployment.

The complete workflow



KVTC = decorrelation + rate-distortion bit allocation + lossless coding

A different route to KV-cache compression

KIVI

SO FAR

Undo RoPE

PCA

Bit allocation

Quantize

DEFLATE

Can **direct quantization** work at 2 bits?

KIVI targets the KV cache in a **tuning-free** setting -
without retraining or a learned compression transform.

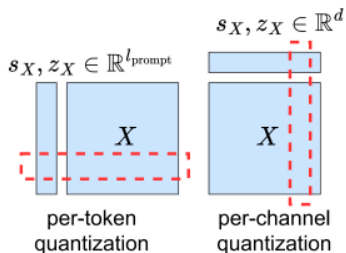
2-BIT TARGET

TUNING-FREE

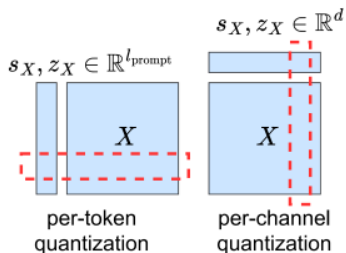
DIRECT KV QUANTIZATION

The first design question is deliberately simple:
What quantization granularity should we use?

Quantization granularity



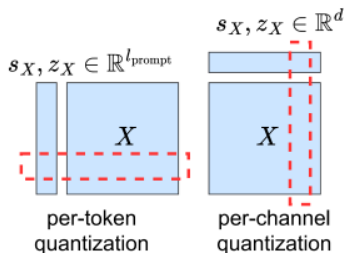
The KV-cache matrix is $X \in \mathbb{R}^{l_{\text{prompt}} \times d}$, with tokens along rows and channels along columns.



The KV-cache matrix is $X \in \mathbb{R}^{l_{\text{prompt}} \times d}$, with tokens along rows and channels along columns.

Per-token quantization assigns one scale s_X and zero point z_X to each row.

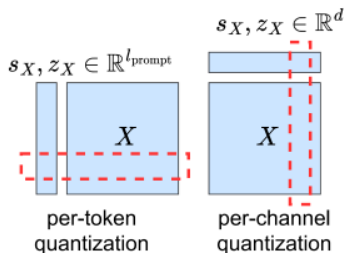
Quantization granularity



The KV-cache matrix is $X \in \mathbb{R}^{l_{\text{prompt}} \times d}$, with tokens along rows and channels along columns.

Per-token quantization assigns one scale s_X and zero point z_X to each row.

Per-channel quantization assigns one scale and zero point to each column.



The KV-cache matrix is $X \in \mathbb{R}^{l_{\text{prompt}} \times d}$, with tokens along rows and channels along columns.

Per-token quantization assigns one scale s_X and zero point z_X to each row.

Per-channel quantization assigns one scale and zero point to each column.

At 2 bits, this grouping direction becomes a central design decision.

2-bit quantization ablation

LLaMA-2-13B KV scheme	CoQA	TruthfulQA
-----------------------	------	------------

16-bit KV

4-bit: K token, V token

2-bit: K token, V token

2-bit: K channel, V channel

2-bit: K token, V channel

2-bit: K channel, V token

2-bit quantization ablation

LLaMA-2-13B KV scheme	CoQA	TruthfulQA
16-bit KV	66.37	29.53
4-bit: K token, V token		
2-bit: K token, V token		
2-bit: K channel, V channel		
2-bit: K token, V channel		
2-bit: K channel, V token		

2-bit quantization ablation

LLaMA-2-13B KV scheme	CoQA	TruthfulQA
16-bit KV	66.37	29.53
4-bit: K token, V token	66.48	29.51
2-bit: K token, V token		
2-bit: K channel, V channel		
2-bit: K token, V channel		
2-bit: K channel, V token		

2-bit quantization ablation

LLaMA-2-13B KV scheme	CoQA	TruthfulQA
16-bit KV	66.37	29.53
4-bit: K token, V token	66.48	29.51
2-bit: K token, V token	52.93	24.98
2-bit: K channel, V channel		
2-bit: K token, V channel		
2-bit: K channel, V token		

2-bit quantization ablation

LLaMA-2-13B KV scheme	CoQA	TruthfulQA
16-bit KV	66.37	29.53
4-bit: K token, V token	66.48	29.51
2-bit: K token, V token	52.93	24.98
2-bit: K channel, V channel	2.88	0.74
2-bit: K token, V channel		
2-bit: K channel, V token		

2-bit quantization ablation

LLaMA-2-13B KV scheme	CoQA	TruthfulQA
16-bit KV	66.37	29.53
4-bit: K token, V token	66.48	29.51
2-bit: K token, V token	52.93	24.98
2-bit: K channel, V channel	2.88	0.74
2-bit: K token, V channel	2.80	0.26
2-bit: K channel, V token		

2-bit quantization ablation

LLaMA-2-13B KV scheme	CoQA	TruthfulQA
16-bit KV	66.37	29.53
4-bit: K token, V token	66.48	29.51
2-bit: K token, V token	52.93	24.98
2-bit: K channel, V channel	2.88	0.74
2-bit: K token, V channel	2.80	0.26
2-bit: K channel, V token	63.53	28.60

2-bit quantization ablation

LLaMA-2-13B KV scheme	CoQA	TruthfulQA
16-bit KV	66.37	29.53
4-bit: K token, V token	66.48	29.51
2-bit: K token, V token	52.93	24.98
2-bit: K channel, V channel	2.88	0.74
2-bit: K token, V channel	2.80	0.26
2-bit: K channel, V token	63.53	28.60

Empirical result

At 4 bits, ordinary per-token quantization is effective. At 2 bits, the grouping direction becomes the main design choice.

LLaMA-2-13B KV scheme	CoQA	TruthfulQA
16-bit KV	66.37	29.53
4-bit: K token, V token	66.48	29.51
2-bit: K token, V token	52.93	24.98
2-bit: K channel, V channel	2.88	0.74
2-bit: K token, V channel	2.80	0.26
2-bit: K channel, V token	63.53	28.60

Empirical result

At 4 bits, ordinary per-token quantization is effective. At 2 bits, the grouping direction becomes the main design choice.

Robust asymmetric pattern

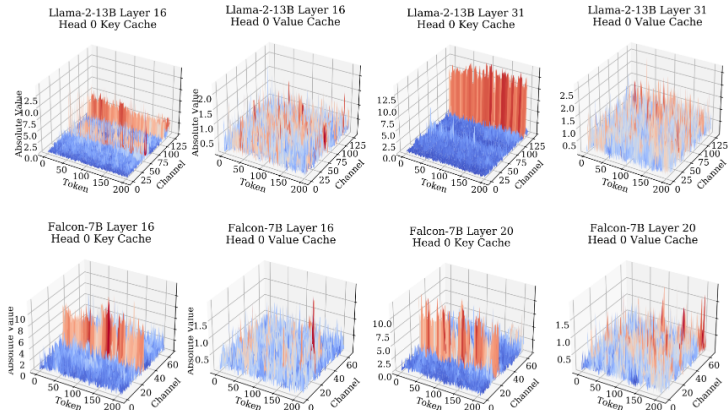
- **Keys** → per-channel
- **Values** → per-token

The remaining 2-bit configurations degrade substantially.

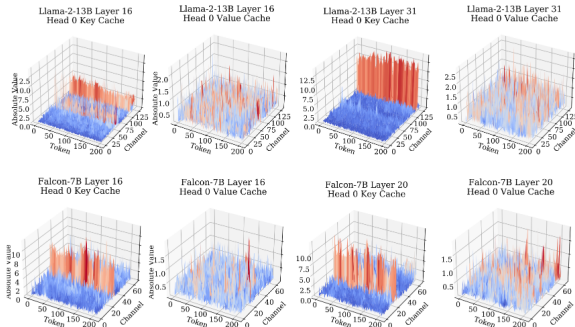
Interpretation

The ablation motivates a distinct quantization strategy for keys and values rather than a single uniform rule.

Empirical distributions of keys and values



Empirical distributions of keys and values



Interpretation. Key activations contain persistent large-magnitude channels, which supports **per-channel quantization** for keys. Value activations do not show the same stable channel-wise outlier structure; their quantization rule is instead justified by downstream attention-output error.

Error propagation through attention

KIVI evaluates two distinct error pathways.

For keys: quantization disturbs the attention score, $A' = \text{softmax}(\mathbf{q}_t K'^\top)$. Because key outliers are channel-localized, per-channel quantization greatly reduces attention-score error.

	Per-token	Per-channel
$\ K - K'\ _F / \ K\ _F$	13.67	4.55
$\ A - A'\ _F / \ A\ _F$	47.00	9.60

Frobenius norm:

$$\|X\|_F = \sqrt{\sum_{i,j} |x_{ij}|^2}.$$

Error propagation through attention

KIVI evaluates two distinct error pathways.

For keys: quantization disturbs the attention score, $A' = \text{softmax}(\mathbf{q}_t K'^\top)$. Because key outliers are channel-localized, per-channel quantization greatly reduces attention-score error.

For values: attention output is a token mixture,

$$[AV]_{i,:} = \sum_{j=1}^T A_{ij} V_{j,:}.$$

Since attention is sparse, only a few tokens dominate the sum. Per-token quantization confines error to each token rather than spreading it across channels.

	Per-token	Per-channel
$\ K - K'\ _F / \ K\ _F$	13.67	4.55
$\ A - A'\ _F / \ A\ _F$	47.00	9.60
$\ V - V'\ _F / \ V\ _F$	4.57	3.73
$\ AV - AV'\ _F / \ AV\ _F$	3.55	49.89

Frobenius norm:

$$\|X\|_F = \sqrt{\sum_{i,j} |X_{ij}|^2}.$$

Error propagation through attention

KIVI evaluates two distinct error pathways.

For keys: quantization disturbs the attention score, $A' = \text{softmax}(\mathbf{q}_t K'^\top)$. Because key outliers are channel-localized, per-channel quantization greatly reduces attention-score error.

For values: attention output is a token mixture,

$$[AV]_{i,:} = \sum_{j=1}^T A_{ij} V_{j,:}.$$

Since attention is sparse, only a few tokens dominate the sum. Per-token quantization confines error to each token rather than spreading it across channels.

	Per-token	Per-channel
$\ K - K'\ _F / \ K\ _F$	13.67	4.55
$\ A - A'\ _F / \ A\ _F$	47.00	9.60
$\ V - V'\ _F / \ V\ _F$	4.57	3.73
$\ AV - AV'\ _F / \ AV\ _F$	3.55	49.89

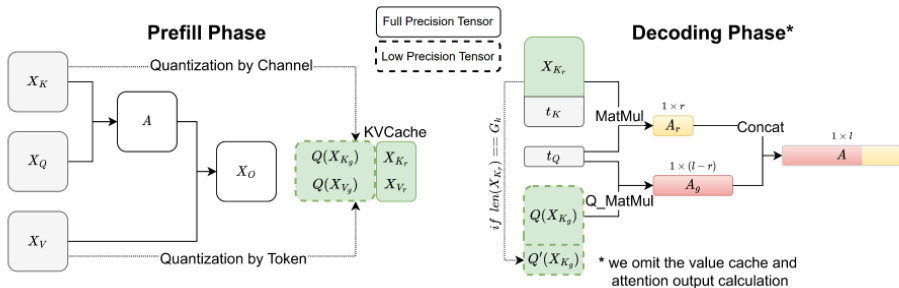
Frobenius norm:

$$\|X\|_F = \sqrt{\sum_{i,j} |x_{ij}|^2}.$$

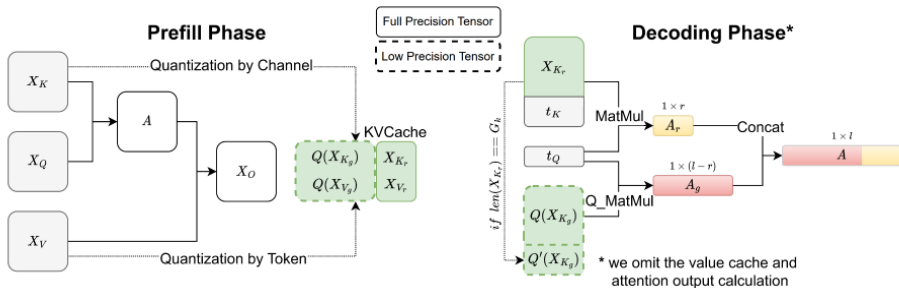
Downstream error is decisive

For values, raw reconstruction errors are similar, but downstream attention-output error is far smaller with token-wise quantization.

Streaming quantization algorithm

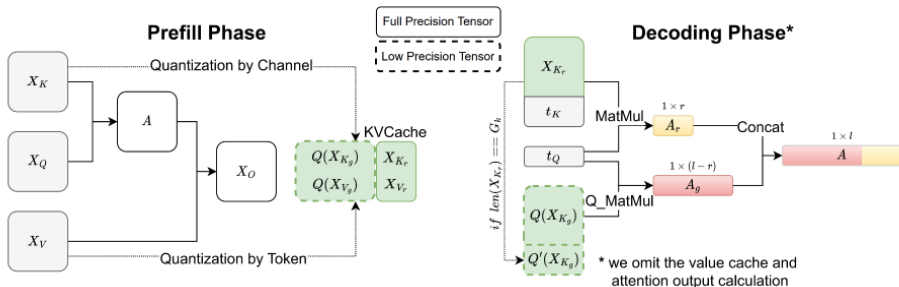


Streaming quantization algorithm



Keys: accumulate tokens into groups of size G and quantize each completed group per-channel.

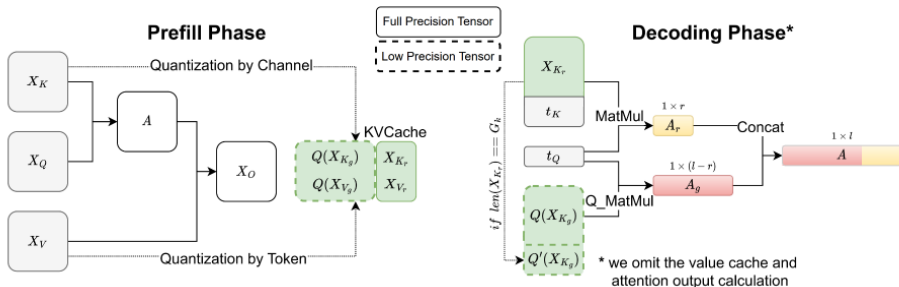
Streaming quantization algorithm



Keys: accumulate tokens into groups of size G and quantize each completed group per-channel.

Values: quantize historical values per-token as they leave the residual window.

Streaming quantization algorithm



Keys: accumulate tokens into groups of size G and quantize each completed group per-channel.

Values: quantize historical values per-token as they leave the residual window.

Residual window: keep the most recent R tokens in full precision and combine quantized and residual portions during attention.

2.6× lower peak memory

$2.6\times$ lower peak memory

up to $4\times$ larger batch size

Measured effect

For the LLaMA-2-7B system experiment, KV-memory reduction supports substantially larger batches at the same device-memory limit.

$2.6\times$ lower peak memory

up to $4\times$ larger batch size

$2.35\text{--}3.47\times$ higher throughput

Measured effect

For the LLaMA-2-7B system experiment, KV-memory reduction supports substantially larger batches at the same device-memory limit.

Reported implementation features

- ▶ up to $4\times$ larger feasible batch size;
- ▶ $2.35\text{--}3.47\times$ higher throughput;
- ▶ fused dequantization and matrix multiplication reduce runtime overhead.

Selected GSM8K exact-match results

Model	16-bit	KIVI-4	KIVI-2
Llama-2-7B	13.50	13.80	12.74
Llama-2-13B	22.67	23.65	20.77
Falcon-7B	4.55	4.47	3.41
Mistral-7B	38.36	37.30	36.01

Selected GSM8K exact-match results

Model	16-bit	KIVI-4	KIVI-2
Llama-2-7B	13.50	13.80	12.74
Llama-2-13B	22.67	23.65	20.77
Falcon-7B	4.55	4.47	3.41
Mistral-7B	38.36	37.30	36.01

Selected LongBench averages

Model	16-bit	KIVI-4	KIVI-2
Llama-2-7B	44.52	44.59	44.27
Llama-2-13B	44.85	44.48	44.69
Mistral-7B	46.58	46.56	45.85

Selected GSM8K exact-match results

Model	16-bit	KIVI-4	KIVI-2
Llama-2-7B	13.50	13.80	12.74
Llama-2-13B	22.67	23.65	20.77
Falcon-7B	4.55	4.47	3.41
Mistral-7B	38.36	37.30	36.01

Selected LongBench averages

Model	16-bit	KIVI-4	KIVI-2
Llama-2-7B	44.52	44.59	44.27
Llama-2-13B	44.85	44.48	44.69
Mistral-7B	46.58	46.56	45.85

Interpretation

- ▶ KIVI preserves most task accuracy at 2 bits for the evaluated LLaMA and Mistral models.
- ▶ Falcon benefits less because multi-query attention already reduces the KV-cache footprint.
- ▶ The contribution combines the asymmetric quantization rule with a small full-precision residual window.

Complete LM-Eval results (LLaMA-2 models)

Model		CoQA	TruthfulQA	GSM8K
Llama-2-7B	16bit	63.88	30.76	13.50
	4bit (K - T, V - T)	64.82	29.85	12.28
	2bit (K - C, V - T)	59.08	33.10	5.76
	2bit (K - T, V - T)	39.88	18.29	0.83
	2bit (K - C, V - C)	3.60	0.27	0.00
	2bit (K - T, V - C)	1.30	0.49	0.08
	KIVI-4	63.78	30.80	13.80
	KIVI-2	63.05	33.95	12.74
Llama-2-13B	16bit	66.37	29.53	22.67
	4bit (K - T, V - T)	66.73	29.14	20.92
	2bit (K - C, V - T)	63.53	28.60	12.21
	2bit (K - T, V - T)	52.93	24.98	4.55
	2bit (K - C, V - C)	2.88	0.74	0.00
	2bit (K - T, V - C)	2.80	0.26	0.08
	KIVI-4	66.38	29.49	23.65
	KIVI-2	66.23	29.84	20.77

Table content. Complete CoQA, TruthfulQA, and GSM8K results for LLaMA-2-7B and LLaMA-2-13B, including all 2-bit granularity combinations and KIVI-4/KIVI-2.

Complete LM-Eval results (LLaMA-2 models)

Model		CoQA	TruthfulQA	GSM8K
Llama-2-7B	16bit	63.88	30.76	13.50
	4bit (K - T, V - T)	64.82	29.85	12.28
	2bit (K - C, V - T)	59.08	33.10	5.76
	2bit (K - T, V - T)	39.88	18.29	0.83
	2bit (K - C, V - C)	3.60	0.27	0.00
	2bit (K - T, V - C)	1.30	0.49	0.08
	KIVI-4	63.78	30.80	13.80
	KIVI-2	63.05	33.95	12.74
Llama-2-13B	16bit	66.37	29.53	22.67
	4bit (K - T, V - T)	66.73	29.14	20.92
	2bit (K - C, V - T)	63.53	28.60	12.21
	2bit (K - T, V - T)	52.93	24.98	4.55
	2bit (K - C, V - C)	2.88	0.74	0.00
	2bit (K - T, V - C)	2.80	0.26	0.08
	KIVI-4	66.38	29.49	23.65
	KIVI-2	66.23	29.84	20.77

Table content. Complete CoQA, TruthfulQA, and GSM8K results for LLaMA-2-7B and LLaMA-2-13B, including all 2-bit granularity combinations and KIVI-4/KIVI-2.

Complete LM-Eval results (LLaMA-2 models)

Model		CoQA	TruthfulQA	GSM8K
Llama-2-7B	16bit	63.88	30.76	13.50
	4bit (K - T, V - T)	64.82	29.85	12.28
	2bit (K - C, V - T)	59.08	33.10	5.76
	2bit (K - T, V - T)	39.88	18.29	0.83
	2bit (K - C, V - C)	3.60	0.27	0.00
	2bit (K - T, V - C)	1.30	0.49	0.08
	KIVI-4	63.78	30.80	13.80
	KIVI-2	63.05	33.95	12.74
Llama-2-13B	16bit	66.37	29.53	22.67
	4bit (K - T, V - T)	66.73	29.14	20.92
	2bit (K - C, V - T)	63.53	28.60	12.21
	2bit (K - T, V - T)	52.93	24.98	4.55
	2bit (K - C, V - C)	2.88	0.74	0.00
	2bit (K - T, V - C)	2.80	0.26	0.08
	KIVI-4	66.38	29.49	23.65
	KIVI-2	66.23	29.84	20.77

Table content. Complete CoQA, TruthfulQA, and GSM8K results for LLaMA-2-7B and LLaMA-2-13B, including all 2-bit granularity combinations and KIVI-4/KIVI-2.

Complete LM-Eval results (Falcon and Mistral)

Model		CoQA	TruthfulQA	GSM8K
Falcon-7B	16bit	59.83	23.20	4.55
	4bit (K - T, V - T)	58.53	22.94	3.26
	2bit (K - C, V - T)	43.93	20.82	1.29
	2bit (K - T, V - T)	25.72	0.91	0.53
	2bit (K - C, V - C)	41.95	17.11	1.52
	2bit (K - T, V - C)	19.53	0.94	0.15
	KIVI-4	59.67	22.58	4.47
	KIVI-2	57.48	24.98	3.41
Mistral-7B	16bit	67.40	30.45	38.36
	4bit (K - T, V - T)	67.80	29.83	36.85
	2bit (K - C, V - T)	61.65	29.64	26.46
	2bit (K - T, V - T)	54.55	25.86	5.00
	2bit (K - C, V - C)	24.40	24.86	2.27
	2bit (K - T, V - C)	10.73	19.12	0.99
	KIVI-4	66.95	30.49	37.30
	KIVI-2	66.35	32.17	36.01

Table content. Complete Falcon-7B and Mistral-7B results. Falcon is the notable case in which KIVI-2 degrades more strongly; KIVI-4 remains close to full precision.

Complete LM-Eval results (Falcon and Mistral)

Model		CoQA	TruthfulQA	GSM8K
Falcon-7B	16bit	59.83	23.20	4.55
	4bit (K - T, V - T)	58.53	22.94	3.26
	2bit (K - C, V - T)	43.93	20.82	1.29
	2bit (K - T, V - T)	25.72	0.91	0.53
	2bit (K - C, V - C)	41.95	17.11	1.52
	2bit (K - T, V - C)	19.53	0.94	0.15
	KIVI-4	59.67	22.58	4.47
	KIVI-2	57.48	24.98	3.41
Mistral-7B	16bit	67.40	30.45	38.36
	4bit (K - T, V - T)	67.80	29.83	36.85
	2bit (K - C, V - T)	61.65	29.64	26.46
	2bit (K - T, V - T)	54.55	25.86	5.00
	2bit (K - C, V - C)	24.40	24.86	2.27
	2bit (K - T, V - C)	10.73	19.12	0.99
	KIVI-4	66.95	30.49	37.30
	KIVI-2	66.35	32.17	36.01

Table content. Complete Falcon-7B and Mistral-7B results. Falcon is the notable case in which KIVI-2 degrades more strongly; KIVI-4 remains close to full precision.

Complete LM-Eval results (Falcon and Mistral)

Model		CoQA	TruthfulQA	GSM8K
Falcon-7B	16bit	59.83	23.20	4.55
	4bit (K - T, V - T)	58.53	22.94	3.26
	2bit (K - C, V - T)	43.93	20.82	1.29
	2bit (K - T, V - T)	25.72	0.91	0.53
	2bit (K - C, V - C)	41.95	17.11	1.52
	2bit (K - T, V - C)	19.53	0.94	0.15
	KIVI-4	59.67	22.58	4.47
	KIVI-2	57.48	24.98	3.41
Mistral-7B	16bit	67.40	30.45	38.36
	4bit (K - T, V - T)	67.80	29.83	36.85
	2bit (K - C, V - T)	61.65	29.64	26.46
	2bit (K - T, V - T)	54.55	25.86	5.00
	2bit (K - C, V - C)	24.40	24.86	2.27
	2bit (K - T, V - C)	10.73	19.12	0.99
	KIVI-4	66.95	30.49	37.30
	KIVI-2	66.35	32.17	36.01

Table content. Complete Falcon-7B and Mistral-7B results. Falcon is the notable case in which KIVI-2 degrades more strongly; KIVI-4 remains close to full precision.

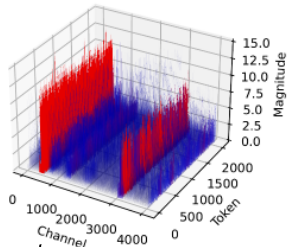
KVQuant

Towards 10 Million Context Length LLM Inference
with KV Cache Quantization

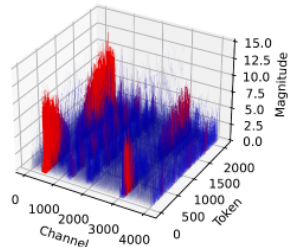
Hooper et al., UC Berkeley · NeurIPS 2024

...but RoPE smears that structure

Layer 10 Keys (pre-RoPE)



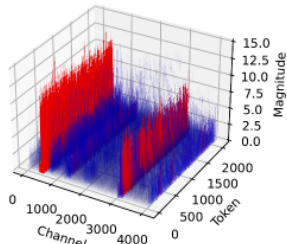
Layer 10 Keys (post-RoPE)



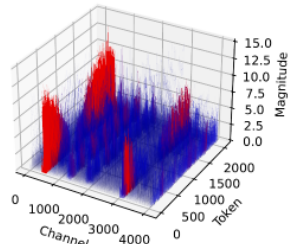
$\tilde{K}_n = R_{\theta,n}^d \cdot K_n$ rotates channels *in pairs*, by an angle that depends on the position n

...but RoPE smears that structure

Layer 10 Keys (pre-RoPE)



Layer 10 Keys (post-RoPE)

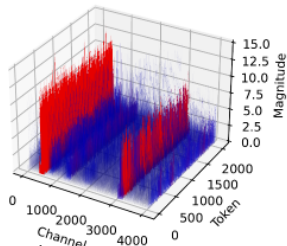


$\tilde{K}_n = R_{\theta,n}^d \cdot K_n$ rotates channels *in pairs*, by an angle that depends on the position n

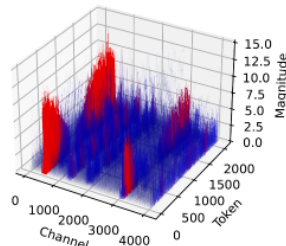
- Same layer, same data — but the clean walls on the left have broken up. A large channel leaks part of its magnitude into its partner, by a different amount at every position.

...but RoPE smears that structure

Layer 10 Keys (pre-RoPE)



Layer 10 Keys (post-RoPE)



$\tilde{K}_n = R_{\theta,n}^d \cdot K_n$ rotates channels *in pairs*, by an angle that depends on the position n

- Same layer, same data — but the clean walls on the left have broken up. A large channel leaks part of its magnitude into its partner, by a different amount at every position.
- So cache K_n and quantize **before** RoPE, then apply the rotation **after dequantization**, fused into the kernel. Another 0.82 perplexity.

Choosing the signposts: what should we minimize?

The obvious objective treats every element as equally important:

$$\min_Q \sum_i (A_i - Q(A_i))^2.$$

Choosing the signposts: what should we minimize?

The obvious objective treats every element as equally important:

$$\min_Q \sum_i (A_i - Q(A_i))^2.$$

But we do not really care about the activations — we care about the **loss**. So perturb A by the quantization error $\Delta A = A - Q(A)$ and see what comes out downstream:

Choosing the signposts: what should we minimize?

The obvious objective treats every element as equally important:

$$\min_Q \sum_i (A_i - Q(A_i))^2.$$

But we do not really care about the activations — we care about the **loss**. So perturb A by the quantization error $\Delta A = A - Q(A)$ and see what comes out downstream:

$$\mathbb{E}[|L(A) - L(A + \Delta A)|^2] \approx \mathbb{E}[(J(A)^\top \Delta A)^2], \quad J(A) = \frac{\partial L}{\partial A}$$

Choosing the signposts: what should we minimize?

The obvious objective treats every element as equally important:

$$\min_Q \sum_i (A_i - Q(A_i))^2.$$

But we do not really care about the activations — we care about the **loss**. So perturb A by the quantization error $\Delta A = A - Q(A)$ and see what comes out downstream:

$$\begin{aligned} \mathbb{E}[|L(A) - L(A + \Delta A)|^2] &\approx \mathbb{E}\left[(J(A)^\top \Delta A)^2\right], \quad J(A) = \frac{\partial L}{\partial A} \\ &= \mathbb{E}\left[\sum_i J_i^2 \Delta A_i^2\right] \quad (\text{errors zero-mean and independent}) \end{aligned}$$

Choosing the signposts: what should we minimize?

The obvious objective treats every element as equally important:

$$\min_Q \sum_i (A_i - Q(A_i))^2.$$

But we do not really care about the activations — we care about the **loss**. So perturb A by the quantization error $\Delta A = A - Q(A)$ and see what comes out downstream:

$$\begin{aligned}\mathbb{E}[|L(A) - L(A + \Delta A)|^2] &\approx \mathbb{E}\left[(J(A)^\top \Delta A)^2\right], \quad J(A) = \frac{\partial L}{\partial A} \\ &= \mathbb{E}\left[\sum_i J_i^2 \Delta A_i^2\right] \quad (\text{errors zero-mean and independent}) \\ &= \sum_i \mathcal{F}_{ii} \Delta A_i^2, \quad \mathcal{F}_{ii} = \mathbb{E}[J_i^2] \quad (\text{diagonal Fisher})\end{aligned}$$

Choosing the signposts: what should we minimize?

The obvious objective treats every element as equally important:

$$\min_Q \sum_i (A_i - Q(A_i))^2.$$

But we do not really care about the activations — we care about the **loss**. So perturb A by the quantization error $\Delta A = A - Q(A)$ and see what comes out downstream:

$$\begin{aligned}\mathbb{E}[|L(A) - L(A + \Delta A)|^2] &\approx \mathbb{E}[(J(A)^\top \Delta A)^2], \quad J(A) = \frac{\partial L}{\partial A} \\ &= \mathbb{E}\left[\sum_i J_i^2 \Delta A_i^2\right] \quad (\text{errors zero-mean and independent}) \\ &= \sum_i \mathcal{F}_{ii} \Delta A_i^2, \quad \mathcal{F}_{ii} = \mathbb{E}[J_i^2] \quad (\text{diagonal Fisher})\end{aligned}$$

$$Q^* \simeq \arg \min_Q \sum_{i=1}^N \mathcal{F}_{ii} (A_i - Q(A_i))^2$$

Turning that into something we can fit

One catch: every channel and token has its own scale, so we cannot fit a shared set of signposts to the raw A_i .

Turning that into something we can fit

One catch: every channel and token has its own scale, so we cannot fit a shared set of signposts to the raw A_i .

$$\hat{A}_i = \frac{A_i - z_i}{s_i} \in [-1, 1] \quad \implies \quad Q(A_i) = s_i Q(\hat{A}_i) + z_i$$

Turning that into something we can fit

One catch: every channel and token has its own scale, so we cannot fit a shared set of signposts to the raw A_i .

$$\hat{A}_i = \frac{A_i - z_i}{s_i} \in [-1, 1] \quad \implies \quad Q(A_i) = s_i Q(\hat{A}_i) + z_i$$

We fit in normalized space, but the error we care about lives in the original space — and dequantization scales it straight back up:

$$\Delta A_i = s_i \Delta \hat{A}_i$$

Turning that into something we can fit

One catch: every channel and token has its own scale, so we cannot fit a shared set of signposts to the raw A_i .

$$\hat{A}_i = \frac{A_i - z_i}{s_i} \in [-1, 1] \quad \implies \quad Q(A_i) = s_i Q(\hat{A}_i) + z_i$$

We fit in normalized space, but the error we care about lives in the original space — and dequantization scales it straight back up:

$$\Delta A_i = s_i \Delta \hat{A}_i$$

Substituting into the boxed objective:

$$Q^* = \arg \min_Q \sum_{i=1}^N \underbrace{\mathcal{F}_{ii} s_i^2}_{\text{weight}} (\hat{A}_i - Q(\hat{A}_i))^2$$

Turning that into something we can fit

One catch: every channel and token has its own scale, so we cannot fit a shared set of signposts to the raw A_i .

$$\hat{A}_i = \frac{A_i - z_i}{s_i} \in [-1, 1] \quad \implies \quad Q(A_i) = s_i Q(\hat{A}_i) + z_i$$

We fit in normalized space, but the error we care about lives in the original space — and dequantization scales it straight back up:

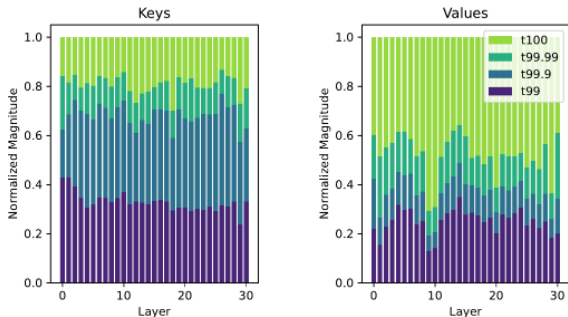
$$\Delta A_i = s_i \Delta \hat{A}_i$$

Substituting into the boxed objective:

$$Q^* = \arg \min_Q \sum_{i=1}^N \underbrace{\mathcal{F}_{ii} s_i^2}_{\text{weight}} (\hat{A}_i - Q(\hat{A}_i))^2$$

Which is just **weighted k -means**, run offline on 16 calibration samples. One datatype per layer, rescaled per vector — which is also what the kernel wants, since all threads share a single lookup table.

A few values are hogging the whole range

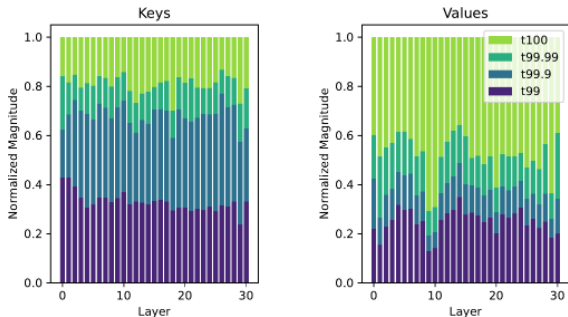


How much of the layer's range you need to cover 99 / 99.9
/ 99.99 / 100% of the values.

Isolating 1% gets us within 0.07 of fp16.

- Read the purple bars: for Keys, **99% of values fit in about a third of the range**. The rest of it exists to serve a few stragglers.

A few values are hogging the whole range

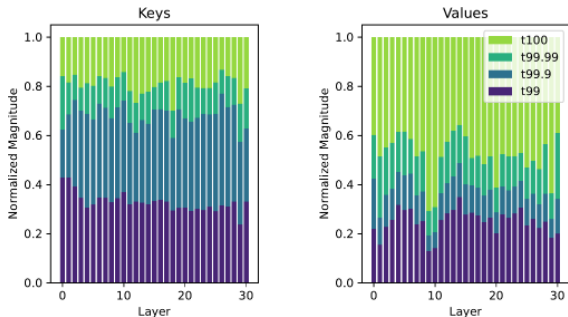


How much of the layer's range you need to cover 99 / 99.9
/ 99.99 / 100% of the values.

Isolating 1% gets us within 0.07 of fp16.

- Read the purple bars: for Keys, **99% of values fit in about a third of the range**. The rest of it exists to serve a few stragglers.
- So pull those out into a **sparse fp16 matrix** and quantize what is left over a much tighter range.

A few values are hogging the whole range

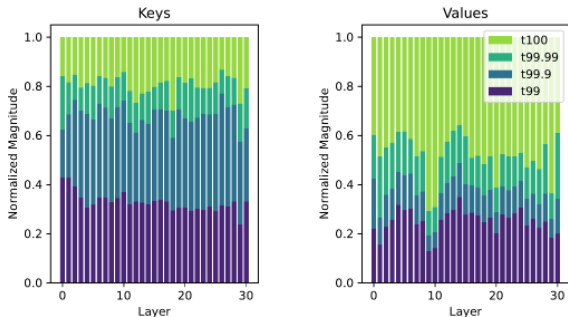


How much of the layer's range you need to cover 99 / 99.9 / 99.99 / 100% of the values.

Isolating 1% gets us within 0.07 of fp16.

- Read the purple bars: for Keys, **99% of values fit in about a third of the range**. The rest of it exists to serve a few stragglers.
- So pull those out into a **sparse fp16 matrix** and quantize what is left over a much tighter range.
- The threshold must be **per-vector**: an outlier in a quiet channel is unremarkable in a loud one. Worth 0.10 perplexity over a per-matrix threshold, at no extra cost.

A few values are hogging the whole range



How much of the layer's range you need to cover 99 / 99.9 / 99.99 / 100% of the values.

Isolating 1% gets us within 0.07 of fp16.

- Read the purple bars: for Keys, **99% of values fit in about a third of the range**. The rest of it exists to serve a few stragglers.
- So pull those out into a **sparse fp16 matrix** and quantize what is left over a much tighter range.
- The threshold must be **per-vector**: an outlier in a quiet channel is unremarkable in a loud one. Worth 0.10 perplexity over a per-matrix threshold, at no extra cost.
- Keep the **first token** in fp16 too — the attention sink is unusually sensitive, especially at 2-bit.

Putting it together

Done once, offline:

- 1 Fisher information $\mathcal{F}_{ij}$ from 16 samples
- 2 Per-channel scales and outlier thresholds for the **Keys**
- 3 Weighted k -means $\rightarrow$ one nuqX lookup table per layer

Putting it together

Done once, offline:

- 1 Fisher information $\mathcal{F}_{ij}$ from 16 samples
- 2 Per-channel scales and outlier thresholds for the **Keys**
- 3 Weighted k -means $\rightarrow$ one nuqX lookup table per layer

LLaMA-7B	PPL	KV (GB)
fp16	5.68	64.0
int3	10.87	12.0

Done during inference:

- 1 Keys: quantized pre-RoPE, per-channel, using the calibrated scales
- 2 Values: per-token scale and topk threshold computed on the CPU, hidden behind the GPU projections
- 3 Lookup-table matvec plus balanced sparse matvec, with RoPE applied inside the kernel

Putting it together

Done once, offline:

- 1 Fisher information $\mathcal{F}_{ij}$ from 16 samples
- 2 Per-channel scales and outlier thresholds for the **Keys**
- 3 Weighted k -means $\rightarrow$ one nuqX lookup table per layer

Done during inference:

- 1 Keys: quantized pre-RoPE, per-channel, using the calibrated scales
- 2 Values: per-token scale and topk threshold computed on the CPU, hidden behind the GPU projections
- 3 Lookup-table matvec plus balanced sparse matvec, with RoPE applied inside the kernel

LLaMA-7B	PPL	KV (GB)
fp16	5.68	64.0
int3	10.87	12.0
KVQuant-3b-1%	5.75	13.3
KVQuant-2b-1%	6.01	9.3

Putting it together

Done once, offline:

- 1 Fisher information $\mathcal{F}_{ij}$ from 16 samples
- 2 Per-channel scales and outlier thresholds for the **Keys**
- 3 Weighted k -means $\rightarrow$ one nuqX lookup table per layer

Done during inference:

- 1 Keys: quantized pre-RoPE, per-channel, using the calibrated scales
- 2 Values: per-token scale and topk threshold computed on the CPU, hidden behind the GPU projections
- 3 Lookup-table matvec plus balanced sparse matvec, with RoPE applied inside the kernel

LLaMA-7B	PPL	KV (GB)
fp16	5.68	64.0
int3	10.87	12.0
KVQuant-3b-1%	5.75	13.3
KVQuant-2b-1%	6.01	9.3

- $4.8\times$ smaller, under 0.1 perplexity lost
- up to $1.7\times$ faster matvec
- 1M context on one A100, 10M on eight

Beyond perplexity: downstream task performance

LLaMA-2-7B-32K · passkey retrieval success rate (Table 2)						
Method	Avg. bits	2K	4K	8K	16K	32K
fp16	16.00	1.00	1.00	1.00	1.00	1.00
KIVI-2-gs32-r128	3.05	0.76	0.72	0.72	0.68	0.70
KVQuant nuq4-1%	4.33	1.00	1.00	1.00	1.00	1.00
KVQuant nuq3-1%	3.33	0.98	1.00	1.00	1.00	1.00
KVQuant nuq2-1%	2.33	1.00	1.00	0.98	1.00	1.00

LongBench (Table 3) and RULER (Table 4) averages, same model

Method	Avg. bits	LongBench avg	RULER avg
fp16	16.00	31.96	56.40
KIVI-2-gs32-r128	3.05–3.17	30.04	39.78
KVQuant-3bit-1%	3.33	31.21	53.65
KVQuant-2bit-1%	2.33	—	36.54

- KIVI keeps a local fp16 residual window, so it represents the *tail* of the context well but loses accuracy on tasks needing the *whole* window — passkey and RULER both stress that.
- KVQuant treats every token equally, so 3-bit tracks fp16 closely; only at 2-bit does RULER show real degradation.

From 2-bit scalar quantization to 1-bit coupled quantization

KIVI identifies the appropriate quantization granularity.
Coupled Quantization additionally exploits statistical dependence across channels.

Joint entropy and inter-channel redundancy

CQ

Treat each channel as a random variable.

$$H(X_1, X_2) = H(X_1) + H(X_2) - I(X_1; X_2).$$

Treat each channel as a random variable.

$$H(X_1, X_2) = H(X_1) + H(X_2) - I(X_1; X_2).$$

Since mutual information is non-negative, $H(X_1, X_2) \leq H(X_1) + H(X_2)$, and more generally

$$H(X_1, \dots, X_c) \leq \sum_{i=1}^c H(X_i).$$

Interpretation

If channels are dependent, jointly encoding them needs fewer effective bits than encoding each one independently.

Joint entropy and inter-channel redundancy

Treat each channel as a random variable.

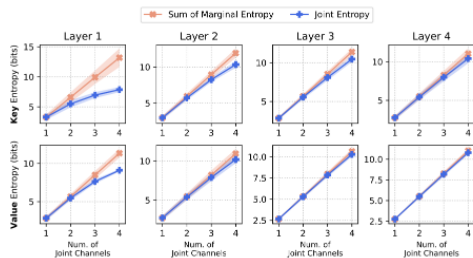
$$H(X_1, X_2) = H(X_1) + H(X_2) - I(X_1; X_2).$$

Since mutual information is non-negative, $H(X_1, X_2) \leq H(X_1) + H(X_2)$, and more generally

$$H(X_1, \dots, X_c) \leq \sum_{i=1}^c H(X_i).$$

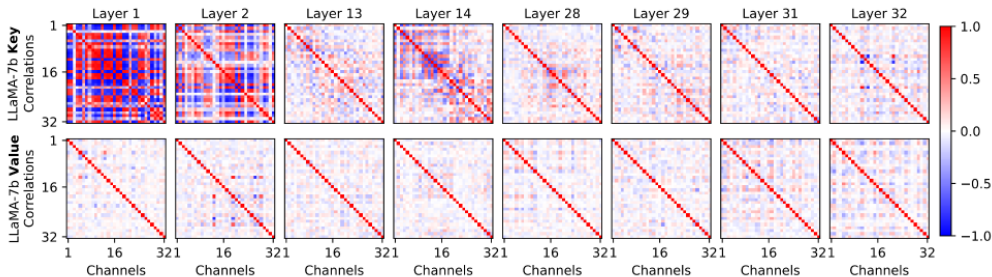
Interpretation

If channels are dependent, jointly encoding them needs fewer effective bits than encoding each one independently.

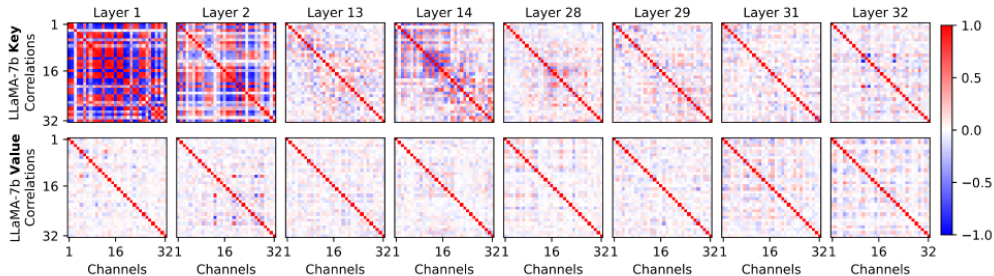


Empirical evidence. Joint entropy grows more slowly than the sum of marginal entropies as additional channels are coupled.

Empirical evidence of inter-channel dependence

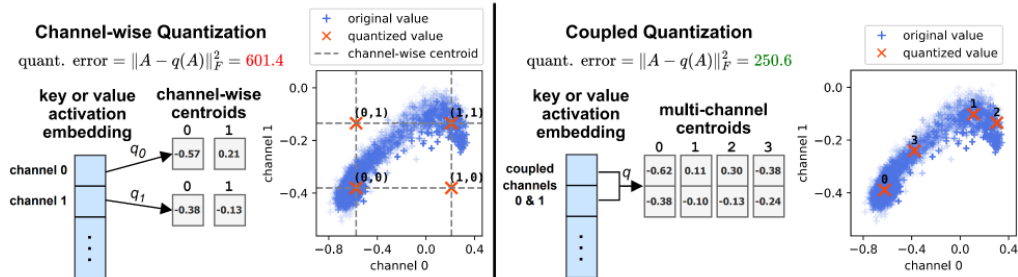


Empirical evidence of inter-channel dependence

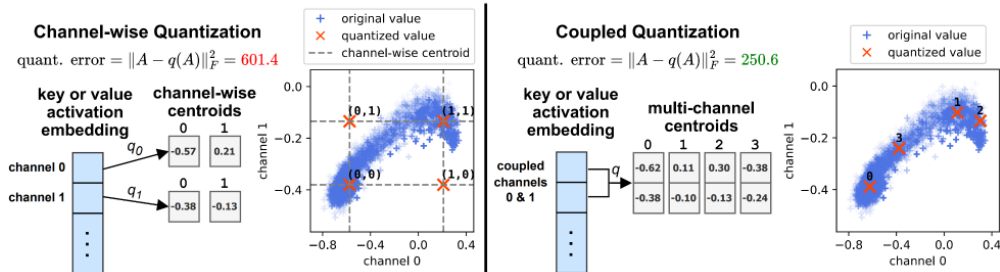


Interpretation. Several key-cache layers show substantial off-diagonal correlations, meaning channels are not independent. Value activations are generally less structured, but still exhibit non-zero dependence. This motivates **joint encoding across channels** instead of independent scalar quantization.

Geometric effect of joint channel coding



Geometric effect of joint channel coding



Interpretation. In independent channel-wise quantization, each channel gets its own scalar centroid, so the Cartesian product of those centroids cannot follow the true correlated data manifold. In **Coupled Quantization**, one code selects a centroid in the joint multi-channel space, which better matches the actual activation geometry and therefore reduces quantization error at the same effective bit budget.

Configuration notation

- ▶ CQ-4c8b couples 4 channels and uses an 8-bit code per group.
- ▶ This is the same code budget as 2-bit scalar quantization, but the codebook lives in $\mathbb{R}^4$.

Configuration notation

- ▶ CQ-4c8b couples 4 channels and uses an 8-bit code per group.
- ▶ This is the same code budget as 2-bit scalar quantization, but the codebook lives in $\mathbb{R}^4$.

Uniform centroid learning

$$\mathbf{C}_i^* = \arg \min_{|\mathbf{C}|=2^b} \sum_j \|\mathbf{a}_{i,j} - q_{\mathbf{C}}(\mathbf{a}_{i,j})\|_2^2.$$

$$q_{\mathbf{C}}(\mathbf{a}) = \arg \min_{\mathbf{c} \in \mathbf{C}} \|\mathbf{a} - \mathbf{c}\|_2^2.$$

Configuration notation

- ▶ CQ-4c8b couples 4 channels and uses an 8-bit code per group.
- ▶ This is the same code budget as 2-bit scalar quantization, but the codebook lives in $\mathbb{R}^4$.

Uniform centroid learning

$$C_i^* = \arg \min_{|C|=2^b} \sum_j \| \mathbf{a}_{i,j} - q_C(\mathbf{a}_{i,j}) \|_2^2.$$

$$q_C(\mathbf{a}) = \arg \min_{\mathbf{c} \in C} \| \mathbf{a} - \mathbf{c} \|_2^2.$$

Fisher-guided centroid learning

$$\text{diag}(F) = g(A) \odot g(A), \quad g(A) = \frac{\partial \mathcal{L}(A)}{\partial A}.$$

Use activation importance as a weighted objective:

$$C_i^* = \arg \min_C \sum_j w_{i,j} \| \mathbf{a}_{i,j} - q_C(\mathbf{a}_{i,j}) \|_2^2.$$

Configuration notation

- CQ-4c8b couples 4 channels and uses an 8-bit code per group.
- This is the same code budget as 2-bit scalar quantization, but the codebook lives in $\mathbb{R}^4$.

Uniform centroid learning

$$C_i^* = \arg \min_{|C|=2^b} \sum_j \|a_{i,j} - q_C(a_{i,j})\|_2^2.$$

$$q_C(a) = \arg \min_{c \in C} \|a - c\|_2^2.$$

Fisher-guided centroid learning

$$\text{diag}(F) = g(A) \odot g(A), \quad g(A) = \frac{\partial \mathcal{L}(A)}{\partial A}.$$

Use activation importance as a weighted objective:

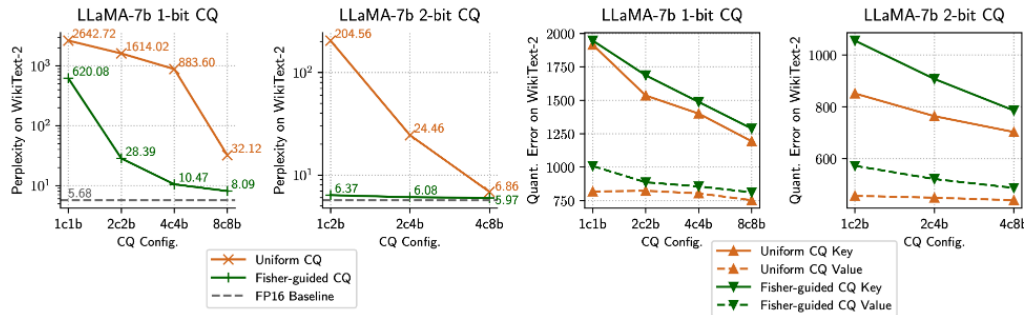
$$C_i^* = \arg \min_C \sum_j w_{i,j} \|a_{i,j} - q_C(a_{i,j})\|_2^2.$$

Methodological distinction

CQ quantizes both keys and values with channel coupling, combining cross-channel vector quantization with importance-aware codebook learning.

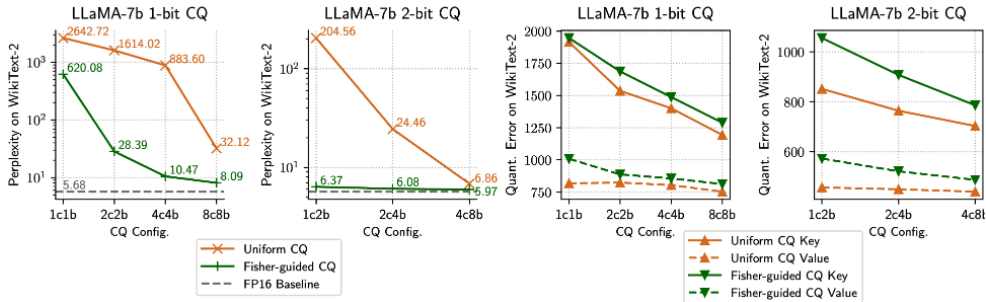
Channel coupling and Fisher guidance

CQ



Channel coupling and Fisher guidance

CQ



Empirical result. Increasing the number of coupled channels sharply improves perplexity at both the 1-bit and 2-bit operating points. Fisher-guided centroid learning further improves model quality by allocating more fidelity to important activations. The key lesson is that both **channel coupling** and **importance-aware codebook** matter.

1-bit perplexity and accuracy

WikiText-2 perplexity

Method	LLaMA-2-13B	Mistral-7B
FP16	4.57	4.76
KVQuant-1b	4709.83	203.73
KVQuant-1b-1%	13.76	10.07
CQ-8c8b (1.00 b/FPN)	6.55	7.25
CQ-8c10b (1.25 b/FPN)	5.47	5.90

1-bit perplexity and accuracy

WikiText-2 perplexity

Method	LLaMA-2-13B	Mistral-7B
FP16	4.57	4.76
KVQuant-1b	4709.83	203.73
KVQuant-1b-1%	13.76	10.07
CQ-8c8b (1.00 b/FPN)	6.55	7.25
CQ-8c10b (1.25 b/FPN)	5.47	5.90

1-bit accuracy (LLaMA-2-13B)

Method	WinoGrande	PIQA	ARC-C
FP16	71.98	79.16	48.29
KVQuant-1b-1%	57.30	70.89	32.59
CQ-8c8b	57.14	73.01	34.30
CQ-8c10b	62.98	74.37	38.74

1-bit perplexity and accuracy

WikiText-2 perplexity

Method	LLaMA-2-13B	Mistral-7B
FP16	4.57	4.76
KVQuant-1b	4709.83	203.73
KVQuant-1b-1%	13.76	10.07
CQ-8c8b (1.00 b/FPN)	6.55	7.25
CQ-8c10b (1.25 b/FPN)	5.47	5.90

1-bit accuracy (LLaMA-2-13B)

Method	WinoGrande	PIQA	ARC-C
FP16	71.98	79.16	48.29
KVQuant-1b-1%	57.30	70.89	32.59
CQ-8c8b	57.14	73.01	34.30
CQ-8c10b	62.98	74.37	38.74

Ablation result

At 2 bits, increasing the number of coupled channels and using Fisher-guided centroids both improve quality. For LLaMA-2-13B, perplexity decreases from approximately 890 (1 channel, no Fisher) to 4.81 (4 channels, Fisher).

Complete WikiText-2 perplexity results

	Bits Per FPN	LLaMA-7b	LLaMA-13b	LLaMA-2-7b	LLaMA-2-13b	Mistral-7b
FP16	16	5.68	5.09	5.12	4.57	4.76
INT4	4.00-4.01	5.98	5.32	5.66	5.01	4.97
INT4-gs128	4.16	5.77	5.16	5.32	4.71	4.82
NF4	4.00-4.01	5.87	5.23	5.47	4.90	4.91
NF4-gs128	4.16	5.77	5.17	5.30	4.71	4.83
KVQuant-4b	4.00-4.02	<u>5.73</u>	<u>5.15</u>	<u>5.18</u>	<u>4.63</u>	4.81
KVQuant-4b-1%	4.32-4.35	5.70	5.11	5.14	4.59	4.78
CQ-2c8b	4.00	5.70	5.11	5.14	4.59	<u>4.79</u>
INT2	2.00-2.01	11779	69965	4708	3942	573
INT2-gs128	2.14	37.37	41.77	117.88	93.09	51.96
NF2	2.00-2.02	3210.5	5785.6	13601	4035.6	902.51
NF2-gs128	2.14	351.23	141.19	634.59	642.44	252.85
KVQuant-2b	2.00-2.02	8.17	7.29	9.75	29.25	7.33
KVQuant-2b-1%	2.32-2.35	6.06	5.40	5.50	4.92	5.16
CQ-4c8b	2.00	5.97	5.32	5.42	4.81	5.11
KVQuant-1b	1.00-1.02	321.58	1617.40	NaN	4709.83	203.73
KVQuant-1b-1%	1.32-1.35	9.93	7.97	9.50	13.76	10.07
CQ-8c8b	1.00	<u>8.09</u>	<u>7.02</u>	<u>7.75</u>	<u>6.55</u>	<u>7.25</u>
CQ-8c10b	1.25	6.78	6.00	6.25	5.47	5.90

Result pattern. CQ is competitive at 4 bits, stronger than dense scalar baselines at 2 bits, and far more stable in the 1-bit regime.

CQ-8c10b (1.25 bits/FPN) approaches FP16 perplexity most closely.

Complete WikiText-2 perplexity results

Bits Per FPN		LLaMA-7b	LLaMA-13b	LLaMA-2-7b	LLaMA-2-13b	Mistral-7b
FP16	16	5.68	5.09	5.12	4.57	4.76
INT4	4.00-4.01	5.98	5.32	5.66	5.01	4.97
INT4-gs128	4.16	5.77	5.16	5.32	4.71	4.82
NF4	4.00-4.01	5.87	5.23	5.47	4.90	4.91
NF4-gs128	4.16	5.77	5.17	5.30	4.71	4.83
KVQuant-4b	4.00-4.02	<u>5.73</u>	<u>5.15</u>	<u>5.18</u>	<u>4.63</u>	4.81
KVQuant-4b-1%	4.32-4.35	5.70	5.11	5.14	4.59	4.78
CQ-2c8b	4.00	5.70	5.11	5.14	4.59	<u>4.79</u>
INT2	2.00-2.01	11779	69065	4708	3942	573
INT2-gs128	2.14	37.37	41.77	117.88	93.09	51.96
NF2	2.00-2.02	3210.5	5785.6	13601	4035.6	902.51
NF2-gs128	2.14	351.23	141.19	634.59	642.44	252.85
KVQuant-2b	2.00-2.02	8.17	7.29	9.75	29.25	7.33
KVQuant-2b-1%	2.32-2.35	6.06	5.40	5.50	4.92	5.16
CQ-4c8b	2.00	5.97	5.32	5.42	4.81	5.11
KVQuant-1b	1.00-1.02	321.58	1617.40	NaN	4709.83	203.73
KVQuant-1b-1%	1.32-1.35	9.93	7.97	9.50	13.76	10.07
CQ-8c8b	1.00	<u>8.09</u>	<u>7.02</u>	<u>7.75</u>	<u>6.55</u>	<u>7.25</u>
CQ-8c10b	1.25	6.78	6.00	6.25	5.47	5.90

Result pattern. CQ is competitive at 4 bits, stronger than dense scalar baselines at 2 bits, and far more stable in the 1-bit regime.

CQ-8c10b (1.25 bits/FPN) approaches FP16 perplexity most closely.

Complete WikiText-2 perplexity results

	Bits Per FPN	LLaMA-7b	LLaMA-13b	LLaMA-2-7b	LLaMA-2-13b	Mistral-7b
FP16	16	5.68	5.09	5.12	4.57	4.76
INT4	4.00-4.01	5.98	5.32	5.66	5.01	4.97
INT4-gs128	4.16	5.77	5.16	5.32	4.71	4.82
NF4	4.00-4.01	5.87	5.23	5.47	4.90	4.91
NF4-gs128	4.16	5.77	5.17	5.30	4.71	4.83
KVQuant-4b	4.00-4.02	5.73	5.15	5.18	4.63	4.81
KVQuant-4b-1%	4.32-4.35	5.70	5.11	5.14	4.59	4.78
CQ-2c8b	4.00	5.70	5.11	5.14	4.59	4.79
INT2	2.00-2.01	11779	69965	4708	3942	573
INT2-gs128	2.14	37.37	41.77	117.88	93.09	51.96
NF2	2.00-2.02	3210.5	5785.6	13601	4035.6	902.51
NF2-gs128	2.14	351.23	141.19	634.59	642.44	252.85
KVQuant-2b	2.00-2.02	8.17	7.29	9.75	29.25	7.33
KVQuant-2b-1%	2.32-2.35	6.06	5.40	5.50	4.92	5.16
CQ-4c8b	2.00	5.97	5.32	5.42	4.81	5.11
KVQuant-1b	1.00-1.02	321.58	1617.40	NaN	4709.83	203.73
KVQuant-1b-1%	1.32-1.35	9.93	7.97	9.50	13.76	10.07
CQ-8c8b	1.00	8.09	7.02	7.75	6.55	7.25
CQ-8c10b	1.25	6.78	6.00	6.25	5.47	5.90

Result pattern. CQ is competitive at 4 bits, stronger than dense scalar baselines at 2 bits, and far more stable in the 1-bit regime.

CQ-8c10b (1.25 bits/FPN) approaches FP16 perplexity most closely.

Complete C4 perplexity results

Bits Per FPN		LLaMA-7b	LLaMA-13b	LLaMA-2-7b	LLaMA-2-13b	Mistral-7b
FP16	16	7.08	6.61	6.63	6.05	5.71
INT4	4.00-4.01	7.40	6.82	7.31	6.59	5.91
INT4-gs128	4.16	7.16	6.67	6.87	6.20	5.76
NF4	4.00-4.01	7.27	6.74	7.09	6.45	5.85
NF4-gs128	4.16	7.16	6.66	6.86	6.20	5.77
KVQuant-4b	4.00-4.02	7.13	6.65	6.70	6.11	5.75
KVQuant-4b-1%	4.32-4.35	7.09	6.62	6.65	6.06	5.72
CQ-2c8b	4.00	<u>7.11</u>	<u>6.64</u>	<u>6.67</u>	<u>6.09</u>	<u>5.74</u>
INT2	2.00-2.01	10892	100870	4708	4220	477
INT2-gs128	2.14	43.49	56.25	113.49	97.04	50.73
NF2	2.00-2.02	2850.1	4680.3	13081.2	4175.6	1102.3
NF2-gs128	2.14	248.32	118.18	420.05	499.82	191.73
KVQuant-2b	2.00-2.02	10.28	9.05	15.16	43.77	8.40
KVQuant-2b-1%	2.32-2.35	7.38	6.83	7.06	6.38	6.08
CQ-4c8b	2.00	<u>7.52</u>	<u>6.96</u>	<u>7.23</u>	<u>6.52</u>	<u>6.17</u>
KVQuant-1b	1.00-1.02	168.90	1316.41	362.94	4223.37	127.07
KVQuant-1b-1%	1.32-1.35	<u>11.18</u>	<u>9.56</u>	16.04	22.87	10.53
CQ-8c8b	1.00	12.13	10.53	<u>12.49</u>	<u>10.53</u>	<u>9.89</u>
CQ-8c10b	1.25	9.12	8.23	9.03	8.01	7.46

Result pattern. C4 reproduces the same trend: the benefit of coupled coding grows as bit width decreases, with the largest gap at 1–2 bits.

Complete C4 perplexity results

	Bits Per FPN	LLaMA-7b	LLaMA-13b	LLaMA-2-7b	LLaMA-2-13b	Mistral-7b
FP16	16	7.08	6.61	6.63	6.05	5.71
INT4	4.00-4.01	7.40	6.82	7.31	6.59	5.91
INT4-gs128	4.16	7.16	6.67	6.87	6.20	5.76
NF4	4.00-4.01	7.27	6.74	7.09	6.45	5.85
NF4-gs128	4.16	7.16	6.66	6.86	6.20	5.77
KVQuant-4b	4.00-4.02	7.13	6.65	6.70	6.11	5.75
KVQuant-4b-1%	4.32-4.35	7.09	6.62	6.65	6.06	5.72
CQ-2c8b	4.00	<u>7.11</u>	<u>6.64</u>	<u>6.67</u>	<u>6.09</u>	<u>5.74</u>
INT2	2.00-2.01	10892	100870	4708	4220	477
INT2-gs128	2.14	43.49	56.25	113.49	97.04	50.73
NF2	2.00-2.02	2850.1	4680.3	13081.2	4175.6	1102.3
NF2-gs128	2.14	248.32	118.18	420.05	499.82	191.73
KVQuant-2b	2.00-2.02	10.28	9.05	15.16	43.77	8.40
KVQuant-2b-1%	2.32-2.35	7.38	6.83	7.06	6.38	6.08
CQ-4c8b	2.00	<u>7.52</u>	<u>6.96</u>	<u>7.23</u>	<u>6.52</u>	<u>6.17</u>
KVQuant-1b	1.00-1.02	168.90	1316.41	362.94	4223.37	127.07
KVQuant-1b-1%	1.32-1.35	<u>11.18</u>	<u>9.56</u>	16.04	22.87	10.53
CQ-8c8b	1.00	12.13	10.53	12.49	10.53	9.89
CQ-8c10b	1.25	9.12	8.23	9.03	8.01	7.46

Result pattern. C4 reproduces the same trend: the benefit of coupled coding grows as bit width decreases, with the largest gap at 1–2 bits.

Zero-shot accuracy (4-bit and 2-bit regimes)

Bits Per FPN		Task	LLaMA-7b	LLaMA-13b	LLaMA-2-7b	LLaMA-2-13b	Mistral-7b
FP16	16	WinoGrande	69.93	72.69	68.90	71.98	73.88
		PIQA	78.67	79.16	78.07	79.16	80.58
		ARC Challenge	41.72	46.42	43.43	48.29	50.34
KVQuant-4b	4.00-4.02	WinoGrande	69.53	72.61	67.96	71.59	73.88
		PIQA	78.62	79.22	77.86	78.94	80.58
		ARC Challenge	42.32	<u>45.99</u>	42.75	46.67	49.06
KVQuant-4b-1%	4.32-4.35	WinoGrande	70.72	73.40	68.67	<u>72.30</u>	<u>73.72</u>
		PIQA	78.40	<u>79.16</u>	78.07	79.27	80.74
		ARC Challenge	41.38	46.76	43.17	47.87	49.91
CQ-2c8b	4.00	WinoGrande	70.40	72.45	68.27	72.53	73.48
		PIQA	<u>78.61</u>	79.11	<u>77.91</u>	78.62	80.52
		ARC Challenge	<u>41.55</u>	<u>45.99</u>	43.34	<u>47.78</u>	<u>49.15</u>
KVQuant-2b	2.00-2.02	WinoGrande	53.59	59.35	51.70	51.30	63.46
		PIQA	72.47	74.81	63.38	65.40	75.46
		ARC Challenge	32.00	34.47	22.44	24.66	38.57
KVQuant-2b-1%	2.32-2.35	WinoGrande	68.03	71.43	67.64	70.17	70.80
		PIQA	77.69	78.51	76.60	78.51	79.65
		ARC Challenge	38.74	45.14	41.47	44.97	47.53
CQ-4c8b	2.00	WinoGrande	67.48	70.72	66.45	69.06	69.38
		PIQA	76.11	78.29	76.12	77.42	79.49
		ARC Challenge	38.48	44.03	39.93	44.11	45.65

Result pattern. At 4 bits, CQ is essentially indistinguishable from strong baselines. At 2 bits, it is much stronger than dense KVQuant-2b and remains competitive with KVQuant-2b-1%.

Zero-shot accuracy (4-bit and 2-bit regimes)

Bits Per FPN		Task	LLaMA-7b	LLaMA-13b	LLaMA-2-7b	LLaMA-2-13b	Mistral-7b
FP16	16	WinoGrande	69.93	72.69	68.90	71.98	73.88
		PIQA	78.67	79.16	78.07	79.16	80.58
		ARC Challenge	44.72	46.42	43.43	48.39	50.34
KVQuant-4b	4.00-4.02	WinoGrande	69.53	72.61	67.96	71.59	73.88
		PIQA	78.62	79.22	77.86	78.94	80.58
		ARC Challenge	42.32	45.99	42.75	46.67	49.06
KVQuant-4b-1%	4.32-4.35	WinoGrande	70.72	73.40	68.67	72.30	73.72
		PIQA	78.40	79.16	78.07	79.27	80.74
		ARC Challenge	41.38	46.76	43.17	47.87	49.91
CQ-2c8b	4.00	WinoGrande	70.40	72.45	68.27	72.53	73.48
		PIQA	78.61	79.11	77.91	78.62	80.52
		ARC Challenge	41.55	45.99	48.34	47.78	49.15
KVQuant-2b	2.00-2.02	WinoGrande	53.59	59.35	51.70	51.30	63.46
		PIQA	72.47	74.81	63.38	65.40	75.46
		ARC Challenge	32.00	34.47	22.44	24.66	38.57
KVQuant-2b-1%	2.32-2.35	WinoGrande	68.03	71.43	67.64	70.17	70.80
		PIQA	77.69	78.51	76.60	78.51	79.65
		ARC Challenge	38.74	45.14	41.47	44.97	47.53
CQ-4c8b	2.00	WinoGrande	67.48	70.72	66.45	69.06	69.38
		PIQA	76.11	78.29	76.12	77.42	79.49
		ARC Challenge	38.48	44.03	39.93	44.11	45.65

Result pattern. At 4 bits, CQ is essentially indistinguishable from strong baselines. At 2 bits, it is much stronger than dense KVQuant-2b and remains competitive with KVQuant-2b-1%.

Zero-shot accuracy (4-bit and 2-bit regimes)

Bits Per FPN		Task	LLaMA-7b	LLaMA-13b	LLaMA-2-7b	LLaMA-2-13b	Mistral-7b
FP16	16	WinoGrande	69.93	72.69	68.90	71.98	73.88
		PIQA	78.67	79.16	78.07	79.16	80.58
		ARC Challenge	41.72	46.42	43.43	48.20	50.34
KVQuant-4b	4.00-4.02	WinoGrande	69.53	72.61	67.96	71.59	73.88
		PIQA	78.62	79.22	77.86	78.94	80.58
		ARC Challenge	42.32	45.92	42.75	46.67	49.06
KVQuant-4b-1%	4.32-4.35	WinoGrande	70.72	73.40	68.67	72.30	73.72
		PIQA	78.40	79.16	78.07	79.27	80.74
		ARC Challenge	41.38	46.76	43.17	47.87	49.91
CQ-2c8b	4.00	WinoGrande	70.40	72.45	68.27	72.53	73.48
		PIQA	78.61	79.11	77.91	78.62	80.52
		ARC Challenge	41.55	45.92	43.34	47.78	49.15
KVQuant-2b	2.00-2.02	WinoGrande	53.59	59.35	51.70	51.30	63.46
		PIQA	72.47	74.81	63.38	65.40	75.46
		ARC Challenge	32.00	34.47	22.44	24.66	38.57
KVQuant-2b-1%	2.32-2.35	WinoGrande	68.03	71.43	67.64	70.17	70.80
		PIQA	77.69	78.51	76.60	78.51	79.65
		ARC Challenge	38.74	45.14	41.47	44.97	47.53
CQ-4c8b	2.00	WinoGrande	67.48	70.72	66.45	69.06	69.38
		PIQA	76.11	78.29	76.12	77.42	79.49
		ARC Challenge	38.48	44.03	39.93	44.11	45.65

Result pattern. At 4 bits, CQ is essentially indistinguishable from strong baselines. At 2 bits, it is much stronger than dense KVQuant-2b and remains competitive with KVQuant-2b-1%.

Zero-shot accuracy (1-bit regime)

Bits Per FPN		Task	LLaMA-7b	LLaMA-13b	LLaMA-2-7b	LLaMA-2-13b	Mistral-7b
CQ-4c8b	2.00	WinoGrande	<u>67.48</u>	<u>70.72</u>	<u>66.45</u>	<u>69.06</u>	<u>69.38</u>
		PIQA	<u>76.11</u>	<u>78.29</u>	<u>76.12</u>	<u>77.42</u>	<u>79.49</u>
		ARC Challenge	<u>38.48</u>	<u>44.03</u>	<u>39.93</u>	<u>44.11</u>	<u>45.65</u>
KVQuant-1b	1.00-1.02	WinoGrande	50.51	48.46	50.91	49.41	49.80
		PIQA	53.26	53.54	53.37	50.92	54.73
		ARC Challenge	21.76	21.33	20.65	21.67	19.88
KVQuant-1b-1%	1.32-1.35	WinoGrande	<u>56.67</u>	61.01	<u>57.77</u>	<u>57.30</u>	58.17
		PIQA	<u>71.38</u>	<u>75.46</u>	69.91	70.89	73.83
		ARC Challenge	29.69	<u>35.32</u>	<u>31.48</u>	32.59	33.19
CQ-8c8b	1.00	WinoGrande	56.51	<u>61.56</u>	55.01	57.14	<u>58.25</u>
		PIQA	71.16	73.99	<u>71.22</u>	<u>73.01</u>	<u>75.24</u>
		ARC Challenge	<u>30.20</u>	33.79	30.20	<u>34.30</u>	<u>33.79</u>
CQ-8c10b	1.25	WinoGrande	60.46	65.27	59.19	62.98	63.93
		PIQA	73.45	75.90	73.07	74.37	77.31
		ARC Challenge	33.28	37.12	34.64	38.74	39.59

Dense 1-bit KVQuant collapses on all three tasks. CQ-8c8b substantially improves accuracy at exactly 1 bit/FPN; CQ-8c10b at 1.25 bits provides the strongest quality among the extreme low-bit configurations.

Zero-shot accuracy (1-bit regime)

Bits Per FPN		Task	LLaMA-7b	LLaMA-13b	LLaMA-2-7b	LLaMA-2-13b	Mistral-7b
CQ-4c8b	2.00	WinoGrande	<u>67.48</u>	<u>70.72</u>	<u>66.45</u>	<u>69.06</u>	<u>69.38</u>
		PIQA	<u>76.11</u>	<u>78.29</u>	<u>76.12</u>	<u>77.42</u>	<u>79.49</u>
		ARC Challenge	<u>38.48</u>	<u>44.03</u>	<u>39.93</u>	<u>44.11</u>	<u>45.65</u>
KVQuant-1b	1.00-1.02	WinoGrande	50.51	48.46	50.91	49.41	49.80
		PIQA	53.26	53.54	53.37	50.92	54.73
		ARC Challenge	21.76	21.33	20.65	21.67	19.88
KVQuant-1b-1%	1.32-1.35	WinoGrande	<u>56.67</u>	61.01	<u>57.77</u>	<u>57.30</u>	58.17
		PIQA	<u>71.38</u>	<u>75.46</u>	69.91	70.89	73.83
		ARC Challenge	29.69	<u>35.32</u>	<u>31.48</u>	32.59	33.19
CQ-8c8b	1.00	WinoGrande	56.51	61.56	55.01	57.14	<u>58.25</u>
		PIQA	71.16	73.99	<u>71.22</u>	<u>73.01</u>	<u>75.24</u>
		ARC Challenge	30.20	33.79	30.20	<u>34.30</u>	<u>33.79</u>
CQ-8c10b	1.25	WinoGrande	60.46	65.27	59.19	62.98	63.93
		PIQA	73.45	75.90	73.07	74.37	77.31
		ARC Challenge	33.28	37.12	34.64	38.74	39.59

Dense 1-bit KVQuant collapses on all three tasks. CQ-8c8b substantially improves accuracy at exactly 1 bit/FPN; CQ-8c10b at 1.25 bits provides the strongest quality among the extreme low-bit configurations.

Zero-shot accuracy (1-bit regime)

Bits Per FPN		Task	LLaMA-7b	LLaMA-13b	LLaMA-2-7b	LLaMA-2-13b	Mistral-7b
CQ-4c8b	2.00	WinoGrande	<u>67.48</u>	<u>70.72</u>	<u>66.45</u>	<u>69.06</u>	<u>69.38</u>
		PIQA	<u>76.11</u>	<u>78.29</u>	<u>76.12</u>	<u>77.42</u>	<u>79.49</u>
		ARC Challenge	<u>38.48</u>	<u>44.03</u>	<u>39.93</u>	<u>44.11</u>	<u>45.65</u>
KVQuant-1b	1.00-1.02	WinoGrande	50.51	48.46	50.91	49.41	49.80
		PIQA	53.26	53.54	53.37	50.92	54.73
		ARC Challenge	21.76	21.33	20.65	21.67	19.88
KVQuant-1b-1%	1.32-1.35	WinoGrande	<u>56.67</u>	61.01	<u>57.77</u>	<u>57.30</u>	58.17
		PIQA	<u>71.38</u>	<u>75.46</u>	69.91	70.89	73.83
		ARC Challenge	29.69	<u>35.32</u>	<u>31.48</u>	32.59	33.19
CQ-8c8b	1.00	WinoGrande	56.51	61.56	55.01	57.14	58.25
		PIQA	71.16	73.99	<u>71.22</u>	<u>73.01</u>	<u>75.24</u>
		ARC Challenge	30.20	33.79	30.20	<u>34.30</u>	<u>33.79</u>
CQ-8c10b	1.25	WinoGrande	60.46	65.27	59.19	62.98	63.93
		PIQA	73.45	75.90	73.07	74.37	77.31
		ARC Challenge	33.28	37.12	34.64	38.74	39.59

Dense 1-bit KVQuant collapses on all three tasks. CQ-8c8b substantially improves accuracy at exactly 1 bit/FPN; CQ-8c10b at 1.25 bits provides the strongest quality among the extreme low-bit configurations.

Comparison of the two approaches

KIVI / CQ

Aspect	KIVI	Coupled Quantization (CQ)
Research objective	Identify the appropriate quantization dimension for keys and values at 2 bits.	Exploit inter-channel dependence to preserve quality at 1–1.25 bits/FPN.
Primary insight	Key cache contains persistent channel outliers; value errors must be assessed after attention mixing.	Joint entropy grows more slowly than the sum of marginal entropies; channels are statistically dependent.
Representation	Grouped scalar quantization: keys per-channel, values per-token.	Vector quantization of coupled channels with learned multi-channel centroids.
Additional mechanism	Recent residual KV window retained in full precision.	Fisher-guided weighted centroid learning; optional full-precision sliding window.
Effective regime	Approximately 2-bit KV cache with small quality loss on evaluated LLaMA/Mistral models.	1-bit and 1.25-bit KV cache with substantially improved quality over prior low-bit baselines.
System implementation	Fused dequantization/matmul and increased feasible batch size.	Avoids sparse outlier path used by dense-and-sparse baselines, at the cost of offline codebook learning.

- ▶ **Motivation:** KV-cache storage and memory traffic are major constraints in long-context and high-batch autoregressive inference.

- ▶ **Motivation:** KV-cache storage and memory traffic are major constraints in long-context and high-batch autoregressive inference.
- ▶ **KIVI:** very-low-bit quantization must respect the distinct error structure of keys and values. The effective 2-bit configuration uses per-channel keys, per-token values, and a small full-precision residual window.

- ▶ **Motivation:** KV-cache storage and memory traffic are major constraints in long-context and high-batch autoregressive inference.
- ▶ **KIVI:** very-low-bit quantization must respect the distinct error structure of keys and values. The effective 2-bit configuration uses per-channel keys, per-token values, and a small full-precision residual window.
- ▶ **Coupled Quantization:** independent scalar coding leaves substantial inter-channel redundancy unused. Joint vector coding exploits this dependency and extends practical compression to approximately 1 bit/FPN.

- ▶ **Motivation:** KV-cache storage and memory traffic are major constraints in long-context and high-batch autoregressive inference.
- ▶ **KIVI:** very-low-bit quantization must respect the distinct error structure of keys and values. The effective 2-bit configuration uses per-channel keys, per-token values, and a small full-precision residual window.
- ▶ **Coupled Quantization:** independent scalar coding leaves substantial inter-channel redundancy unused. Joint vector coding exploits this dependency and extends practical compression to approximately 1 bit/FPN.
- ▶ **Overall implication:** successful extreme KV-cache quantization depends on preserving statistical structure at two levels:
 - across axes: token versus channel granularity (KIVI);
 - across channels: joint dependence and learned vector codebooks (CQ).