Stabilizing Quantized KV Caches via Subspace Embeddings and Manifold Interpolation

Indrajeet Roy [1]

Abstract

Large Language Models (LLMs) function as autoregressive predictors by repeatedly applying linear transformations and inner products between a current state vector and a history of projected vectors (10). To avoid the $O(T^2)$ cost of recomputing these linear maps at every step, modern implementations store the history in a matrix structure known as the *Key–Value (KV) cache* (6). While this strategy resolves the computational bottleneck, it introduces a memory constraint that necessitates aggressive quantization, replacing continuous vectors in $\mathbb{R}^d$ with low-precision discrete representations (7; 4).

This quantization alters the algebraic structure of the vector space. Specifically, it introduces a metric mismatch: small perturbations in the storage space (measured by Hamming distance) do not correspond to bounded perturbations in the inner-product space (measured by Euclidean distance) (1). Instead, bit-level errors induce large, sparse perturbation vectors that can distort the geometry of the attention mechanism and destabilize the projection.

This paper proposes a linear-algebraic framework for stabilizing such quantized computations using classical error-correcting codes (5). Data integrity is modeled over the finite field $\mathbb{F}_2$, where discrete vectors are embedded into higher-dimensional subspaces defined by the generator matrices of Hamming and Golay codes (8). The null-space properties of the corresponding parity-check matrices are exploited to detect and correct structured perturbations before the vectors are lifted back to $\mathbb{R}^d$. Complementing this algebraic protection, a geometric stabilization strategy is introduced: interpolation-based reconstruction of detected erasures, which exploits the local smoothness of the state vector sequence. Experiments on large-scale matrices derived from GPT-2 and LLaMA-3.1-8B (9; 2) demonstrate that these subspace embeddings restore numerical stability, yielding performance comparable to high-precision representations.

[1]Mathematics Department, College of Sciences, Northeastern University, Boston, MA, USA. Correspondence to: Indrajeet Roy <roy.i@northeastern.edu>.

1. Introduction

Large Language Models (LLMs) can be seen as high-dimensional dynamical systems governed by linear algebraic operations. These systems function as *autoregressive* predictors: they generate sequences recursively, where the output of step t becomes a basis vector for the computation at step $t + 1$ (9).

From a mathematical standpoint, this recursion creates a distinct complexity challenge. At every time step, the model must evaluate inner products between a current state vector and the entire history of previous state vectors. Naively recomputing these inner products at every step implies a complexity of $\mathcal{O}(T^2)$ for a sequence of length T.

To resolve this, implementations employ a matrix factorization strategy known as the *Key–Value (KV) cache* (6). Rather than re-evaluating the linear operators on the entire history, the system stores the projected history vectors in two growing matrices, $K, V \in \mathbb{R}^{T\times d}$. This reduces the current step to a rank-1 update and a matrix-vector product, lowering complexity to $\mathcal{O}(T)$. However, as T grows ($T \to 10^5$), the memory requirement to store these matrices in high-precision floating point becomes intractable (7).

The standard solution is **quantization**: a discretization map that projects continuous vectors in $\mathbb{R}^d$ onto low-precision integer lattices (e.g., mapping $\mathbb{R}^d \to \mathbb{Z}_4^d$) (3). While this compression makes storage feasible, it degrades the geometric properties required for stable linear algebra.

In high-precision arithmetic, quantization error is typically modeled as small-magnitude additive noise ($\epsilon \sim \mathcal{N}(0, \sigma^2)$), which averages out in high-dimensional dot products. In the regime of 4-bit integers, however, this assumption fails. The storage medium is susceptible to bit-level perturbations. Because of the weighted binary representation, a single

bit-flip does not act as Gaussian noise; it acts as a sparse, large-magnitude impulse in the vector space (1). When such a corrupted vector is used in an inner product, it distorts the geometry of the attention mechanism, causing the probability distribution to collapse.

This paper addresses this instability not as a hardware reliability issue, but as a problem of linear algebra over finite fields. The core premise is that while the matrix operations rely on real-valued inner products, the integrity of the stored entries is best preserved by viewing them as elements of the vector space $\mathbb{F}_2^n$. This separation motivates the application of classical Error-Correcting Codes (ECC) as an algebraic protection layer (8).

This work proposes a framework where quantized vectors are projected into higher-dimensional sparse subspaces defined by the generator matrices of Hamming and Golay codes. By utilizing the kernel (null-space) of their corresponding parity-check matrices, structural perturbations can be geometrically isolated and corrected before they propagate into the real-valued calculation. Furthermore, a manifold-aware stabilization technique is introduced, exploiting the local smoothness of the sequence of state vectors to interpolate uncorrectable errors, effectively bridging the gap between discrete coding theory and continuous vector calculus.

2. Background

2.1. The Linear Algebra of Attention

Large Language Models (LLMs) are composed of many *Transformer layers* stacked together. While the full model is deep and highly non-linear, each layer applies the same underlying sequence of linear algebraic operations. While the full model architecture is non-linear, the computational core of the transformer is the *Attention Mechanism* (10). From a linear algebraic perspective, this mechanism acts as a data-dependent aggregation operator that computes a new state vector as a convex combination of historical vectors.

Let the input sequence at step t be represented by a vector $\mathbf{x} \in \mathbb{R}^d$. The layer applies three parallel linear transformations parameterized by the matrices $W_Q, W_K, W_V \in \mathbb{R}^{d\times d}$. These projections map the input into three distinct subspaces:

$$\mathbf{q} = \mathbf{x}W_Q, \quad \mathbf{k} = \mathbf{x}W_K, \quad \mathbf{v} = \mathbf{x}W_V.$$

These vectors correspond to the standard terminology of *Query*, *Key*, and *Value*.

2.1.1. The Affine Transformation of History

The system maintains a history of the sequence through step t. Let $K \in \mathbb{R}^{t\times d}$ and $V \in \mathbb{R}^{t\times d}$ be matrices where the i-th rows correspond to the projected key and value vectors of the i-th step in the sequence.

The attention mechanism first computes an vector $\mathbf{s} \in \mathbb{R}^t$ by evaluating the inner product between the current query vector $\mathbf{q}$ and the rows of the history matrix K: $\mathbf{s} = \mathbf{q}K^\top$.

Geometrically, this operation projects the current state $\mathbf{q}$ onto the subspace spanned by the history K. The resulting scalar components $s_i = \langle \mathbf{q}, \mathbf{k}_i \rangle$ measure the alignment (colinearity) between the current state and previous states.

2.1.2. Projection onto the Normalized Probability Space

The vector $\mathbf{s}$ is not suitable for averaging. It is mapped to the probability space Δ^{t-1} via the *Softmax* function $\sigma : \mathbb{R}^t \to \Delta^{t-1}$, defined component-wise as: $\boldsymbol{\alpha} = \sigma(\mathbf{s}), \quad \text{where } \alpha_i = \frac{e^{s_i}}{\sum_{j=1}^t e^{s_j}}$. This non-linear normalization ensures that $\sum \alpha_i = 1$ and $\alpha_i > 0$.

2.1.3. Output as a Convex Combination

The final output vector $\mathbf{z} \in \mathbb{R}^d$ is computed as the weighted sum of the rows of V, using the coefficients $\boldsymbol{\alpha}$: $\mathbf{z} = \boldsymbol{\alpha}^\top V = \sum_{i=1}^t \alpha_i \mathbf{v}_i$. Algebraically, this operation extracts information from the subspace spanned by V. If the query $\mathbf{q}$ is orthogonal to a key $\mathbf{k}_i$, the corresponding weight $\alpha_i \to 0$, and the value $\mathbf{v}_i$ contributes negligibly to the sum. Conversely, high alignment results in a large contribution (10).

The vulnerability addressed in this paper arises in the storage of the matrices K and V. In the standard implementation, these matrices are not stored in high-precision $\mathbb{R}$. Instead, they are discretized. A perturbation to an entry in K (caused by bit errors) alters the geometry of the inner product $\mathbf{q}K^\top$, potentially redistributing the probability mass of $\boldsymbol{\alpha}$ toward an incorrect index, resulting in the selection of an unintended basis vector from V (11).

2.2. Recursive Matrix Formulation (The KV Cache)

Sequence Length (T)	Arithmetic Complexity (FLOPs $\propto T^2$)	Matrix Storage (Size $\propto T \cdot d$)
1,000	10^6	0.5 GB
10,000	10^8	5.0 GB
100,000	10^{10}	50.0 GB
128,000	1.6×10^{10}	64.0 GB

Table 1. **Computational vs. Storage Complexity.** Direct evaluation implies quadratic arithmetic complexity ($\mathcal{O}(T^2)$). Recursive storage reduces arithmetic cost to linear ($\mathcal{O}(T)$) but shifts the constraint to storage capacity ($\mathcal{O}(T \cdot d)$), creating a memory bottleneck for large T.

In a direct implementation of the autoregressive process, the vector $\mathbf{s}$ would be computed by projecting the entire history sequence at every time step t. This results in a cumulative arithmetic complexity of $\sum_{t=1}^{T} t = \mathcal{O}(T^2)$, which is intractable for long sequences.

To resolve this, the linear operators are reformulated recursively. Since the past state vectors are static, the system does not need to recompute the projections K_{t-1} and V_{t-1}. Instead, at step t, the system computes only the new basis vectors $\mathbf{k}_t$ and $\mathbf{v}_t$, and appends them to the stored matrices via a row-wise concatenation:

$$K_t = \begin{bmatrix} K_{t-1} \\ \mathbf{k}_t \end{bmatrix}, \qquad V_t = \begin{bmatrix} V_{t-1} \\ \mathbf{v}_t \end{bmatrix}.$$

This technique, known in the literature as the **Key–Value (KV) cache** (6), reduces the arithmetic complexity of the current step to $\mathcal{O}(T)$ (a single matrix-vector product).

However, as illustrated in Table 2.2, this optimization transforms the bottleneck from arithmetic operations to storage capacity. The matrices K and V grow linearly. For large context windows (e.g., $T = 10^5$), the storage requirement exceeds the capacity of standard high-bandwidth memory buffers. This requires the use of **quantization**: a projection $\Pi : \mathbb{R} \to \mathbb{Z}_q$ that maps the real-valued entries of K and V onto a discrete finite set to reduce the bit-width of the stored matrices (4).

2.3. Algebraic Instability of Discretized Storage

Memory constraints require values to be discretized through a mapping $\Pi : \mathbb{R} \to \mathcal{S}$, where $\mathcal{S} \subset \mathbb{Z}$ is a finite set of integers. As a result, the attention computation operates in a mixed algebraic setting: the query vector $\mathbf{q} \in \mathbb{R}^d$ remains continuous, while the stored matrices K and V are discrete approximations.

This discretization introduces two distinct sources of error:

1. **Quantization Error:** A bounded, deterministic error $\|\mathbf{x} - \Pi(\mathbf{x})\|$, commonly modeled as uniform additive noise.

2. **Storage Corruption:** Random perturbations caused by physical noise in the memory hardware.

While quantization error is bounded and typically well behaved, storage corruption poses a more serious linear algebraic challenge. The storage channel is modeled as a stochastic process in which the binary representation of a stored integer undergoes independent bit flips with probability p.

2.3.1. The Metric Mismatch

The instability arises from a mismatch between the error metrics of the storage space and the semantic vector space. The mapping from stored binary vectors in $\mathbb{F}_2^B$ to real-valued vectors in $\mathbb{R}^d$ is not continuous with respect to storage errors.

In the storage domain, errors are naturally measured using the *Hamming distance*, where a single bit flip has unit cost (8). However, when binary representations are interpreted as integers, bit positions carry different weights. A flip in the most significant bit produces a much larger numerical change than a flip in a least significant bit. As a result, a single-bit error can induce a perturbation $\Delta\mathbf{k}$ whose Euclidean norm is comparable to the full range of the quantization grid.

In contrast to floating-point arithmetic, where bit errors often lead to small relative changes, errors in low-precision integer representations manifest as **sparse, large-magnitude impulse noise**. This mismatch between discrete storage errors and continuous semantic distances is the root cause of instability in the attention computation (1).

2.3.2. Propagation through the Softmax Projection

This perturbation propagates directly into the attention computation. Let $\tilde{\mathbf{k}} = \mathbf{k} + \mathbf{e}_{\text{impulse}}$ denote a corrupted key vector. The resulting inner product becomes

$$\tilde{s} = \langle \mathbf{q}, \tilde{\mathbf{k}} \rangle = \langle \mathbf{q}, \mathbf{k} \rangle + \langle \mathbf{q}, \mathbf{e}_{\text{impulse}} \rangle.$$

Because the error vector $\mathbf{e}_{\text{impulse}}$ can have large magnitude, the second term may dominate the sum, overwhelming the contribution from the true key vector.

When these distorted scores are passed through the softmax projection, the exponential amplification magnifies the outlier. As a result, the probability mass concentrates on a single index, producing an approximately one-hot distribution (11). This causes the attention mechanism to ignore the valid history and instead place nearly all weight on the corrupted entry, destabilizing the subsequent computation.

3. Theoretical Framework

Having established that the metric mismatch between the Hamming and Euclidean spaces destabilizes the attention mechanism, this section formulates a protection strategy. The integrity of the recursive matrices K and V is framed as a linear algebra problem over the finite field of two elements, $\mathbb{F}_2$ (5).

3.1. Stochastic Error Model over $\mathbb{F}_2$

The storage interface is modeled as a *Binary Symmetric Channel*. Let the quantized integer values be represented as vectors in a finite-dimensional vector space $\mathbb{F}_2^k$.

To facilitate error correction, a linear map (encoding) $\phi : \mathbb{F}_2^k \to \mathbb{F}_2^n$ is defined where $n > k$. Let $\mathbf{c} \in \mathbb{F}_2^n$ be the stored codeword. The retrieval process is modeled as an transformation:

$$\mathbf{r} = \mathbf{c} \oplus \mathbf{e},$$

where $\oplus$ denotes vector addition modulo 2, and $\mathbf{e} \in \mathbb{F}_2^n$ is a stochastic error vector.

The components of the error vector $\mathbf{e} = [e_1, \ldots, e_n]$ are modeled as independent and identically distributed (i.i.d.) Bernoulli random variables with parameter p:

$$P(e_i = 1) = p, \quad P(e_i = 0) = 1 - p.$$

Here, p represents the transition probability of the channel. From an algebraic standpoint, p dictates the expected Hamming weight of the error vector, $E[w_H(\mathbf{e})] = n \cdot p$.

Table 2. **Stochastic Regimes.** Error probabilities used to model increasing perturbation density.

p	Frequency	Density
10^{-4}	1 in 10^4	Sparse
10^{-3}	1 in 10^3	Moderate
10^{-2}	1 in 10^2	Dense

As summarized in Table 2, system stability is analyzed under three distinct regimes of p. The regime $p = 10^{-2}$ is particularly significant from a theoretical perspective, as the high density of errors challenges the correction capacity of standard linear codes, requiring the use of the higher-dimensional subspace embeddings described in the following sections.

3.2. Subspace Projections via Linear Block Codes

The protection mechanism is based on restricting valid cached representations to a k-dimensional linear subspace $\mathcal{C} \subset \mathbb{F}_2^n$. By embedding data from $\mathbb{F}_2^k$ into a higher-dimensional space with $n > k$, redundancy is introduced, creating separation between valid vectors and corrupted ones (8).

This construction relies on two linear operators: a *generator* that maps data vectors into the code subspace, and a *parity-check operator* whose kernel characterizes the set of valid codewords.

3.2.1. The Generator Matrix (Image Construction)

Encoding is defined by an injective linear map $\phi : \mathbb{F}_2^k \to \mathbb{F}_2^n$ that embeds a data vector $\mathbf{d}$ into the code subspace $\mathcal{C}$. This map is represented by a *generator matrix* $G \in \mathbb{F}_2^{k \times n}$ with full row rank, such that

$$\mathcal{C} = \mathrm{Im}(G) = \{\mathbf{d}G \mid \mathbf{d} \in \mathbb{F}_2^k\}.$$

Encoding consists of computing the codeword $\mathbf{c} = \mathbf{d}G$.

In this work, *systematic* generator matrices of the form $G = [I_k \mid P]$ are used. This choice preserves the original data bits in the first k coordinates of the codeword, while the remaining $n - k$ coordinates store parity information that imposes linear constraints.

The effectiveness of the embedding depends on the *codimension* $n - k$. The code subspace $\mathcal{C}$ contains 2^k valid vectors, while the space $\mathbb{F}_2^n$ contains 2^n vectors. As $n - k$ increases, valid codewords become sparse, and small-weight error vectors $\mathbf{e}$ are likely to move $\mathbf{c}$ outside $\mathcal{C}$.

3.2.2. The Parity-Check Matrix (Kernel Characterization)

Rather than testing membership in $\mathcal{C}$ via the generator, it is more efficient to characterize $\mathcal{C}$ as the kernel of a linear operator. The *parity-check matrix* $H \in \mathbb{F}_2^{(n-k) \times n}$ is defined so that

$$\mathcal{C} = \ker(H) = \{\mathbf{x} \in \mathbb{F}_2^n \mid H\mathbf{x}^\top = \mathbf{0}\}.$$

Consistency between encoding and decoding requires that

$$GH^\top = \mathbf{0},$$

ensuring that all encoded codewords lie in the kernel of H.

3.2.3. Syndrome Decoding and Quotient Spaces

Suppose a stored codeword $\mathbf{c}$ is corrupted by an additive error $\mathbf{e}$, yielding the retrieved vector $\mathbf{r} = \mathbf{c} \oplus \mathbf{e}$. The error detection mechanism relies on the algebraic concept of the syndrome.

Definition 3.1 (Syndrome)**.** Given a parity-check matrix $H \in \mathbb{F}_2^{(n-k) \times n}$ and a received vector $\mathbf{r} \in \mathbb{F}_2^n$, the **syndrome** $\mathbf{z} \in \mathbb{F}_2^{n-k}$ is defined as:

$$\mathbf{z} = H\mathbf{r}^\top.$$

Since $\mathbf{c} \in \ker(H)$, linearity implies $\mathbf{z} = H(\mathbf{c} \oplus \mathbf{e})^\top = H\mathbf{e}^\top$. Thus, the syndrome depends only on the error vector $\mathbf{e}$, not the data.

The syndrome $\mathbf{z}$ identifies the coset of $\mathcal{C}$ in the quotient space $\mathbb{F}_2^n/\mathcal{C}$ that contains the error vector (8). Two cases arise:

1. If $\mathbf{z} = \mathbf{0}$, then $\mathbf{r} \in \mathcal{C}$ and the vector is assumed valid.

2. If $\mathbf{z} \neq \mathbf{0}$, the error must satisfy the linear system $H\mathbf{e}^\top = \mathbf{z}$.

Decoding is therefore posed as an optimization problem: find the vector $\hat{\mathbf{e}}$ of minimum Hamming weight such that $H\hat{\mathbf{e}}^\top = \mathbf{z}$. This vector, known as the *coset leader*, represents the most likely error. Subtracting it from the received vector,

$$\hat{\mathbf{c}} = \mathbf{r} \oplus \hat{\mathbf{e}},$$

projects the corrupted vector back into the code subspace, recovering the original representation when the error lies within the correction capability of the code.

4. Specific Subspace Configurations

Having established the general framework of subspace embeddings, specific variations of the code subspace $\mathcal{C}$ are now analyzed. The trade-off between the error-correction radius t and the expansion factor n/k is quantified below.

4.1. The Hamming$(7, 4)$ Subspace

The Hamming$(7, 4)$ code embeds 4-dimensional data vectors into a 7-dimensional ambient space using a systematic generator $G = [I_4 \mid P]$ (8).

The parity sub-matrix P is designed such that the rows of the parity-check matrix H are linearly independent. For the $(7, 4)$ configuration, the parity-check matrix $H \in \mathbb{F}_2^{3\times 7}$ is constructed by setting its columns to be the $2^3 - 1 = 7$ non-zero vectors of $\mathbb{F}_2^3$:

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

The corresponding generator matrix $G \in \mathbb{F}_2^{4\times 7}$ satisfying $GH^\top = \mathbf{0}$ is:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

4.1.1. Geometric Properties

The Hamming$(7, 4)$ subspace exhibits a distinct geometric property: the Hamming spheres of radius $t = 1$ centered at the valid codewords form a partition of the vector space $\mathbb{F}_2^7$.

- **Minimum Distance:** $d_{\min} = 3$. This allows for the unique correction of single-bit errors ($t = 1$).
- **Perfect Packing:** The number of vectors in a Hamming ball of radius 1 is $\sum_{i=0}^{1} \binom{7}{i} = 1 + 7 = 8$. Since there are $2^4 = 16$ codewords, the total coverage is $16 \times 8 = 128$, which exactly equals the cardinality of the vector space 2^7.

Algebraically, this implies that every vector $\mathbf{v} \in \mathbb{F}_2^7$ is either a codeword or at distance 1 from a unique codeword. There are no wasted vectors in the space that do not map to a syndrome.

4.1.2. Decoding as a Column Bijection

The error correction mechanism relies on the bijection between the standard basis vectors of the error space and the columns of H. Let $\mathbf{e}_j$ be the j-th standard basis vector (representing a bit-flip at index j). The syndrome operation is a linear map that extracts the j-th column of H:

$$\mathbf{z} = H\mathbf{e}_j^\top = \mathbf{h}_j.$$

Since the columns of H are distinct by construction, the map $\mathbf{e}_j \mapsto \mathbf{z}$ is invertible for all weight-1 errors.

For example, consider the encoding of the integer value 2 ($\mathbf{d} = [0, 1, 0, 0]$). The codeword is $\mathbf{c} = [0, 1, 0, 0, 1, 0, 1]$. If a perturbation occurs at the third bit ($e_3 = 1$), the retrieved vector is $\mathbf{r} = \mathbf{c} \oplus \mathbf{e}_3$. The syndrome computation yields:

$$\mathbf{z} = H(\mathbf{c} \oplus \mathbf{e}_3)^\top = H\mathbf{c}^\top \oplus H\mathbf{e}_3^\top = \mathbf{0} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}.$$

The resulting syndrome $[0, 1, 1]^\top$ is identical to the 3rd column of H (using 0-based indexing for the data bits corresponding to column 6 in the matrix above). This uniquely identifies $\mathbf{e}_3$ as the error vector, allowing for exact reconstruction via $\hat{\mathbf{c}} = \mathbf{r} \oplus \mathbf{e}_3$.

4.2. The Extended Hamming$(8, 4)$ Subspace

The standard Hamming$(7, 4)$ code has minimum Hamming distance $d_{\min} = 3$. As a result, error vectors of weight two ($\|\mathbf{e}\|_0 = 2$) are not correctable. More critically, because the columns of the parity-check matrix are closed under addition, the syndrome of a double-bit error can coincide with the syndrome of a single-bit error at a different location:

$$\mathbf{h}_i \oplus \mathbf{h}_j = \mathbf{h}_k.$$

This linear dependence makes certain two-bit errors indistinguishable from one-bit errors, causing the decoder to miscorrect and increase the total error.

To address this issue, the code is extended by adding a single global parity bit, enlarging the space from $\mathbb{F}_2^7$ to $\mathbb{F}_2^8$. This extension increases the minimum distance to $d_{\min} = 4$ and

enables *SECDED* (Single Error Correction, Double Error Detection). The extended parity-check matrix $\bar{H} \in \mathbb{F}_2^{4\times 8}$ is constructed by appending a row of ones and an additional parity column:

$$\bar{H} = \begin{bmatrix} H & \mathbf{0} \\ \mathbf{1}^\top & 1 \end{bmatrix}.$$

With this structure, decoding relies on two simultaneous checks applied to a retrieved vector $\mathbf{r}$:

1. **Syndrome Check:** $\mathbf{z} = H\mathbf{r}_{1:7}^\top$.
2. **Global Parity Check:** $p = \sum_{i=1}^{8} r_i \pmod 2$.

4.2.1. Erasure Detection Logic

The combination of these two checks distinguishes error types:

- **Single-bit error ($w = 1$):** The syndrome is nonzero and the parity check fails ($p = 1$). The error location is uniquely identified and corrected.
- **Double-bit error ($w = 2$):** The syndrome is nonzero but the parity check passes ($p = 0$).

In the second case, the decoder detects that the vector has left the code subspace but cannot uniquely identify the error pattern. Rather than attempting an incorrect correction, the decoder declares an **erasure** at time step t and treats the value as missing, denoted $v_t = \text{NaN}$.

4.3. Manifold-Aware Reconstruction

When an erasure is declared, the missing state vector is reconstructed using the geometric structure of the sequence. The sequence of vectors $\{\mathbf{v}_t\}_{t=0}^T$ is modeled as samples from a smooth curve $\gamma : [0, T] \to \mathbb{R}^d$.

4.3.1. Continuity Assumption

It is assumed that the generating process is continuous with constant K, so that

$$\|\mathbf{v}_t - \mathbf{v}_{t-1}\|_2 \le K.$$

This assumption implies that consecutive state vectors vary smoothly, and that large discontinuities are unlikely.

4.3.2. Recovery as a Boundary Value Problem

Reconstruction of the erased vector $\mathbf{v}_t$ is posed as a local optimization problem. The estimate $\hat{\mathbf{v}}_t$ is chosen to minimize the second-order difference of the trajectory:

$$\hat{\mathbf{v}}_t = \arg\min_{\mathbf{x}} \|(\mathbf{v}_{t+1} - \mathbf{x}) - (\mathbf{x} - \mathbf{v}_{t-1})\|_2^2.$$

This objective favors local linearity of the trajectory. Differentiating with respect to $\mathbf{x}$ and setting the result to zero yields

$$-2(\mathbf{v}_{t+1} - \mathbf{x}) - 2(\mathbf{x} - \mathbf{v}_{t-1}) = 0,$$

which gives the solution

$$\hat{\mathbf{v}}_t = \frac{1}{2}(\mathbf{v}_{t-1} + \mathbf{v}_{t+1}).$$

This linear interpolation replaces the corrupted discrete value with the geometric midpoint of its neighbors, preserving the stability of the inner product space $\mathbb{R}^d$ and preventing the error amplification described earlier.

4.4. The Extended Binary Golay$(24, 12)$ Subspace

To handle higher noise levels where the error weight $\|\mathbf{e}\|_0$ exceeds the correction capability of Hamming codes, the **Extended Binary Golay Code** ($\mathcal{G}_{24}$) is employed. This is a $(24, 12, 8)$ linear block code that embeds a 12-bit data vector into a 24-bit binary space, providing substantially stronger error protection (8).

In the quantized KV cache, the input space $\mathbb{F}_2^{12}$ is constructed as the direct sum of three 4-bit integer subspaces,

$$\mathbb{F}_2^{12} \cong \mathbb{F}_2^4 \oplus \mathbb{F}_2^4 \oplus \mathbb{F}_2^4.$$

This decomposition allows three quantized state coefficients to be jointly encoded into a single Golay codeword, enabling protection against errors that affect multiple coefficients simultaneously.

4.4.1. Algebraic Construction

The code is constructed using the *Paley construction* (8). Let $Q \subset \mathbb{Z}_{23}$ denote the set of quadratic residues modulo 23. Define an 11×11 binary matrix S by

$$s_{ij} = \begin{cases} 1, & \text{if } (j - i) \in Q, \\ 0, & \text{otherwise.} \end{cases}$$

The parity submatrix $B \in \mathbb{F}_2^{12\times 12}$ is obtained by bordering S with a row and column of ones and setting the diagonal entries to zero. This yields a symmetric matrix

$$B = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

Self-Orthogonality. A key property of this construction is that B is orthogonal over $\mathbb{F}_2$, satisfying $BB^\top = I_{12}$. Using B, the systematic generator and parity-check matrices are defined as

$$G_{24} = [I_{12} \mid B], \qquad H_{24} = [B^\top \mid I_{12}].$$

Because B is symmetric, these matrices are closely related. Code validity follows from

$$G_{24}H_{24}^\top = B \oplus B = \mathbf{0} \pmod 2,$$

ensuring that all encoded vectors lie in the kernel of H_{24}.

4.4.2. Combinatorial Properties

The Golay code has minimum Hamming distance $d_{\min} = 8$, which guarantees correction of all error vectors with weight $w(\mathbf{e}) \leq 3$. The code is closely related to the Steiner system $S(5, 8, 24)$, a combinatorial design in which every 5-element subset of coordinates is contained in exactly one weight-8 codeword. This structure implies that $\mathcal{G}_{24}$ achieves an optimal packing of Hamming spheres in 24 dimensions.

4.4.3. Coset Decoding of Higher-Weight Errors

In contrast to Hamming codes, where syndromes identify individual bit positions, the Golay syndrome $\mathbf{z} \in \mathbb{F}_2^{12}$ corresponds to error patterns distributed across the entire 24-bit block.

Consider an error vector $\mathbf{e}$ that introduces one bit error in each of the three 4-bit partitions,

$$\mathbf{e} = \mathbf{e}_1 \oplus \mathbf{e}_2 \oplus \mathbf{e}_3,$$

so that $w(\mathbf{e}) = 3$. Although such an event would exceed the correction capability of Hamming codes, it lies within the correction radius of the Golay subspace.

Decoding proceeds by computing the syndrome $\mathbf{z} = H_{24}\mathbf{r}^\top$. Because the coset leaders of all weight-3 error patterns are unique, there exists a deterministic inverse map $\psi : \mathbb{F}_2^{12} \to \mathbb{F}_2^{24}$ that recovers the error vector exactly. The corrected data is then obtained as

$$\hat{\mathbf{d}} = (\mathbf{r} \oplus \psi(H_{24}\mathbf{r}^\top))_{1:12}.$$

This joint encoding ensures that even when each constituent 4-bit value is corrupted by a single bit error, the overall state vector can be fully restored using the geometric structure of the Golay subspace.

5. Empirical Evaluation

5.1. Evaluation Metric

To assess the impact of storage corruption on model performance, the standard metric of Perplexity is employed.

Definition 5.1 (Perplexity)**.** Given a token sequence $X = (x_1, \ldots, x_T)$, the **Perplexity** (PPL) is defined as the exponential of the negative log-likelihood averaged over the sequence length:

$$\mathrm{PPL}(X) = \exp\left(-\frac{1}{T}\sum_{t=1}^{T} \ln P_\theta(x_t \mid x_{<t})\right).$$

This metric quantifies the uncertainty of the model; lower values indicate better predictive performance, while large divergences signal catastrophic algebraic instability.

5.2. Numerical Stability and Divergence

Table 3 reports PPL under increasing bit-flip probability p. Two baselines are important for interpreting these results. First, **FP16 (Oracle)** represents the idealized computation in continuous space $\mathbb{R}^d$, where cache values are stored without quantization or bit-level corruption. Second, **Int4 (Unprotected)** represents the quantized cache without algebraic protection. The baseline gap at $p = 0$ (13.92 vs. 12.49) isolates the *deterministic quantization penalty* introduced by the discretization map Π even in the absence of corruption (7).

As p increases, any additional growth in PPL reflects *stochastic instability* caused by bit flips in the stored matrices K and V. In particular, a sharp rise in PPL indicates that corrupted entries are producing large, spurious inner products that dominate the attention scores, destabilizing the softmax projection and shifting probability mass toward incorrect indices.

Table 3. **Perplexity (PPL) Divergence.** Comparison of subspace projections under increasing perturbation densities. The unprotected metric diverges catastrophically at $p = 10^{-2}$.

Protection Mode	**Baseline**	$p = 10^{-4}$	$p = 10^{-3}$	$p = 10^{-2}$
FP16 (Oracle)	12.49	12.49	12.49	12.49
Int4 (Unprotected)	13.92	14.59	23.44	2667.64
Int4-Hamming(7,4)	13.92	13.92	13.90	20.77
Int4-Hamming(8,4)	13.92	13.88	14.04	93.18
H(8,4) + Interp	**13.92**	**13.91**	**13.96**	**14.79**
Int12-Golay	**13.92**	**13.92**	**13.92**	**14.60**

Interpretation of the High-Noise Regime ($p = 10^{-2}$). At $p = 10^{-2}$, the unprotected quantized cache collapses (PPL $= 2667.64$). This is consistent with the metric-mismatch mechanism: a small Hamming perturbation in storage can induce a large Euclidean perturbation after dequantization, producing outlier attention scores. The softmax then concentrates probability mass on the corrupted entry, effectively selecting an incorrect column of V repeatedly (11). In contrast, both Golay and H(8,4)+Interpolation

remain near the quantized baseline (≈ 13.92), indicating that the protected cache preserves stable inner products despite corruption.

5.3. Analysis of Algebraic Failures

A notable anomaly in Table 3 is that raw Hamming$(8, 4)$ performs *worse* (PPL 93.18) than Hamming$(7, 4)$ (PPL 20.77) at $p = 10^{-2}$, despite having strictly stronger detection guarantees. This effect is explained by how each scheme treats multi-bit events, as shown by the correction and detection statistics in Tables 4 and 5. Table 4 counts vectors corrected as single-bit errors, while Table 5 counts vectors flagged as multi-bit errors (erasures).

Table 4. **Correction Capacity (Single-bit Errors).** Number of vectors projected back to the code subspace (raw counts).

Protection Mode	$p = 0$	$p = 10^{-4}$	$p = 10^{-3}$	$p = 10^{-2}$
FP16	0	0	0	0
Int4-Hamming(7,4)	0	130,698	1,298,906	12,633,477
Int4-Hamming(8,4)	0	130,865	1,296,255	12,154,744
Int12-Golay	0	149,934	1,498,804	14,973,685

Table 5. **Detection Capacity (Multi-bit Errors).** Number of vectors detected as corrupted but uncorrectable (erasures).

Protection Mode	$p = 0$	$p = 10^{-4}$	$p = 10^{-3}$	$p = 10^{-2}$
Int4-Hamming(7,4)	0	0	0	0
Int4-Hamming(8,4)	0	60	5,218	489,986
Int12-Golay	0	0	0	5,578

5.3.1. The Miscorrection vs. Erasure Trade-off

At $p = 10^{-2}$, Hamming$(8, 4)$ flags approximately 4.9×10^5 retrieved vectors as multi-bit errors (Table 5). This is the intended SECDED behavior: ambiguous patterns are detected rather than miscorrected. However, the *action* taken after detection matters.

1. **Hamming$(7, 4)$ (miscorrection):** The Hamming$(7, 4)$ decoder has no mechanism to distinguish a two-bit syndrome collision from a true one-bit error. It therefore applies a single-bit correction even when the true error weight is two. While this is algebraically incorrect, it often returns a nonzero value, meaning the downstream dot products still receive a usable (though biased) signal.

2. **Hamming$(8, 4)$ (erasure without reconstruction):** The extended code correctly detects these events as ambiguous. In the raw variant, the corresponding cached vector is effectively removed (e.g., zeroed). In attention, this removes a term from the weighted sum, reducing the available signal in V and degrading the context vector $\mathbf{z}$. When this occurs hundreds of thousands of times, the accumulated signal loss produces the large PPL increase (93.18).

This explains why a "stronger" detector can yield worse performance if detected errors are handled by hard erasure without recovery.

5.4. Validation of the Smoothness Hypothesis

The interpolation strategy directly tests the smoothness assumption from Section 5.5: if the cached state vectors vary smoothly across time, then a detected erasure can be approximated well using neighboring vectors. Table 6 reports the increase in perplexity relative to the $p = 0$ quantized baseline.

For Hamming$(8, 4)$ without interpolation, the degradation at $p = 10^{-2}$ is $+80.69$, reflecting the cumulative cost of hard erasures. When interpolation is enabled, the degradation drops to $+2.30$, indicating that the erased vectors can be reconstructed accurately enough to preserve stable inner products. In other words, the sequence has low local curvature for a large fraction of timesteps, making linear interpolation an effective geometric regularizer.

Table 6. **Relative Degradation (Δ PPL).** Increase in perplexity relative to the noise-free baseline.

Protection Mode	$p = 0$	$p = 10^{-4}$	$p = 10^{-3}$	$p = 10^{-2}$
FP16	0.00	0.00	0.00	0.00
Int4	+1.43	+2.10	+10.95	+2655.15
Int4-Hamming(7,4)	+1.43	+1.43	+1.42	+8.28
Int4-Hamming(8,4)	+1.43	+1.39	+1.56	+80.69
H(8,4) + Interp	**+1.43**	**+1.42**	**+1.47**	**+2.30**
Int12-Golay	+1.43	+1.43	+1.43	+2.12

5.5. Summary of Efficiency

Table 7 summarizes performance at the critical regime $p = 10^{-2}$, where unprotected quantization fails catastrophically. Golay provides the strongest algebraic protection and corrects substantially more errors than Hamming-based schemes. However, Hamming$(8, 4)$ with interpolation achieves comparable stability (PPL 14.79 vs. 14.60) while preserving standard 8-bit alignment. This supports the central conclusion that combining algebraic detection with geometric reconstruction can match the robustness of higher-dimensional embeddings at lower practical overhead.

References

[1] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332, 2022.

Table 7. **System Performance Summary at $p = 10^{-2}$.** Algebraic overhead (bits/value) vs. stability and correction capacity.

Metric	Int4	Ham(7,4)	H(8,4)	H(8,4)+Interp	Golay
PPL	2667.64	20.77	93.18	**14.79**	**14.60**
PPL Increase	213×	1.66×	7.46×	1.18×	1.17×
Errors Corrected	0	12.6M	12.2M	12.2M	15.0M
Errors Detected	0	0	490K	490K	5.6K
Bits/Value	4	7	8	8	8

Foundational work on outlier emergence in quantized LLMs.

[2] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Albert Alet, Amjad Raman, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. Technical Report covering Llama 3 and 3.1.

[3] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

[4] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*, 2024.

[5] Kuang-Hua Huang and Jacob A Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE Transactions on Computers*, 100(6):518–528, 1984. Foundational work establishing the linear algebraic model for protecting matrix-vector products.

[6] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E Gonzalez, Ion Stoica, and Hao Zhang. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023.

[7] Zirui Liu, Jiayi Yuan, Shaochen Jin, Hongye andail, et al. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

[8] Florence J MacWilliams and Neil JA Sloane. *The Theory of Error-Correcting Codes*. Elsevier, 1977. The definitive reference for the algebraic construction of Golay codes and the geometry of Hamming spaces.

[9] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[11] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.