

GeoMotionGPT: Geometry-Aligned Motion Understanding with Large Language Models

Zhankai Ye^{1,*} Bofan Li^{1,*} Yukai Jin¹ Shuoqiu Li¹
Wei Wang² Yanfu Zhang³ Shangqian Gao^{1,†} Xin Liu^{1,†}

¹Florida State University ²Texas Tech University ³College of William & Mary
{zy22b, bl21n, yj25b, sl24cs, sg24bi, xliu15}@fsu.edu wei.wang@ttu.edu yzhang105@wm.edu

Abstract

Discrete motion tokenization allows large language models (LLMs) to process motion as sequences of vocabulary items, but standard pipelines connect a motion codebook to the LLM only through token IDs. We introduce GeoMotionGPT, which instead gives the codebook and its corresponding motion-token embeddings a shared orthogonality prior. A decoder-only quantizer uses Gumbel-Softmax assignments and utilization regularization; a fixed sparse isometry transfers the learned codebook Gram matrix to the LLM embedding space at initialization; and soft orthogonality penalties are applied during tokenizer training and LLM fine-tuning. Under the GPT-2 setting, GeoMotionGPT improves a descriptive aggregate of retrieval and captioning metrics by **22.4%** over the strongest reported baseline on HumanML3D and by **14.4%** on KIT-ML. Controlled ablations on both datasets attribute these gains to the tokenizer, sparse initialization, and regularization. The results support orthogonality as a useful structural prior while also showing that the model does not dominate every individual metric.

1 Introduction

Human motion understanding is important for embodied agents that must interpret and act in the physical world (Zhao et al., 2023; Driess et al., 2023). Recent systems represent motion as discrete tokens and use LLMs for motion understanding, reasoning, and generation (Wu et al., 2024a; Zhou et al., 2024; Zhang et al., 2023; Guo et al., 2022b). This interface is attractive because it turns a continuous signal into a sequence that an LLM can consume with its existing language vocabulary.

Most motion-language pipelines train the tokenizer and the language model in separate stages (Zhang et al., 2023; Guo et al., 2022b; Jiang

et al., 2023; Wang et al., 2024; Lai et al., 2025). A vector-quantized model first compresses motion into codebook indices, after which new LLM vocabulary entries are learned for those indices. Token identity links the two stages, but the pairwise relations among the learned code vectors are not transferred explicitly to the initial LLM embeddings. The language model must therefore learn those relations from downstream supervision.

We study whether a simple structural prior can make this handoff more controlled. GeoMotionGPT¹ encourages the motion code vectors to be approximately orthogonal and uses a fixed sparse isometry to initialize the corresponding LLM motion-token embeddings. The projection preserves the codebook Gram matrix exactly at initialization. During LLM fine-tuning, a soft orthogonality penalty discourages the motion-token embeddings from collapsing into redundant directions while still allowing task-driven adaptation.

Orthogonality is a deliberately limited prior. It does not encode the semantic topology of human motion or align motion vectors directly with text embeddings. Instead, it provides a well-conditioned starting structure for newly introduced tokens without requiring paired contrastive targets. Similar actions can still acquire related contextual representations through the LLM objective.

The framework has three components. First, a decoder-only quantizer uses Gumbel-Softmax assignments (Jang et al., 2017), a straight-through estimator, and utilization regularization so that the codebook can be optimized jointly with reconstruction and geometry objectives. Second, a parameter-free sparse projection places each code dimension in a distinct LLM embedding dimension, preserving all pairwise inner products at initialization. Third, soft orthogonality penalties are applied in both training stages: to the codebook during tok-

*Equal contribution. †Corresponding authors.

¹Project resources: [GitHub](#) and [Hugging Face](#).

tokenizer training and to the motion-token embedding block during LLM fine-tuning.

On HumanML3D and KIT-ML, GeoMotionGPT improves a descriptive aggregate of retrieval and captioning metrics by **22.4%** and **14.4%**, respectively, over the strongest reported baselines under the GPT-2 setting. The aggregate is not a standard benchmark metric, so we report every component separately and avoid treating it as evidence of uniform dominance. Controlled ablations on both datasets show that removing the tokenizer design, sparse initialization, or orthogonality penalty reduces performance.

Our contributions can be summarized as follows:

- We formulate the tokenizer-to-LLM handoff as transfer of a structural prior and use approximate orthogonality to keep newly introduced motion-token directions well conditioned.
- We combine a decoder-only Gumbel-Softmax quantizer with a sparse isometric initialization whose preservation of pairwise inner products follows directly from its construction.
- Experiments and controlled ablations on HumanML3D and KIT-ML characterize the benefits and trade-offs of the tokenizer, projection, and regularization choices.

2 Related Work

Motion Understanding Using LLMs. Motion-language methods differ mainly in how they connect motion and text. Some learn a shared space through contrastive or retrieval objectives (Petrovich et al., 2023; Tevet et al., 2022). Others convert motion into discrete or token-like sequences and apply language-modeling objectives (Guo et al., 2022b; Jiang et al., 2023; Wang et al., 2024; Zhu et al., 2025). Self-supervised motion objectives, including masked modeling, can strengthen these representations (Guo et al., 2024), while unified formulations cover multiple granularities and interaction settings (Park et al., 2025; Wu et al., 2025). More broadly, multimodal language models show that a new modality can be introduced through adapters or projections into an LLM embedding space (Alayrac et al., 2022; Li et al., 2023).

VQ-VAE and Its Variants. Variational autoencoders optimize an evidence lower bound that combines reconstruction with latent regularization (Kingma and Welling, 2014). Increasing the

weight of the KL term can encourage more factorized latent representations, although it may reduce reconstruction fidelity (Higgins et al., 2017). VQ-VAE instead uses a learned discrete codebook at the bottleneck, with auxiliary losses for codebook learning and encoder commitment (van den Oord et al., 2017); hierarchical discrete latents extend this design across multiple representation scales (Razavi et al., 2019). Later quantizers simplify the discretization rule, encode residuals with multiple codebooks, improve large-vocabulary utilization, or structure codebook reuse (Mentzer et al., 2024; Lee et al., 2022; Zhu et al., 2024; Zhang et al., 2024). SoftVQ-VAE explores a differentiable continuous tokenizer for image generation (Chen et al., 2025). Our focus is different: we use differentiable assignments to optimize the geometry and utilization of a motion codebook that will initialize LLM motion-token embeddings.

Orthogonality in Representation Learning. Orthogonality is a common geometric bias for learning stable and well-conditioned representations. Spectral control can improve optimization by limiting amplification or suppression of signals and gradients (Jia et al., 2017). Orthogonal transformations also preserve inner products and norms, a property used in work on compatible and approximately isometric representations (Ricci et al., 2025; Qi et al., 2020; Lezama et al., 2018). Practical approaches include Stiefel-manifold optimization, normalization methods, and partial orthogonalization (Huang et al., 2018, 2020). Related constraints have been studied in convolutional networks, initialization methods, and attention models (Bansal et al., 2018; Xie et al., 2017; Zhang et al., 2021; Fei et al., 2022).

Positioning of GeoMotionGPT. Prior motion tokenizers primarily optimize reconstruction and code usage (Zhang et al., 2023; Guo et al., 2022b, 2024; Hong et al., 2025). DVQ adds differentiable assignments so that an explicit codebook geometry penalty can be optimized together with those objectives. Prior orthogonality methods usually regularize a single representation space (Jia et al., 2017; Huang et al., 2018, 2020; Bansal et al., 2018; Zhang et al., 2021; Fei et al., 2022). We apply the same form of regularizer to the Stage-1 codebook and the Stage-2 motion-token embeddings, with a fixed isometry transferring the codebook Gram matrix at initialization. This is a structural bridge between two training stages, not a direct semantic alignment between motion and text.

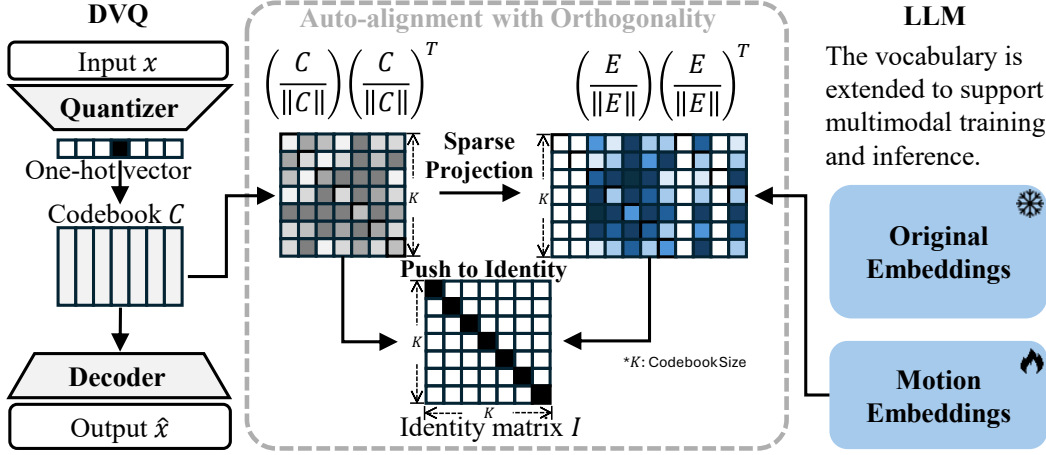


Figure 1: Overview of GeoMotionGPT. Left: a DVQ motion tokenizer maps an input motion x to discrete codebook indices and reconstructs $\hat{x}$. Middle: orthogonality regularization encourages the normalized codebook Gram matrix to approach the identity, and a fixed sparse isometry preserves this matrix when initializing the motion-token embeddings. Right: the LLM vocabulary is extended with trainable motion-token embeddings while the original text embeddings remain frozen.

3 Our Approach

3.1 Orthogonality as a Shared Structural Prior

Let $\mathcal{M}$ denote the continuous motion space, $\mathcal{Z} = \{1, \dots, K\}$ the discrete motion-token index set, and $\mathcal{S}$ the LLM embedding space.

Existing approaches typically learn a quantization map $q : \mathcal{M} \rightarrow \mathcal{Z}$ and a separate embedding map $\phi : \mathcal{Z} \rightarrow \mathcal{S}$. The index $k \in \mathcal{Z}$ links the two stages, but the pairwise inner products among code vectors are not normally used to initialize the corresponding LLM embeddings.

We use orthogonality as a shared structural prior for this handoff. As illustrated in Figure 1, the motion codebook $\mathbf{C} = \{\mathbf{c}_1, \dots, \mathbf{c}_K\} \subset \mathbb{R}^D$ is encouraged to approximate an orthonormal set:

$$\langle \mathbf{c}_i, \mathbf{c}_j \rangle \approx \delta_{ij}, \quad (1)$$

where δ_{ij} is the Kronecker delta. Exact row orthogonality is possible only when $K \leq D$; our experiments use $K = D = 512$, and training imposes a soft penalty rather than a hard constraint.

To enforce this condition during training, we apply orthogonal regularization to the codebook. Let $\hat{\mathbf{C}}$ be the row-normalized codebook where $\hat{\mathbf{c}}_k = \mathbf{c}_k / \|\mathbf{c}_k\|_2$. We compute the Gram matrix $\mathbf{G} = \hat{\mathbf{C}}\hat{\mathbf{C}}^\top$ and define the orthogonal loss as:

$$\mathcal{L}_{\text{ortho}} = \|\mathbf{G} - \mathbf{I}_K\|_F^2. \quad (2)$$

This loss discourages redundant code directions.

Two properties of this objective matter. First, it is supervision-free and closed form: unlike contrastive or metric-based alignment, it needs no positive and negative pairs, similarity targets, or text encoder because the target is the identity matrix. Second, it is a *soft* penalty rather than a hard constraint, and not an assertion that similar actions must be mutually unrelated. Eq. 2 is one term in a composite objective dominated by reconstruction during tokenizer training and by the task loss during fine-tuning (Section 3.4). Its role is to reduce directional redundancy. The objective does not preserve the neighborhood topology of the original motion space, and semantic relations among actions must still be learned from the task objective.

3.2 Structure-Preserving Sparse Projection

We next transfer the learned codebook Gram matrix to the LLM motion-token embeddings at initialization. A learned dense map has no built-in inner-product preservation guarantee, so we instead use a fixed sparse projection.

We map the D -dimensional motion codes into the higher-dimensional LLM space $\mathbb{R}^{D'}$ ($D' \gg D$) by distributing them across randomly selected active dimensions. Specifically, we define a fixed projection matrix $\mathbf{P} \in \{0, 1\}^{D' \times D}$ initialized by randomly selecting D unique row indices $\mathcal{I} \subset \{1, \dots, D'\}$ to act as identity mappings, while setting all other entries to zero. As illustrated in Figure 1, the embedding is computed as:

$$\mathbf{e}_k = \mathbf{P}\mathbf{c}_k. \quad (3)$$

Intuitively, this operation scatters the motion code values into the high-dimensional vector $\mathbf{e}_k$ at random positions, filling the remaining $D' - D$ dimensions with zeros.

This projection is an isometric embedding and preserves pairwise inner products regardless of which unique row indices are selected.

Lemma. *Let $\mathbf{P}$ be a sparse projection matrix where each column contains exactly one '1' at a unique row index and '0' elsewhere. If the original codebook vectors $\{\mathbf{c}_k\}$ are pairwise orthogonal, then the projected embeddings $\{\mathbf{e}_k\}$ are also pairwise orthogonal.*

Proof. The inner product in the LLM space is:

$$\mathbf{e}_i^\top \mathbf{e}_j = (\mathbf{P}\mathbf{c}_i)^\top (\mathbf{P}\mathbf{c}_j) = \mathbf{c}_i^\top (\mathbf{P}^\top \mathbf{P}) \mathbf{c}_j.$$

Since $\mathbf{P}$ maps source dimension to a unique target dimension without overlap, the columns of $\mathbf{P}$ are orthonormal. Thus, $\mathbf{P}^\top \mathbf{P} = \mathbf{I}_D$. It follows that:

$$\mathbf{e}_i^\top \mathbf{e}_j = \mathbf{c}_i^\top \mathbf{I}_D \mathbf{c}_j = \mathbf{c}_i^\top \mathbf{c}_j.$$

If $\mathbf{c}_i \perp \mathbf{c}_j$, then $\mathbf{e}_i \perp \mathbf{e}_j$.

Two clarifications define the scope of this result. (i) *The guarantee also covers approximate orthogonality.* Since $\mathbf{P}^\top \mathbf{P} = \mathbf{I}_D$ by construction, the argument applies to *every* pair of codes. Writing $\hat{\mathbf{E}}$ for the row-normalized projected embeddings, $\hat{\mathbf{E}}\hat{\mathbf{E}}^\top = \hat{\mathbf{C}}\hat{\mathbf{C}}^\top$, hence $\|\hat{\mathbf{E}}\hat{\mathbf{E}}^\top - \mathbf{I}_K\|_F = \|\hat{\mathbf{C}}\hat{\mathbf{C}}^\top - \mathbf{I}_K\|_F$: the projection transfers the *exact degree* of approximate orthogonality attained by the soft penalty, which is the regime that actually arises in training. (ii) *It is an initialization mechanism, not a permanent constraint.* $\mathbf{P}$ is fixed, but the motion-token embeddings are optimized in Stage 2. The equality of the two Gram matrices therefore holds at initialization only. Stage-2 regularization encourages approximate orthogonality after fine-tuning, but it does not preserve exact codebook-to-embedding correspondence.

3.3 Decoder-Only Vector Quantization (DVQ)

Because the sparse projection preserves the codebook Gram matrix at initialization, the first stage must learn a useful codebook structure. Standard VQ-VAE uses a non-differentiable nearest-neighbor assignment and relies on surrogate updates for the encoder and codebook. We instead use a Gumbel-Softmax relaxation so that reconstruction, utilization, and geometry terms can be optimized in one computation graph.

Given motion input x , the quantizer produces logits $\mathbf{z} = Q(x) \in \mathbb{R}^K$. We sample a relaxed as-

signment with Gumbel-Softmax (Jang et al., 2017) and discretize it for the forward pass:

$$\begin{aligned} \mathbf{y}_{\text{soft}} &= \text{GumbelSoftmax}(\mathbf{z}; \tau), \\ \mathbf{y}_{\text{hard}} &= \text{one-hot}\left(\arg \max_k y_{\text{soft},k}\right), \end{aligned} \quad (4)$$

where τ is the temperature. The selected motion embedding is

$$\mathbf{h} = \mathbf{y}_{\text{hard}}^\top \mathbf{C}. \quad (5)$$

where $y_{\text{soft},k}$ is the k -th entry of $\mathbf{y}_{\text{soft}}$. We use the straight-through estimator (STE) (Bengio et al., 2013): the forward pass uses $\mathbf{y}_{\text{hard}}$, while the backward pass follows the relaxed assignment. The decoder output is $\hat{x} = D(\mathbf{h})$.

The STE gradient is biased, but its forward path matches the discrete representation used at inference. Without the STE, training optimizes only the continuous relaxation; in our ablation this mismatch destabilizes late training and leaves 323 of 512 codes unused (Table 8). Gumbel noise is used only during training. At inference, tokenization is a deterministic $\arg \max$ over the logits.

We also regularize codebook utilization to prevent token collapse. Let q_k be the normalized mini-batch assignment frequency of code k , computed from the scheduled soft-to-hard utilization statistics described in Appendix A.1. To enforce a balanced distribution, we maximize its self-entropy:

$$\mathcal{L}_{\text{util}} = -H(\mathbf{q}) = -\sum_{k=1}^K q_k \log q_k, \quad (6)$$

with the convention $0 \log 0 = 0$. Minimizing this term favors higher-entropy assignment frequencies. It encourages balanced use but does not guarantee a uniform distribution. The two regularizers are complementary: $\mathcal{L}_{\text{util}}$ shapes the *marginal* distribution over codes (how often each is selected), whereas $\mathcal{L}_{\text{ortho}}$ shapes their *geometry* (how they are arranged). Balanced usage alone does not stop codes from becoming near-duplicates, and orthogonality alone does not stop a few codes from dominating assignments.

3.4 Two-Stage Orthogonality Regularization

We apply the same orthogonality prior in two stages, as illustrated in Figure 1.

In Stage 1, we train DVQ with a composite objective that adds geometry and utilization terms to the reconstruction loss $\mathcal{L}_{\text{rec}} = \|x - \hat{x}\|_2^2$:

$$\mathcal{L}_{\text{DVQ}} = \mathcal{L}_{\text{rec}} + \lambda_{\text{ortho}} \mathcal{L}_{\text{ortho}} + \lambda_{\text{util}} \mathcal{L}_{\text{util}}. \quad (7)$$

$\mathcal{L}_{\text{ortho}}$ and $\mathcal{L}_{\text{util}}$ are defined in Eq. 2 and Eq. 6.

In Stage 2, we initialize the LLM motion-token block $\mathbf{E} \in \mathbb{R}^{K \times D'}$ by sparse projection and extend the original token embedding matrix $\mathbf{E}_{\text{org}} \in \mathbb{R}^{N \times D'}$ as $\mathbf{E}_{\text{new}} = [\mathbf{E}_{\text{org}}; \mathbf{E}] \in \mathbb{R}^{(N+K) \times D'}$. We freeze the original text embeddings and optimize the motion-token embeddings together with the trainable LLM parameters. A soft orthogonality term continues to discourage redundancy within the motion-token block:

$$\mathcal{L}_{\text{tuning}} = \underbrace{-\mathbb{E}_{(X,Y)} \sum_{t=1}^{|Y|} \log p_{\theta}(y_t | y_{<t}, X)}_{\text{task loss}} + \underbrace{\lambda'_{\text{orth}} \left\| \hat{\mathbf{E}} \hat{\mathbf{E}}^{\top} - \mathbf{I}_K \right\|_F^2}_{\text{orthogonality regularization}} \quad (8)$$

where $\hat{\mathbf{E}} \in \mathbb{R}^{K \times D'}$ contains row-normalized motion embeddings, $\hat{\mathbf{e}}_k = \mathbf{e}_k / \|\mathbf{e}_k\|_2$. The input $X = [X_{\text{prompt}}, X_{\text{motion}}]$ contains prompt and DVQ motion tokens, Y is the target response, and p_{θ} is the LLM.

Why two stages, and what bridges them. Two-stage training is common in motion-language systems: a tokenizer is learned first, then an LLM is adapted. Our bridge carries more than token identity. Stage 1 shapes the codebook Gram matrix, and the sparse projection copies that matrix to the initial motion-token embeddings. The Stage-2 penalty then encourages low pairwise cosine similarity as the embeddings adapt. It does not constrain each tuned embedding to remain close to its projected Stage-1 code. Contrastive learning could provide semantic supervision earlier, but it would require paired text representations during tokenizer training; our tokenizer remains motion-only.

4 Experiment

We describe the experimental setup, compare with prior methods, and report controlled ablations and diagnostics.

4.1 Experimental Settings

We evaluate three language backbones: GPT-2 (Radford et al., 2019), Qwen3 0.6B (Yang et al., 2025), and Llama 3.2 1B (Dubey et al., 2024). GPT-2 is fully fine-tuned, while Qwen3 0.6B and Llama 3.2 1B are adapted with LoRA (Hu et al., 2022).

All models are trained and evaluated on two standard benchmarks: HumanML3D (Guo et al.,

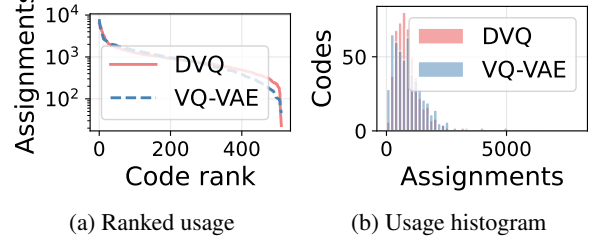


Figure 2: Codebook usage on the evaluation set. Both tokenizers use all 512 codes, but DVQ has a less skewed count distribution than the VQ-VAE baseline.

2022a) and KIT-ML (Plappert et al., 2016). HumanML3D is a benchmark for text-conditioned human motion understanding and generation. We follow its preprocessing pipeline and represent each frame as a 263-dimensional feature vector; each motion has three text captions. KIT-ML is a widely used benchmark with more diverse action descriptions, which provides a complementary testbed for cross-dataset evaluation.

We follow the motion-to-text evaluation protocol used by MotionGPT3 (Zhu et al., 2025).

Training protocol. DVQ uses a 512-code, 512-dimensional codebook and is trained for 500 epochs with batch size 512. We use AdamW with a base learning rate of 2×10^{-4} , 3% linear warmup, and cosine decay; quantizer parameters use a learning rate of 10^{-4} . The Gumbel-Softmax temperature stays at 0.4 for 300 epochs, decays exponentially to 0.01 over the next 100, and then remains fixed. The utilization statistic uses soft assignments for 150 epochs; its hardness coefficient then increases linearly from 0 to 1 over 50 epochs. Training uses stochastic Gumbel-Softmax assignments, whereas inference applies deterministic $\arg \max$ tokenization. For Stage-2 motion understanding, we train for 100 epochs with batch size 320, learning rate 10^{-4} , weight decay 10^{-2} , and cosine scheduling. Unless otherwise specified, each experiment runs on one NVIDIA B200 GPU. Appendix A.1 gives the remaining optimizer and adaptation details.

4.2 Evaluation Metrics

To assess codebook quality, we compare the per-code usage distributions of GeoMotionGPT and a VQ-VAE baseline over the evaluation set, reporting codebook utilization (the percentage of codes with non-zero usage) and the standard deviation of usage counts, which quantifies how concentrated the assignments are. For a compact descriptive summary, we report an aggregate that combines retrieval and

Table 1: Motion-to-text results on HumanML3D under the GPT-2 setting. GeoMotionGPT has the highest descriptive aggregate, 22.4% above MotionGPT3, but does not lead on every retrieval metric.

Approach	R@1	R@2	R@3	MMDist↓	BLEU-1↑	BLEU-4↑	R-L↑	CIDEr↑	BERTScore↑	Avg.↑
Real	0.523	0.725	0.828	2.901	-	-	-	-	-	-
TM2T (Guo et al., 2022b)	0.516	-	0.823	2.935	48.90	7.00	38.10	16.80	32.20	34.99
MotionGPT (Jiang et al., 2023)	0.543	-	0.827	2.821	48.20	12.50	37.40	29.20	32.40	38.03
LaMPM2T (Li et al., 2024)	0.547	-	0.831	2.808	47.80	13.04	37.10	28.90	32.70	38.07
MoTe (Wu et al., 2024b)	0.577	-	0.871	2.649	46.70	11.15	37.40	31.50	30.30	38.24
MotionGPT3 (Zhu et al., 2025)	0.573	0.773	0.864	2.430	59.08	19.41	46.18	28.72	35.23	43.71
GeoMotionGPT (Ours)	0.533	0.729	0.817	2.680	65.65	25.88	51.32	59.71	49.03	53.48

Table 2: Motion-to-text results on KIT-ML (Plappert et al., 2016) under the GPT-2 setting. GeoMotionGPT has the highest descriptive aggregate, 14.4% above MotionGPT.

Approach	R@1	R@2	R@3	MMDist↓	BLEU-1↑	BLEU-4↑	R-L↑	CIDEr↑	BERTScore↑	Avg.↑
TM2T (Guo et al., 2022b)	0.328	0.504	0.617	5.129	36.38	10.11	49.11	38.82	24.75	34.58
MotionGPT (Jiang et al., 2023)	0.148	0.270	0.348	7.107	44.04	19.45	44.76	47.46	37.56	36.46
MotionGPT3 (Zhu et al., 2025)	0.201	0.333	0.458	5.706	44.55	17.48	45.21	16.73	26.72	30.63
GeoMotionGPT (Ours)	0.328	0.574	0.676	3.565	44.11	15.69	42.48	58.81	36.57	41.71

Table 3: Numerical codebook statistics under an identical codebook size ($K=512$). Lower Max Count, Std., CV, and Gini all indicate a less skewed usage distribution; Rec. is the final validation reconstruction loss.

Tokenizer	Used	Dead	Max Cnt.↓	Std.↓	CV↓	Gini↓	Rec.↓
VQ-VAE	512	0	7,971	840.36	0.865	0.405	0.2421
DVQ (Ours)	512	0	6,300	791.20	0.814	0.358	0.3597

captioning metrics. It is not a standard benchmark measure and is not used to replace the individual metrics. For method m , let $\mathcal{K}_m \subseteq \{1, 2, 3\}$ contain the reported R-Precision cutoffs and define $\bar{R}_m = |\mathcal{K}_m|^{-1} \sum_{k \in \mathcal{K}_m} R_k$. The aggregate is

$$\text{Avg} = \frac{100 \cdot \bar{R}_m + B_1 + B_4 + \text{R-L} + C + S}{6},$$

where B_1 and B_4 denote BLEU-1 and BLEU-4 (Papineni et al., 2002), R-L denotes ROUGE-L (Lin, 2004), C is CIDEr (Vedantam et al., 2015), and S is BERTScore (Zhang et al., 2019). R-Precision (Aslam and Yilmaz, 2005) is scaled by 100. Some published baselines omit R@2, so their $\bar{R}_m$ averages R@1 and R@3; rows with all three recalls average all three. This difference in metric availability is another reason to treat the aggregate as descriptive only.

4.3 Codebook Distribution Analysis

We first examine how often each learned code is selected on the evaluation set. As shown in Fig. 2a, both methods exhibit a heavy-tailed usage pattern where a small subset of codes is used frequently, while many codes are used less often. Compared with the VQ-VAE baseline, DVQ is less skewed:

the most frequent codes are less dominant, and more codes occupy moderate usage ranges. The histogram in Fig. 2b shows the same pattern. Because both methods use every code at least once, these plots measure balance rather than the number of dead codes.

Numerical codebook statistics. Table 3 quantifies this trend. Both tokenizers keep all 512 codes alive, so utilization alone does not separate them; the dispersion statistics do. DVQ lowers the most frequent code count from 7,971 to 6,300 and reduces every measure of skew: usage standard deviation, coefficient of variation, and Gini coefficient. For transparency we also report reconstruction quality, where VQ-VAE is in fact stronger (final validation reconstruction loss 0.2421 vs. 0.3597). DVQ’s role in our framework is therefore not better raw reconstruction. Its differentiable assignment allows the codebook geometry penalty to be optimized together with reconstruction, which is evaluated in Section 4.6.

4.4 Performance on Motion Understanding

Table 1 compares GeoMotionGPT with published HumanML3D results under the GPT-2 setting. Relative to MotionGPT3, the descriptive Average increases by **22.4%**. The largest differences are in caption metrics: CIDEr increases by **107.9%**, BLEU-4 by **33.3%**, and BERTScore by **39.2%** (Zhu et al., 2025). Retrieval recall is slightly below the best method, and MMDist is slightly higher.

Table 4: Ablation on HumanML3D under GPT-2. *Baseline* removes geometry-preserving initialization and orthogonality regularization. We vary the tokenizer, projection, regularizer, and regularization weight. The corresponding KIT-ML ablation appears in Table 7.

Study	Setting	R@1	R@2	R@3	MMDist↓	BLEU-1↑	BLEU-4↑	R-L↑	CIDEr↑	BERTScore↑	Avg.↑
Baseline	Init $\times$, $R=0$	0.488	0.668	0.712	2.943	58.70	20.41	46.52	44.80	41.90	45.77
Tokenizer	PQ-VAE	0.534	0.716	0.804	2.920	60.03	20.97	46.35	44.06	41.14	46.84
Tokenizer	DVQ (Ours)	0.533	0.729	0.817	2.680	65.65	25.88	51.32	59.71	49.03	53.48
Projection	Linear	0.486	0.665	0.754	4.200	54.34	16.73	40.32	33.54	30.19	39.77
Projection	Sparse (Ours)	0.533	0.729	0.817	2.680	65.65	25.88	51.32	59.71	49.03	53.48
Regularization	Pairwise Cosine Dist.	0.513	0.691	0.781	3.070	59.34	20.75	45.73	43.71	41.53	46.20
Regularization	Orthogonal (Ours)	0.533	0.729	0.817	2.680	65.65	25.88	51.32	59.71	49.03	53.48
Ortho. Weight	Init $\checkmark$, $R = 0$	0.525	0.721	0.812	2.820	57.30	19.90	47.40	47.40	42.90	47.23
Ortho. Weight	Init $\checkmark$, $R = 10^{-4}$	0.530	0.727	0.819	2.640	63.02	24.41	49.14	55.47	44.54	50.96
Ortho. Weight	Init $\checkmark$, $R = 10^{-3}$	0.530	0.727	0.819	2.650	63.02	22.45	49.13	55.47	44.54	50.96
Ortho. Weight	Init $\checkmark$, $R = 10^{-2}$	0.533	0.729	0.817	2.680	65.65	25.88	51.32	59.71	49.03	53.48
Ortho. Weight	Init $\checkmark$, $R = 10^{-1}$	0.540	0.742	0.831	2.670	60.11	21.94	49.04	51.87	43.63	49.50
Ortho. Weight	Init $\checkmark$, $R = 1$	0.530	0.727	0.819	2.650	63.02	24.41	49.17	55.47	44.54	50.97
Ortho. Weight	Init $\times$, $R = 10^{-2}$	0.509	0.699	0.804	2.800	63.66	24.29	47.14	50.69	41.49	49.06

Table 5: Backbone transfer on HumanML3D using the adaptation strategy specified for each language backbone.

Backbone	Adapt.	R@1	R@2	R@3	MMDist↓	BLEU-1↑	BLEU-4↑	R-L↑	CIDEr↑	BERTScore↑	Avg.↑
GPT-2	Full	0.533	0.729	0.817	2.680	65.65	25.88	51.32	59.71	49.03	53.48
Qwen3 0.6B	LoRA	0.376	0.550	0.656	3.980	62.06	22.45	47.65	45.98	42.43	45.55
Llama 3.2 1B	LoRA	0.444	0.644	0.740	3.220	63.80	24.18	49.32	50.60	45.45	49.05

On KIT-ML (Plappert et al., 2016) (Table 2), GeoMotionGPT has the highest Average (41.71), **14.4%** above MotionGPT. It also has the best R@2, R@3, MMDist, and CIDEr in the table. MotionGPT remains better on BLEU-4 and BERTScore, and TM2T ties our R@1 while leading on ROUGE-L.

Retrieval and captioning trade-off. The two metric families move differently across datasets. On HumanML3D our recalls sit marginally below MoTe and MotionGPT3 while captioning improves by a wide margin; on KIT-ML the pattern partly inverts. We therefore do not claim that GeoMotionGPT dominates every metric against every baseline, and cross-method comparisons of individual metrics are in any case only partially informative, since different designs respond differently to dataset characteristics and emphasize retrieval-style alignment versus generation quality. The controlled ablations within one framework provide cleaner evidence than the cross-paper aggregate. Replacing any core component reduces the aggregate on both datasets, although individual metrics do not always move together (Section 4.6).

4.5 Transfer across LLM Backbones

Table 5 identifies GPT-2 with full fine-tuning as the strongest tested configuration, but the compar-

Table 5: Controlled Qwen3 0.6B comparison on HumanML3D with identical LoRA adaptation. *Baseline* removes all proposed components.

Setting	R@1	R@3	MMDist↓	BLEU-4↑	CIDEr↑	Avg.↑
Baseline	0.348	0.588	4.490	10.81	21.02	29.96
GeoMotionGPT	0.376	0.656	3.980	22.45	45.98	45.55

ison is not controlled: the two larger backbones use LoRA because of our compute budget. Its single-column Qwen3 panel holds the backbone and LoRA settings fixed and varies only our components. Under this matched setup, GeoMotionGPT raises Average from 29.96 to 45.55, a **52.0%** relative increase. Retrieval improves (R@1 0.348 $\rightarrow$ 0.376; MMDist 4.490 $\rightarrow$ 3.980), and captioning improves as well (CIDEr 21.02 $\rightarrow$ 45.98). This controlled result supports transfer beyond GPT-2 under LoRA, but it does not establish how the backbones would rank under identical full fine-tuning.

4.6 Ablation Studies

Removing every proposed component. The *Baseline* row of Table 4 ablates all of our designs at once, using neither geometry-preserving initialization nor orthogonal regularization. It reaches an Average of 45.77, while the full model reaches 53.48, a relative increase of **16.8%**. Retrieval and captioning metrics both improve, although the gains vary by metric. The same holds on KIT-ML, where the

Table 6: Representative motion-to-text outputs. Blue text marks details recovered by GeoMotionGPT. The first output also adds an unsupported picking action, and the third omits the rightward part of the reference.

GT Motions				
Text Description	A man takes a step forward, takes his left arm and moves it right to left then takes a step back.	A person jumps to his left.	This person stumbles left and right while moving forward.	A person claps their hands.
VQ-VAE Predicted Text	A person walking forward and then bending down to pick up something	a person jumps and lands.	a person walks forward and then walks sideways to the left.	a person juggles two balls with their hands.
Ours	A person steps forward, picks something up with their left hand, and then steps back.	A person jumps sideways to the left.	A person walks forward and then stumbles to the left.	Person is clapping their hands.

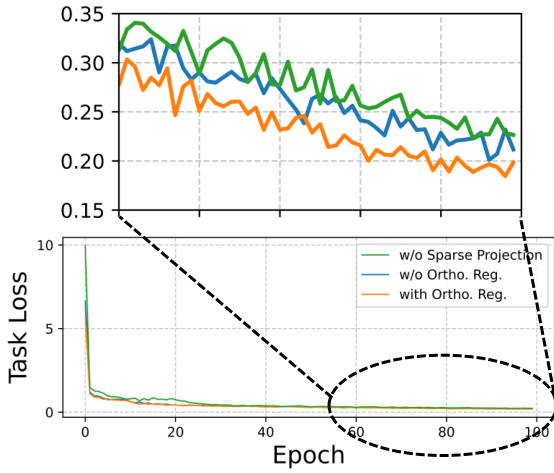


Figure 3: Stage-2 training loss with the full method, without orthogonality loss, and without sparse projection initialization.

corresponding baseline reaches 36.52 against 41.71 (Table 7).

Impact of Tokenizer Design. Replacing PQ-VAE (Hong et al., 2025) with DVQ improves caption quality and alignment together (CIDEr: 44.06 $\rightarrow$ 59.71, MMDist: 2.920 $\rightarrow$ 2.680). R@1 is essentially unchanged (0.534 vs. 0.533) while R@2 and R@3 both improve, and Average rises by **14.2%** (46.84 $\rightarrow$ 53.48).

Impact of Sparse Projection. Compared with learned linear projection, sparse projection improves MMDist from 4.20 to 2.68 and CIDEr from 33.54 to 59.71. Average increases by **34.5%** (39.77 $\rightarrow$ 53.48), showing that the initialization choice matters in this setting.

Impact of Regularization Method. Replacing pairwise cosine-distance regularization (Wang et al., 2018) with orthogonality regularization improves every metric reported in the HumanML3D ablation, including MMDist (3.07 $\rightarrow$ 2.68) and CIDEr (43.71 $\rightarrow$ 59.71). Average rises by **15.8%** (46.20 $\rightarrow$ 53.48). The ordering also holds on KIT-ML (38.32 $\rightarrow$ 41.71), though there the cosine variant keeps an edge on pure text-overlap metrics (Table 7).

Impact of Sparse Projection-Based Initialization. Under the same orthogonal-loss ratio (10^{-2}), replacing sparse projection with stochastic initialization reduces Average by **8.3%** (53.48 $\rightarrow$ 49.06). Figure 3 is consistent with this trend: without sparse projection, training converges more slowly and remains at higher loss, indicating a less favorable optimization start.

Effect of Orthogonal Loss Ratio. Adding orthogonal regularization substantially improves over the no-orthogonal baseline, with the best ratio (10^{-2}) raising Average by **13.2%**. A stronger ratio (10^{-1}) gives the highest recalls but degrades captioning quality, and Figure 3 shows faster, lower-loss optimization with the regularizer. Across the tested range from 10^{-4} to 1, Average stays between 49.50 and 53.48. The coefficient is therefore not highly sensitive within this single-run sweep, and 10^{-2} is also best among the tested KIT-ML settings.

Cross-dataset ablation on KIT-ML. Table 7 reproduces the component ordering on a smaller dataset. Replacing DVQ with PQ-VAE reduces

Table 7: Ablation on KIT-ML under GPT-2, corresponding to Table 4. *Baseline* removes all proposed components. Each tested replacement reduces the descriptive Average.

Study	Setting	R@1	R@2	R@3	MMDist↓	BLEU-1↑	BLEU-4↑	R-L↑	CIDEr↑	BERTScore↑	Avg.↑
Baseline	Init $\mathbf{X}$, $R=0$	0.140	0.269	0.380	6.048	49.38	19.85	44.02	45.56	34.00	36.52
Tokenizer	PQ-VAE	0.121	0.214	0.280	7.893	45.42	15.91	43.25	37.98	34.85	32.99
Projection	Linear	0.074	0.134	0.191	9.159	41.89	14.59	41.60	31.69	33.83	29.48
Regularization	Pairwise Cosine Dist.	0.156	0.291	0.389	5.959	49.76	20.66	46.41	47.00	38.24	38.32
Ortho. Weight	Init $\checkmark$, $R = 0$	0.322	0.511	0.620	4.069	44.37	15.05	40.14	54.12	35.54	39.61
Ortho. Weight	Init $\checkmark$, $R = 10^{-2}$ (Ours)	0.328	0.574	0.676	3.565	44.11	15.69	42.48	58.81	36.57	41.71
Ortho. Weight	Init $\checkmark$, $R = 1$	0.282	0.453	0.565	4.521	43.32	16.33	41.10	55.30	33.83	38.87

Table 8: Geometry and stability diagnostics. **Top:** mean absolute pairwise cosine similarity (PCS); lower is closer to orthogonal. **Bottom:** removing the straight-through estimator (STE) from Stage-1 training.

Token space	Mean PCS ↓	
VQ-VAE codebook	9.62×10^{-2}	
DVQ codebook (ours)	4.51×10^{-4}	
GPT-2 text embeddings	2.68×10^{-1}	
Motion-token emb. (ours, tuned)	2.24×10^{-3}	
Stage-1 statistic	w/o STE	with STE
Best total val. loss↓	0.7849	0.5501
Final val. rec. loss↓	0.9908	0.3589
Code utilization↑	36.91%	100.00%
Dead codes↓	323/512	0/512
Max code count↓	50,482	6,300

Average from 41.71 to 32.99; learned projection gives 29.48, pairwise cosine regularization gives 38.32, and removing all components gives 36.52. The 10^{-2} orthogonal-loss ratio also outperforms both no regularization (39.61) and the over-strong ratio of 1 (38.87). The advantage is not uniform: pairwise cosine regularization remains better on the text-overlap metrics, whereas orthogonality is stronger on retrieval and CIDEr. We therefore treat KIT-ML as a directional cross-dataset check rather than an independent precision estimate.

Geometry and stability diagnostics. Two further checks directly measure the learned geometry and the stability of the discrete training path.

Table 8 (top) shows that the DVQ codebook is over two orders of magnitude closer to orthogonal than the VQ-VAE codebook (4.51×10^{-4} versus 9.62×10^{-2}). After sparse-projection initialization and Stage-2 fine-tuning, the motion-token embeddings retain a mean PCS of 2.24×10^{-3} . These values confirm approximate orthogonality within each space, but they do not measure one-to-one correspondence between tuned embeddings and their Stage-1 code vectors.

The lower panel isolates the STE while keep-

ing the rest of the Gumbel-Softmax pipeline fixed. Without the STE, validation reconstruction loss initially falls to 0.2991 but rises to 0.9908 by the final epoch, and only 189 of 512 codes remain active. With the STE, all codes remain active, the final reconstruction loss is 0.3589, and the maximum code count falls from 50,482 to 6,300. The biased STE gradient is therefore important here because its hard forward path matches the discrete representation used at inference. Qualitative token sequences remain in Appendix A.6.

4.7 Case Study

Table 6 compares the VQ-VAE baseline and GeoMotionGPT on representative motion sequences. GeoMotionGPT recovers the leftward direction in the second example and the clapping action in the fourth, which the VQ-VAE baseline misses. The comparison also exposes remaining errors: the first GeoMotionGPT output adds a picking action that is absent from the reference, and the third captures only the leftward part of a left-and-right stumble. These examples illustrate behavior but do not establish a causal link between code usage and caption quality. Token-level usage patterns and a second case study are provided in Appendix A.6.

5 Conclusion

We presented GeoMotionGPT, which gives a motion codebook and its LLM motion-token embeddings a shared orthogonality prior. The method combines decoder-only DVQ, sparse isometric initialization, and regularization in both training stages. Under GPT-2, it raises a descriptive aggregate of retrieval and captioning metrics by **22.4%** on HumanML3D and **14.4%** on KIT-ML relative to the strongest reported baselines. Controlled ablations support the contribution of each component. Geometry diagnostics show low pairwise cosine similarity in the Stage-1 codebook and the tuned Stage-2 motion-token block; they do not establish direct semantic alignment with text embeddings.

Limitations

Our ablations cover tokenizer design, projection strategy, regularization form, and orthogonal-loss ratio, but remain centered on our own components; we have not compared against other geometry-aware objectives such as whitening-, spectral-, or entropy-based constraints.

Our strongest results use fully fine-tuned GPT-2, while Qwen3 0.6B and Llama 3.2 1B are adapted with LoRA under our compute budget. The cross-backbone comparison therefore confounds model capacity with adaptation budget. Holding both fixed, the method improves Qwen3 0.6B (Section 4.5), but we have not tested full fine-tuning of larger instruction-tuned LLMs. All experiments use two English benchmarks with the same skeleton representation, and we evaluate understanding rather than motion generation. Because DVQ trades reconstruction loss for a better-conditioned codebook (Section 4.3), its effect on synthesis remains open.

Our geometric evidence is diagnostic rather than semantic. PCS measures pairwise orthogonality within a space; it does not establish semantic alignment or preservation after fine-tuning. Exact Gram preservation applies only at initialization and requires $K \leq D$; we test only $K = D = 512$. Likewise, full utilization does not imply semantically distinct codes, and the qualitative examples do not replace blinded human evaluation.

The headline Average is a non-standard descriptive aggregate. Several published baselines omit R@2, so their recall term averages R@1 and R@3, whereas rows with complete reporting average R@1, R@2, and R@3. Individual metrics are therefore the primary evidence. We also report single values rather than means and confidence intervals for our main experiments, so the statistical uncertainty of the gains is unknown.

Acknowledgments

We thank the anonymous reviewers for their constructive feedback, which substantially improved the clarity and the empirical scope of this work.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei,

Marianne Monteiro, Jacob L. Menick, Sebastian Borgeaud, and 8 others. 2022. [Flamingo: a visual language model for few-shot learning](#). *Advances in neural information processing systems*, 35:23716–23736.

Javed A Aslam and Emine Yilmaz. 2005. [A geometric interpretation and analysis of r-precision](#). In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 664–671.

Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. 2018. [Can we gain more from orthogonality regularizations in training deep networks?](#) *Advances in Neural Information Processing Systems*, 31.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. [Estimating or propagating gradients through stochastic neurons for conditional computation](#). *arXiv preprint arXiv:1308.3432*.

Hao Chen, Ze Wang, Xiang Li, Ximeng Sun, Fangyi Chen, Jiang Liu, Jindong Wang, Bhiksha Raj, Zicheng Liu, and Emad Barsoum. 2025. [Softqvae: Efficient 1-dimensional continuous tokenizer](#). In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 28358–28370.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, and 3 others. 2023. [PaLM-E: An embodied multimodal language model](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8469–8488. PMLR.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. [The llama 3 herd of models](#). *arXiv preprint arXiv:2407.21783*.

Yanhong Fei, Yingjie Liu, Xian Wei, and Mingsong Chen. 2022. [O-vit: Orthogonal vision transformer](#). *arXiv preprint arXiv:2201.12133*.

Chuan Guo, Yuxuan Mu, Muhammad Gohar Javed, Sen Wang, and Li Cheng. 2024. [Momask: Generative masked modeling of 3d human motions](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1900–1910.

Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. 2022a. [Generating diverse and natural 3d human motions from text](#). In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5152–5161.

Chuan Guo, Xinxin Zuo, Sen Wang, and Li Cheng. 2022b. [Tm2t: Stochastic and tokenized modeling for the reciprocal generation of 3d human motions and](#)

- texts. In *European Conference on Computer Vision*, pages 580–597.
- Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. *beta-vae: Learning basic visual concepts with a constrained variational framework*. In *International conference on learning representations*.
- Fangzhou Hong, Vladimir Guzov, Hyo Jin Kim, Yuting Ye, Richard Newcombe, Ziwei Liu, and Lingni Ma. 2025. *Egoclm: Multi-modal language model of egocentric motions*. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5344–5354.
- Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. *LoRA: Low-rank adaptation of large language models*. In *International Conference on Learning Representations*.
- Lei Huang, Li Liu, Fan Zhu, Diwen Wan, Zehuan Yuan, Bo Li, and Ling Shao. 2020. *Controllable orthogonalization in training dnns*. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6429–6438.
- Lei Huang, Xianglong Liu, Bo Lang, Adams Yu, Yongliang Wang, and Bo Li. 2018. *Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks*. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.
- Eric Jang, Shixiang Gu, and Ben Poole. 2017. *Categorical reparameterization with gumbel-softmax*. In *International Conference on Learning Representations*.
- Kui Jia, Dacheng Tao, Shenghua Gao, and Xiangmin Xu. 2017. *Improving training of deep neural networks via singular value bounding*. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4344–4352.
- Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. 2023. *Motiongpt: Human motion as a foreign language*. *Advances in Neural Information Processing Systems*, 36:20067–20079.
- Diederik P Kingma and Max Welling. 2014. *Auto-encoding variational bayes*. In *2nd International Conference on Learning Representations*.
- Zengyuan Lai, Jiarui Yang, Songpengcheng Xia, Lizhou Lin, Lan Sun, Renwen Wang, Jianran Liu, Qi Wu, and Ling Pei. 2025. *Radarllm: Empowering large language models to understand human motion from millimeter-wave point cloud sequence*. *Preprint*, arXiv:2504.09862.
- Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. 2022. *Autoregressive image generation using residual quantization*. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11523–11532.
- José Lezama, Qiang Qiu, Pablo Musé, and Guillermo Sapiro. 2018. *Ole: Orthogonal low-rank embedding-a plug and play geometric loss for deep learning*. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8109–8118.
- Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. *Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models*. In *International conference on machine learning*, pages 19730–19742. PMLR.
- Zhe Li, Weihao Yuan, Yisheng He, Lingteng Qiu, Shenhao Zhu, Xiaodong Gu, Weichao Shen, Yuan Dong, Zilong Dong, and Laurence T Yang. 2024. *Lamp: Language-motion pretraining for motion generation, retrieval, and captioning*. *arXiv preprint arXiv:2410.07093*.
- Chin-Yew Lin. 2004. *Rouge: A package for automatic evaluation of summaries*. In *Text summarization branches out*, pages 74–81.
- Fabian Mentzer, David Minnen, Eiríkur Agustsson, and Michael Tschannen. 2024. *Finite scalar quantization: VQ-VAE made simple*. In *The Twelfth International Conference on Learning Representations*.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. *Bleu: a method for automatic evaluation of machine translation*. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.
- Jeongeun Park, Sungjoon Choi, and Sangdoo Yun. 2025. *A unified framework for motion reasoning and generation in human interaction*. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10698–10707.
- Mathis Petrovich, Michael J Black, and Gül Varol. 2023. *Tmr: Text-to-motion retrieval using contrastive 3d human motion synthesis*. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9488–9497.
- Matthias Plappert, Christian Mandery, and Tamim Asfour. 2016. *The kit motion-language dataset*. *Big data*, 4(4):236–252.
- Haozhi Qi, Chong You, Xiaolong Wang, Yi Ma, and Jitendra Malik. 2020. *Deep isometric learning for visual recognition*. In *International conference on machine learning*, pages 7824–7835. PMLR.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. *Language models are unsupervised multitask learners*. *OpenAI blog*, 1(8):9.

- Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. 2019. [Generating diverse high-fidelity images with vq-vae-2](#). *Advances in neural information processing systems*, 32.
- Simone Ricci, Niccolò Biondi, Federico Pernici, Ioannis Patras, and Alberto Del Bimbo. 2025. [\$\lambda\$ -orthogonality regularization for compatible representation learning](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Guy Tevet, Brian Gordon, Amir Hertz, Amit H Bermano, and Daniel Cohen-Or. 2022. [Motionclip: Exposing human motion generation to clip space](#). In *European Conference on Computer Vision*, pages 358–374. Springer.
- Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. [Neural discrete representation learning](#). *Advances in neural information processing systems*, 30.
- Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. [Cider: Consensus-based image description evaluation](#). In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.
- Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. 2018. [Cosface: Large margin cosine loss for deep face recognition](#). In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5265–5274.
- Yuan Wang, Di Huang, Yaqi Zhang, Wanli Ouyang, Jile Jiao, Xuetao Feng, Dan Xu, and Shixiang Tang. 2024. [Motiongpt-2: A general-purpose motion-language model for motion generation and understanding](#). *arXiv preprint arXiv:2410.21747*.
- Bizhu Wu, Jinheng Xie, Keming Shen, Zhe Kong, Jianfeng Ren, Ruibin Bai, Rong Qu, and Linlin Shen. 2025. [Mg-motionllm: A unified framework for motion comprehension and generation across multiple granularities](#). In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 27849–27858.
- Qi Wu, Yubo Zhao, Yifan Wang, Xinhang Liu, Yu-Wing Tai, and Chi-Keung Tang. 2024a. [Motion-agent: A conversational framework for human motion generation with llms](#). *arXiv preprint arXiv:2405.17013*.
- Yiming Wu, Wei Ji, Kecheng Zheng, Zicheng Wang, and Dong Xu. 2024b. [Mote: Learning motion-text diffusion model for multiple generation tasks](#). *arXiv preprint arXiv:2411.19786*.
- Di Xie, Jiang Xiong, and Shiliang Pu. 2017. [All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation](#). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6176–6185.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. [Qwen3 technical report](#). *arXiv preprint arXiv:2505.09388*.
- Aston Zhang, Alvin Chan, Yi Tay, Jie Fu, Shuohang Wang, Shuai Zhang, Huajie Shao, Shuochao Yao, and Roy Ka-Wei Lee. 2021. [On orthogonality constraints for transformers](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, volume 2, pages 375–382. Association for Computational Linguistics.
- Baoquan Zhang, Huaibin Wang, Chuyao Luo, Xutao Li, Guotao Liang, Yunming Ye, Xiaochen Qi, and Yao He. 2024. [Codebook transfer with part-of-speech for vector-quantized image modeling](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7757–7766.
- Jianrong Zhang, Yangsong Zhang, Xiaodong Cun, Shaoli Huang, Yong Zhang, Hongwei Zhao, Hongtao Lu, and Xi Shen. 2023. [T2m-gpt: Generating human motion from textual descriptions with discrete representations](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. [Bertscore: Evaluating text generation with bert](#). *arXiv preprint arXiv:1904.09675*.
- Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. 2023. [Learning fine-grained bimanual manipulation with low-cost hardware](#). *arXiv preprint arXiv:2304.13705*.
- Zixiang Zhou, Yu Wan, and Baoyuan Wang. 2024. [Avatargpt: All-in-one framework for motion understanding planning generation and beyond](#). In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1357–1366.
- Bingfan Zhu, Biao Jiang, Sunyi Wang, Shixiang Tang, Tao Chen, Linjie Luo, Youyi Zheng, and Xin Chen. 2025. [Motiongpt3: Human motion as a second modality](#). *arXiv preprint arXiv:2506.24086*.
- Lei Zhu, Fangyun Wei, Yanye Lu, and Dong Chen. 2024. [Scaling the codebook size of vq-gan to 100,000 with a utilization rate of 99%](#). *Advances in Neural Information Processing Systems*, 37:12612–12635.

A Appendix

A.1 Additional Implementation Details

This section provides implementation details that complement the training protocol in Section 4.1.

Backbone Adaptation. We evaluate three language backbones of increasing size: GPT-2, Qwen3 0.6B, and Llama 3.2 1B. GPT-2 uses full-parameter fine-tuning: all transformer weights are updated together with the newly added motion-token embedding block, while the original text-token embeddings remain frozen as described in Section 3.4. Qwen3 0.6B and Llama 3.2 1B are adapted with LoRA (Hu et al., 2022) using rank 16 and scaling factor 32; the base weights stay frozen, and only the low-rank adapters and the motion-token embedding block receive gradient updates. This split reflects our compute budget rather than a methodological requirement. It also motivates the controlled comparison in Table 5, which holds the Qwen3 backbone and its LoRA configuration fixed and varies only our proposed components, so that the effect of the method is not confounded with the adaptation strategy. For LLM fine-tuning, the cosine annealing scheduler uses $T_{\max} = 200$ and $\eta_{\min} = 10^{-6}$.

Data Preprocessing and Token Granularity. HumanML3D motions are sampled at 20 FPS and contain 20 to 200 frames after preprocessing; each frame is represented by the standard 263-dimensional feature vector, and each motion is paired with three text captions. KIT-ML is processed with the same pipeline, using its 251-dimensional frame features, and provides a smaller,

more diverse cross-dataset check. Motion features are z-normalized with the training-set mean and standard deviation before entering the tokenizer. DVQ operates on temporally downsampled motion features: three stride-2 convolutional stages shorten the sequence by a factor of 8 before quantization, so each discrete token summarizes a window of 8 frames, or 0.4 seconds at 20 FPS, rather than a single pose. A maximum-length motion of 200 frames therefore maps to 25 tokens. This reduces the length of the token sequences consumed by the language model, which lowers the fine-tuning cost and keeps motion-token sequences at a length scale comparable to their paired captions.

Tokenizer Architecture. The DVQ encoder is a one-dimensional convolutional network. An input convolution lifts the frame features to 512 channels; three downsampling stages follow, each consisting of a stride-2 convolution and a residual block of depth 3 with dilation growth rate 3; a final convolution maps the 512 channels to $K = 512$ logits per temporal position. The encoder therefore emits the assignment logits $\mathbf{z} = Q(x)$ of Eq. 4 directly, and token assignment involves no distance computation against the codebook. The decoder mirrors the encoder with nearest-neighbor upsampling and convolutions, and its input consists exclusively of 512-dimensional code embeddings. Stage-1 training uses randomly cropped 32-frame windows, corresponding to 4 tokens per training clip; full sequences are tokenized at inference.

DVQ Optimization. DVQ uses a 512-code, 512-dimensional codebook trained for 500 epochs with batch size 512 under AdamW. The base learning

Table 9: Token sequences for four ground-truth motions. DVQ uses a wider variety of token IDs in these examples; token IDs alone do not establish temporal coherence or semantic quality.

GT Motions				
Text Description	Person leans forward slightly and moves right hand in a wiping motion.	The person drinks from the big jug.	The man takes a step and bends raising a foot to wipe a table.	A person rests their hands on their knees while squatting.
VQ-VAE Motion Token IDs	330 330 330 330 330 330 330 330 330 330 330 330 330 330 287 287 287 287 287 330 330 330 330 288	493 243 243 243 243 243 243 243 243 243 248 28 10 10 10 173 173 280 119 128 153 153 153 153	41 499 276 411 17 17 17 17 17 17 17 17 17 17 59 229 229 65 65 65 65 65 41	296 296 296 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27 27
DVQ Token IDs	379 19 379 177 177 343 189 225 385 330 343 343 177 343 177 19 352 330 414 385 12 385 385 414	70 185 70 70 70 384 70 457 70 70 70 504 223 82 200 495 457 296 51 206 37 254 500 484	337 232 156 345 370 19 370 370 19 370 370 370 370 38 38 358 421 284 219 500 72 55 219	165 235 11 324 59 507 298 24 24 24 11 428 284 405 463 38 284 324 211 139 469 328 9 69

rate is 2×10^{-4} with 3% linear warmup, and the cosine schedule described in the main text decays the learning rate to zero over the remaining epochs. We apply weight decay of 10^{-4} to non-bias, non-normalization parameters and no weight decay to bias terms, normalization layers, or quantizer parameters. Quantizer parameters, including the codebook, use the reduced learning rate of 10^{-4} (half the base rate), which keeps codebook updates conservative relative to the rest of the network while the assignment distribution is still changing. The codebook is initialized from a Gaussian with standard deviation 0.02, matching the scale of standard transformer embedding initialization. After training, we retain the checkpoint with the lowest total validation loss rather than the final-epoch weights, and this checkpoint is used both for reporting Stage-1 statistics and for tokenizing the corpus consumed in Stage 2.

Temperature and Utilization Schedules. Two schedules govern the discrete path during Stage-1 training. First, the Gumbel-Softmax temperature τ stays at 0.4 for the first 300 epochs, decays exponentially to 0.01 over the next 100 epochs, and then remains fixed. A moderate early temperature keeps the relaxed assignments informative enough for gradients to reach many codes, while the late low-temperature phase sharpens the relaxation toward the hard assignments used at inference. Second, the utilization statistic $\mathbf{q}$ in Eq. 6 is computed from soft assignment probabilities for the first 150 epochs; a hardness coefficient then increases linearly from 0 to 1 over the following 50 epochs, after which the statistic is computed from hard assignments. In the implementation, $\mathbf{q}$ is the mini-batch mean of the assignment probabilities over all temporal positions, and the objective is expressed through the normalized perplexity $\exp(H(\mathbf{q}))/K$, a monotone transform of the entropy in Eq. 6; the hardness coefficient interpolates between the perplexity of the relaxed assignments and that of the straight-through hard assignments. This soft-to-hard transition lets the utilization loss spread probability mass across the codebook early in training, and later aligns the regularized statistic with the actual discrete code usage. Training uses stochastic Gumbel-Softmax samples throughout, whereas inference applies deterministic $\arg \max$ tokenization.

Regularization Coefficients. Unless otherwise stated, the Stage-1 objective uses $\lambda_{\text{ortho}} = 10^{-2}$ and $\lambda_{\text{util}} = 10^{-2}$, and the Stage-2 instruction-

tuning objective uses $\lambda'_{\text{ortho}} = 10^{-2}$, which is the setting denoted *Init ✓*, $R = 10^{-2}$ in Table 4. As reported in Section 4.6, downstream performance is stable across four orders of magnitude of this coefficient, with the descriptive Average staying between 49.50 and 53.48 over the tested range from 10^{-4} to 1 on HumanML3D. The coefficient therefore does not require careful per-dataset tuning, and the same value is used for both HumanML3D and KIT-ML.

Decoder-Only Design Rationale. DVQ’s decoder exclusively consumes code embeddings, so no continuous latent path can bypass the codebook during reconstruction. All information available to the decoder passes through the discrete code selection, so the reconstruction gradient must shape the code vectors themselves instead of flowing through a continuous shortcut around them. Combined with the Gumbel-Softmax relaxation, this design allows the reconstruction, utilization, and orthogonality terms to be optimized jointly in a single computation graph, without the surrogate codebook and commitment losses used by standard VQ-VAE training.

A.2 Gumbel-Softmax and Straight-Through Details

For completeness, we provide the detailed definition of the Gumbel-Softmax operator (Jang et al., 2017) used in DVQ, which is omitted from the main text. The $\text{GumbelSoftmax}(\cdot; \cdot)$ used in Eq. 4 is defined as:

$$\text{GumbelSoftmax}(\mathbf{z}; \tau) = \text{softmax}\left(\frac{\mathbf{z} + \mathbf{g}}{\tau}\right), \quad (9)$$

where $\mathbf{g}$ is sampled element-wise from the $\text{Gumbel}(0, 1)$ distribution via $g_k = -\log(-\log u_k)$ with $u_k \sim \text{Uniform}(0, 1)$, and τ is the temperature parameter controlling the smoothness of the categorical distribution. As $\tau \rightarrow 0$, the output approaches a one-hot vector, recovering discrete sampling from the categorical distribution defined by the logits; as τ grows, the output flattens toward a uniform mixture.

With the straight-through estimator (Bengio et al., 2013), the forward pass discretizes the relaxed sample,

$$\mathbf{y}_{\text{hard}} = \text{one-hot}\left(\arg \max_k y_{\text{soft},k}\right), \quad (10)$$

while the backward pass propagates gradients through $\mathbf{y}_{\text{soft}}$ as if the discretization were the identity map. The resulting gradient is biased, but the

forward computation matches the hard, discrete representation used at inference. In practice this property is decisive: as Table 8 reports, removing the straight-through estimator while keeping the rest of the pipeline fixed destabilizes late training (final validation reconstruction loss 0.9908 versus 0.3589) and collapses utilization, with 323 of 512 codes left unused. Gumbel noise is injected only during training; at inference, tokenization is a deterministic $\arg \max$ over the logits, so the discrete motion-token sequence for a given input is unique and reproducible.

A.3 Additional Discussion of the Sparse Projection

Three properties of the sparse projection in Section 3.2 are worth stating explicitly.

First, it adds no learnable parameters. The projection matrix $\mathbf{P} \in \{0, 1\}^{D' \times D}$ is fully determined by the index set $\mathcal{I}$ of D unique target rows; no weights are trained, and applying $\mathbf{P}$ amounts to scattering the D code coordinates into D fixed positions of a D' -dimensional zero vector. Its memory and compute overhead are therefore negligible compared with a learned dense map.

Second, the guarantee is independent of the random choice of $\mathcal{I}$. Any selection of D unique rows yields orthonormal columns and hence $\mathbf{P}^\top \mathbf{P} = \mathbf{I}_D$, so the transferred Gram matrix, and with it the exact degree of approximate orthogonality attained in Stage 1, does not depend on the sampled indices. The initialization is consequently insensitive to this source of randomness, in contrast to a learned linear projection whose outcome depends on its own optimization trajectory. The ablation in Table 4 quantifies this difference: replacing the sparse projection with a learned linear projection reduces the descriptive Average from 53.48 to 39.77 on HumanML3D, and the corresponding KIT-ML replacement is the weakest tested variant (Table 7).

Third, the projection is an initialization mechanism only. After Stage 2 fine-tuning begins, the motion-token embeddings are free parameters, and the equality between the codebook Gram matrix and the embedding Gram matrix holds only at the first step. The Stage-2 orthogonality term then acts as a soft geometric prior rather than a constraint tying each embedding to its originating code; Table 8 shows that the tuned embeddings nevertheless retain a mean absolute pairwise cosine similarity of 2.24×10^{-3} , far below that of the surrounding GPT-2 text embeddings (2.68×10^{-1}).

A.4 Motion Vocabulary and Instruction Format

Stage 2 represents motions as text-like token strings. The LLM vocabulary is extended with $K + 3$ special tokens: `<motion_id_0>` through `<motion_id_511>` correspond to the DVQ codes, and three additional tokens mark the start of a motion span, the end of a motion span, and a mask placeholder reserved for masked-completion prompts. For GPT-2, these tokens are appended after the original 50,257-entry text vocabulary, and their embeddings form the motion-token block $\mathbf{E} \in \mathbb{R}^{K \times D'}$ of Section 3.4 with $D' = 768$. A tokenized motion is rendered as the start-of-motion token, the sequence of its code tokens, and the end-of-motion token, in that order.

The motion-to-text task uses a fixed instruction template: the input is `Generate text: <Motion_Placeholder>`, where the placeholder is replaced by the rendered motion string, and the target output is one of the reference captions. Training minimizes the autoregressive cross-entropy of Eq. 8 on the target caption. Consistent with Section 3.4, the original text embeddings stay frozen during fine-tuning; only the motion-token embedding block and the trainable LLM parameters receive gradients.

A.5 Evaluation Details and Compute

We follow the motion-to-text evaluation protocol of MotionGPT3 (Zhu et al., 2025). The retrieval metrics (R-Precision at ranks 1–3 and MMDist) assess how well the generated caption aligns with its source motion in a shared text-motion embedding space, whereas the captioning metrics (BLEU-1, BLEU-4 (Papineni et al., 2002), ROUGE-L (Lin, 2004), CIDEr (Vedantam et al., 2015), and BERTScore (Zhang et al., 2019)) compare the generated caption against the reference captions, of which HumanML3D provides three per motion. The two families probe different failure modes: a caption can be lexically close to a reference while retrieving poorly, and vice versa, which is why Section 4.4 reports both and treats the aggregate Average as descriptive only. As noted in the main text, several published baselines omit R@2, so their recall term averages R@1 and R@3 while fully reported rows average all three cutoffs; this asymmetry is a further reason not to read the aggregate as a benchmark measure.

Unless otherwise specified, each experiment

runs on a single NVIDIA B200 GPU. Stage-1 DVQ training uses 500 epochs at batch size 512, and Stage-2 motion understanding uses 100 epochs at batch size 320 with learning rate 10^{-4} , weight decay 10^{-2} , and cosine scheduling. We report single-run results without confidence intervals, a limitation discussed in the Limitations section.

A.6 Qualitative Case Study on Codebook Usage

Table 9 lists the discrete token sequences produced by the VQ-VAE baseline and by DVQ for four ground-truth motions, alongside their reference captions. The examples were chosen to span different motion types: a localized upper-body action (wiping), an object-directed action (drinking), a compound locomotion-and-manipulation sequence (stepping and wiping a table), and a largely static pose (squatting with hands on knees).

Two patterns are visible. First, the VQ-VAE baseline tends to assign long runs of a single code. In the first example, code 330 covers 18 of the 24 positions and the entire sequence uses only three distinct codes; in the fourth example, a single code (ID 27) accounts for 21 of 24 positions. DVQ tokenizes the same motions with a wider variety of code IDs, using 11 and 19 distinct codes on these two sequences, respectively. Second, the contrast is smallest where the motion itself is varied: on the drinking and table-wiping examples, the VQ-VAE sequences already switch codes several times, and DVQ mainly redistributes usage rather than adding diversity.

These per-example observations are consistent with the aggregate usage statistics in Table 3, where DVQ shows a lower maximum code count, standard deviation, coefficient of variation, and Gini coefficient than the VQ-VAE baseline under the identical codebook size.

We emphasize the limits of this evidence. Token IDs alone do not establish temporal coherence or semantic quality: a long run of one code can be an appropriate encoding for a near-static segment, such as the held squat in the fourth example, and a diverse sequence is not automatically a better one. The concern raised by the baseline is therefore not repetition per se, but the concentration of assignments onto few codes across many different motions, which the dispersion statistics of Section 4.3 measure directly. We accordingly treat these sequences as illustrations of the tokenizers’ qualitative behavior, and rely on the aggregate distribution

statistics and the downstream understanding results of Section 4.4 for quantitative conclusions.