Bailando++: 3D Dance GPT with Choreographic Memory

Li Siyao, Weijiang Yu, Tianpei Gu, Chunze Lin, Quan Wang, Chen Qian,
Chen Change Loy, *Senior Member*, Ziwei Liu, *Member, IEEE*

Abstract—Our proposed music-to-dance framework, *Bailando*++, addresses the challenges of driving 3D characters to dance in a way that follows the constraints of choreography norms and maintains temporal coherency with different music genres. *Bailando*++ consists of two components: a choreographic memory that learns to summarize meaningful dancing units from 3D pose sequences, and an actor-critic Generative Pre-trained Transformer (GPT) that composes these units into a fluent dance coherent to the music. In particular, to synchronize the diverse motion tempos and music beats, we introduce an actor-critic-based reinforcement learning scheme to the GPT with a novel beat-align reward function. Additionally, we consider learning human dance poses in the rotation domain to avoid body distortions incompatible with human morphology, and introduce a musical contextual encoding to allow the motion GPT to grasp longer-term patterns of music. Our experiments on the standard benchmark show that *Bailando*++ achieves state-of-the-art performance both qualitatively and quantitatively, with the added benefit of the unsupervised discovery of human-interpretable dancing-style poses in the choreographic memory. Code and video demo are available at https://github.com/lisiyao21/Bailando/.

Index Terms—Dance Generation, Multi-modal, 3D Human Motion, VQ-VAE, GPT

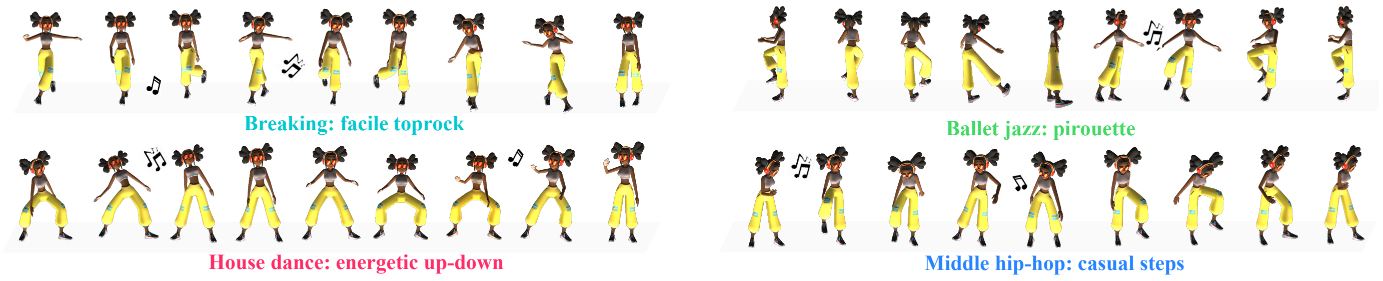


Fig. 1: **Dance examples generated by our proposed method** on various types of music. The character is from Mixamo [1]

1 Introduction

THE task of generating 3D dance sequences that are conditioned on music has practical significance for various real-world applications, such as helping human artists to choreograph and driving the performance of virtual avatars. However, it is still very challenging to produce a satisfactory dancing sequence for a given piece of music due to two main challenges: 1) The spatial quality constraint: not all physically feasible 3D human poses are suitable for dancing. The subset of poses that are suitable for dancing must meet stricter positional standards and be visually expressive and emotionally infectious based on the norms of choreography. 2) Temporal coherency with the music: the generated dancing sequence must be consistent with the rhythm of the music across various genres of beats while maintaining fluidity in the overall movements.

Many existing approaches to dance generation aim to address both of the aforementioned challenges in a single network that directly maps music to a high-dimensional, continuous space of 3D joint sequences [2], [3], [4], [5], [6], [7]. However, such methods often suffer from instability in practice and are prone to producing nonstandard poses that fall outside the dancing subspace, such as freezing or meaningless swaying, due to the lack of explicit constraints on the target domain to ensure spatial quality. To address this issue, some studies collect real dancing clips as dance units and choreograph them by splicing these units together [8], [9]. While this approach guarantees the spatial quality of the generated dance by directly manipulating human-recorded data, it requires a significant amount of manual work to collect the dance units and is not compatible with different rhythms. Additionally, the fixed length and speed of the units make them incompatible with different kinds of music beats.

To address the limitations of current dance generation methods, we propose a novel framework called ***Bailando++*** that consists of two main components designed to address the spatial and temporal challenges, respectively:

(1) First, to address the spatial challenge, we create a finite dictionary of quantized dancing units, known as the choreographic memory, which summarizes the fundamental and reusable components of movements in the dancing-style subspace. Rather than manually selecting these dance units, we use VQ-VAE [10] to encode and quantize 3D joint sequences in an unsupervised manner, resulting in a codebook where each learned code represents a unique dancing pose. To further increase the range of poses that the choreographic memory can represent, we divide 3D poses

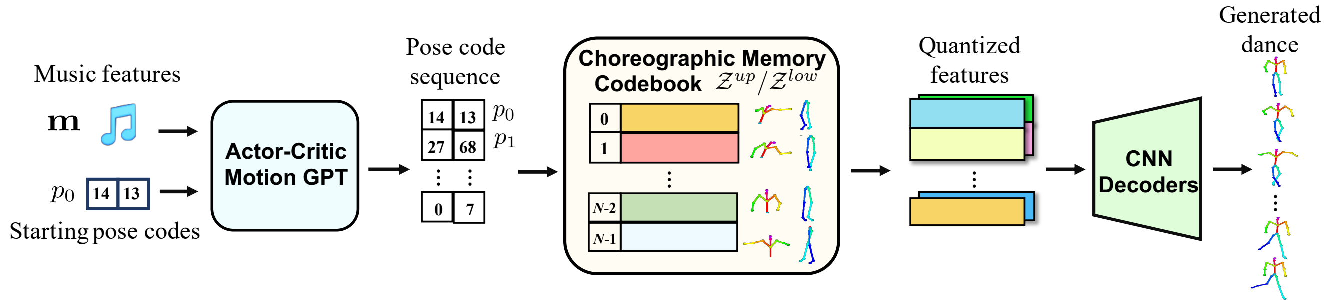


Fig. 2: **Dance generation pipeline of *Bailando*++.** Given a piece of music, an actor-critic motion GPT autoregressively predicts the future upper-lower pose code pairs according to the music features and starting pose codes. The pose code sequence is then embedded to quantized features via a learned choreographic memory and finally decoded into a dance sequence by a CNN-based decoder.

into compositional upper and lower halves of the body and learn separate VQ-VAEs for each half. This allows any piece of dance to be represented as a sequence of paired pose codes.

(2) Second, to generate temporally harmonious dance sequences, we introduce a GPT-like [11] network called motion GPT that translates music and source pose codes into targeted future pose codes. Since the 3D poses are divided into compositional half-bodies in the choreographic memory, we enhance our motion GPT with a proposed cross-conditional causal attention layer to maintain the coherence of the generated body. Additionally, to achieve accurate temporal synchronization between diverse motion tempos and music beats, we use an on-policy reinforcement learning scheme to further improve the performance of motion GPT through actor-critic [12] finetuning with a newly designed beat-align reward function.

The procedure for inferring dance sequences using *Bailando*++ is illustrated in Figure 2. Given a piece of music and a starting pose code pair, the actor-critic GPT autoregressively predicts a sequence of future pose codes. These pose codes are then mapped to corresponding quantized features in the choreographic memory and decoded into a 3D dance sequence using dedicated CNN-based decoders of the learned pose VQ-VAE.

An earlier version (*CVPR 2022, oral*) of this work appears in Siyao *et al.* [13]. In the original *Bailando*, choreography is performed in the domain of 3D joint positions, which leads to two issues when mapping the generated dance sequence to an avatar character. First, the 3D joint positions must be transformed into rotation angles using inverse kinematics (IK) [14], which increases the workload for avatar animation and can introduce errors due to rotational ambiguity (*i.e.*, one set of joint positions may have multiple rotational solutions) or mismatched bone lengths between the source model and the target avatar. As shown in Figure 3(a), when using the original Bailando to drive the Mixamo character [1] to kneel, the feet are bent unnaturally (as indicated by the blue box) after IK. Second, since the generated 3D points have no explicit morphological constraints, the original *Bailando* may result in body distortions that are incompatible with human physiology, such as inconsistent orientations of the upper and lower body (as indicated by the red box in Figure 3(a)).

To address these issues, we have upgraded *Bailando* to generate 3D dance sequences in the domain of joint rotation angles, which are more suitable for avatar animation. Specifically, we generate dancing sequences of joint angles in SPML format [15], which can be directly applied to drive 3D avatars without the need for IK. Since SMPL joint angles are organized in a hierarchical structure (*i.e.*, a joint angle in SMPL format only defines the local rotation of that joint), the orientations of both the upper and lower body are consistent with the direction of the global root (the "Pelvis"), which helps to avoid unreasonable distortions.

The aforementioned improvement cannot be achieved by simply substituting the training data from 3D positions to the SMPL format. As we will discuss in Section 3.2, the hierarchical structure of the SMPL model means that poses that appear different in 3D position space may have little difference when expressed in the rotation domain. For example, when an agent performs a fouette turn (a ballet movement involving spinning in place), the 3D joint positions in different directions are distinct, but the corresponding SMPL expressions are very similar because most joint angles remain the same except for the global root. This ambiguity makes it more difficult for VQ-VAE to summarize spatially representative poses from dance data in the rotation domain, which reduces the space that the choreographic memory can represent and leads to poor performance overall.

To address this problem, we propose a hybrid training strategy for the pose VQ-VAEs that collectively learn from both the 3D position and rotation domains. Specifically, we train a VQ-VAE to quantize pose codes from the 3D position domain and decode them into SMPL format sequences. This strategy not only ensures the spatial representation of the choreographic memory, but also takes advantage of morphological constraints, resulting in higher quality dance generation than models trained solely on 3D rotation data. In addition, we propose a contextual music encoding to improve *Bailando*'s ability to process music features over a longer period of time and produce smoother dance movements. These improvements, along with the other features of Bailando++, make it more effective on the AIST++ standard benchmark than the performance reported by Siyao *et al.* [13].

In summary, our work makes the following contributions: 1) We create a choreographic memory to encode and quantize dancing-style 3D poses using VQ-VAE in an unsupervised manner. 2) We introduce an actor-critic GPT, incorporated with the choreographic memory and cross-conditional causal attention, to align diverse motion tempos with different genres of music beats. 3) We propose a hybrid training strategy that allows the proposed pipeline to generate 3D rotation outputs while maintaining high spatial quality. 4) We introduce a musical contextual encoding to capture long-term music patterns. Extensive experiments show that our proposed *Bailando*++ significantly outperforms existing state-of-the-art

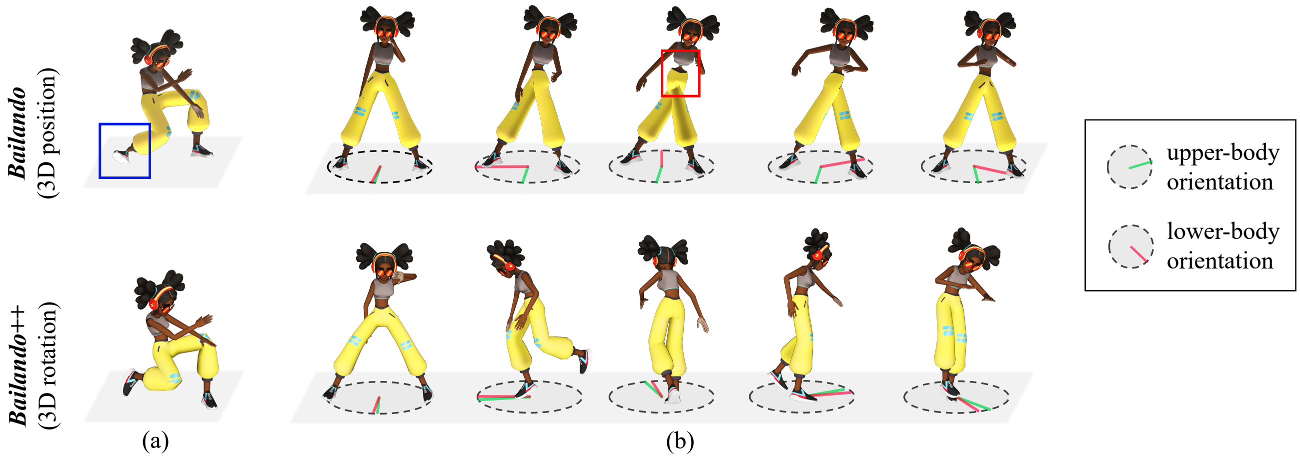


Fig. 3: **Comparison between *Bailando* and *Bailando*++.** Here we present two examples where the avatar (a) kneels and (b) turn around. Errors that disobey the human morphology occur in avatar animation of the original *Bailando* (blue and red box). The inverse kinematics that transfer the 3D positions to 3D joint rotations of the original *Bailando* is conducted under the computer graphics software *Unity*.

approaches both quantitatively and qualitatively. In comparison to the original Bailando [13], this version adds the hybrid training strategy and contextual music encoding for improved performance and practical use. Ablation studies are added to demonstrate the effectiveness of the new components. The supplementary video can refer to https://youtu.be/jht6NpwqLM4.

2 Related Work

2.1 Graph-based Motion Synthesis and Music to Dance

Producing realistic human motions has been long studied. A typical class of approaches is *graph-based* methods. They are developed on the idea of “cropping and pasting”, which cut motion clips from existing data as individual nodes and splice these nodes to synthesize new motions according to proper rules [16], [17], [18], [19]. For music to dance, further constraints on the music rhythms, including source-target music similarity [20], beat-wise motion connectivity [21], and deep rhythm signatures [9], are introduced into the linking rules of the graph-based methods to align the motion with music beats. However, since the tempos, length, and speed of the cropped dance units are fixed, the graph-based methods would encounter temporal conflicts on diverse rhythms. For example, the dance units cropped in music of 4/4 time signature cannot synthesize movement for 3/4, while the motion tempos of 60 beats per minute (BPM) is not adaptable for 80 BPM. As a result, this kind of works can perform well in restricted rhythm ranges but is not compatible with various genres of music beats in wild scenarios.

2.2 Learning-based Dance Generation

In recent years, with the emergence of deep learning, many works design a dedicated network structure, including CNNs [22], RNNs [2], [4], [23], [24], GCNs [25], [26], [27], GANs [28], [29] and Transformers [7], [30], [31], to map the given music to a joint sequence of the continuous human pose space directly. Specifically, Holden *et al.* [22] exploit an convolutional-network-(CNN)-based autoencoder to learn a deep representation of human motion and learn an additional stacked network to map control signals to the learned representation for motion synthesis. Alemi *et al.* [2] apply Factored Conditional Restricted Boltzmann Machine (FCRBM) to recurrently predict next motion frame with the concatenation of history motion and future music as input, while Tang *et al.* train a Long Short-Term Memory (LSTM) [32]-based autoencoder to extract music features and feed the encoded features to another LSTM to generate motions. Yan and Li *et al.* [25] propose to generate dance sequences via skeleton-based graph convolutions and graph upsamplings from sampled latent vectors in Gaussian process. Ren *et al.* [26] use a spatio-temporal graph convolutional network to map the music features encoded by GRU to skeleton sequences, supervised with a generative adversarial loss [33], while Sun *et al.* [29] also conduct an adversarial loss to train a mapping network composed of mixed CNNs and LSTMs. Recently, along with the development of Transformer [34], many works try to integrate the attention-based structure into music-to-dance mapping networks. Huang *et al.* [24] apply a Transformer-based music encoder to get long-range audio features that are fed to a subsequent LSTM for iterative pose generation. Li and Yang *et al.* [7] employ two Transformers to separately encode the music features and the historic motion features and autoregressively predict the future pose via a third cross-modal Transfomer. Li *et al.* [31] design a Transformer with attention layers specific for the hierarchical connections of skeleton and apply the Transformers to separately predict the key poses in 3D joint rotation format and connecting curves among key poses. Although these methods exploits diverse network structures, they share the same core routine that directly mapping the music features to 3D joint pose sequence in a discriminative manner. Due to lacking explicit restrictions to keep the generated pose within the spatial constraint, such methods would regress to nonstandard poses that are beyond the dancing *subspace* during inference, resulting in instability in real uses. In recent studies, there has been exploration into generating 3D motions using denoising diffusion probabilistic models (DDPM) [35], [36], [37] One notable work in this area is EDGE [37], which also uses a Transformer as the decoder backbone. Different from other Transformer-based methods, EDGE generates dance motions by iteratively reconstructing pose pieces of fixed length from Gaussian noises. These generated motion pieces are then stitched together to form long sequences, with the constraint that the first half of each piece aligns with the second half of the

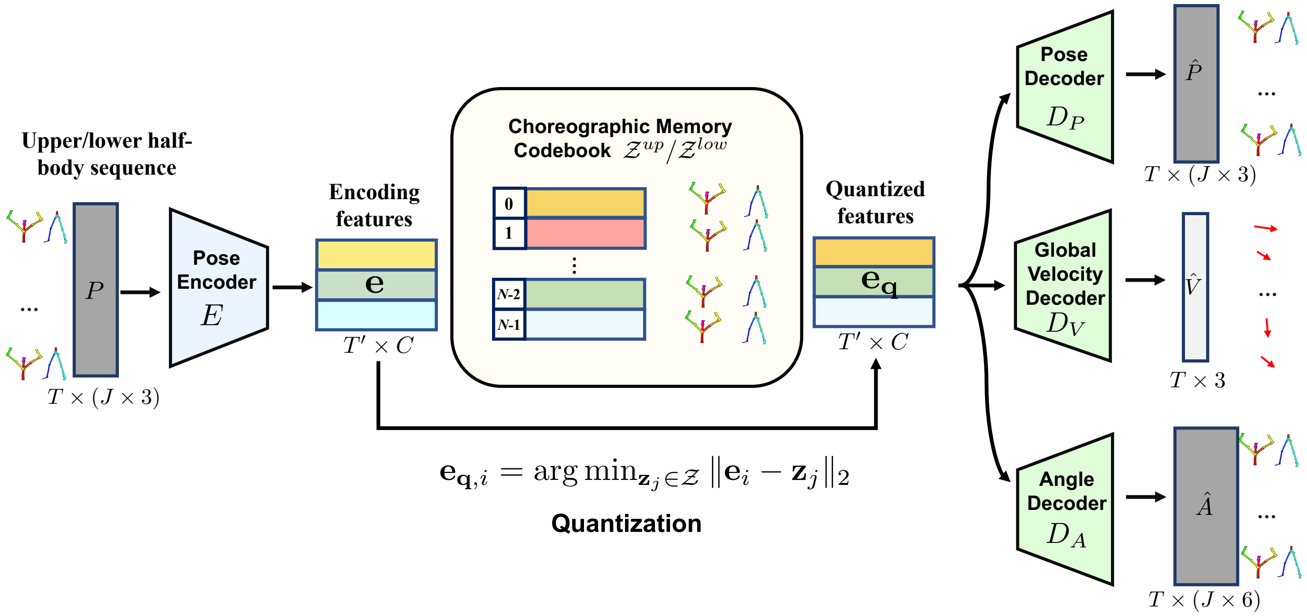


Fig. 4: **Structure of 3D Pose VQ-VAE.** The proposed 3D pose VQ-VAE is learned to encode and summarize meaningful dancing units to choreographic memory, and to reconstruct the target pose sequence from quantized features. The parameters of encoder and decoders and the codebook are jointly learned during training.

previous piece.

Besides various kinds of methods, different 3D dancing sequence data are made from mocap and reconstruction [2], [4], [38]. Recently, a large-scale 3D dancing dataset AIST++ [7] is built from multi-camera videos along with the music in different styles and speeds, facilitating both training and testing of this task.

2.3 Two-stage Generation using VQ-VAE and GPT

The two-stage approaches, which first encode data using VQ-VAE and afterwards learn a probabilistic model (GPT) to generate the encoding from quantized codebook, have been applied in multiple generative areas [39], [40], [41]. For example, Dhariwal *et al.* [39] extracts audio features and generate songs according to the lyrics, while most recently Esser *et al.* [41] encode perceptually rich image constituents to quantized patches and tames the Transformer to generate contextually plausible images in large resolutions. In our work, we encode and quantize meaningful dancing constituents into a choreographic memory and generate visually satisfactory dance by jointly translating the music and existing movements to targeted future poses.

3 Our Approach

The overview of our dance generation framework, *Bailando*++, is shown in Figure 2. Unlike other learning-based methods, we do not learn a direct mapping from audio features to the continuous domain of 3D joint sequence. Instead, we first encode and quantize the spatially standard dance movements into a finite codebook $\mathcal{Z} = \{\mathbf{z}_i\}_{i=0}^{N-1}$ as choreographic memory in Section 3.1, where N is the codebook length and every code $\mathbf{z}_i$ is shown to represent a dancing-like pose with contextual semantic information. Specifically, we learn VQ-VAEs on the upper and lower half bodies separately and represent the dance movement into a sequence of compositional upper-and-lower pose code pairs $\boldsymbol{p} = [p^u, p^l]$. In light of the benefits of dance generation in the 3D rotation domain, we further apply the pose VQ-VAE to synthesize motion in SMPL [15] format. However, poses expressed in the 3D rotation format do not have as sufficient distinguishability as those in spatial distribution, and hence it is not feasible to summarize spatially representative movements from unsupervised training of VQ-VAE on rotation data. Therefore, we propose a hybrid training strategy to make 3D joint positions a strong spatial prior of motion synthesis on rotation angles in Section 3.2. With the learned VQ-VAE, we train networks to choreograph from input music signals to summarized quantize codes in choreographic memory. First, we apply an attention-based contextual music encoder to augment the contextual input audio features within a sliding window in Section 3.4. Then, we introduce a motion GPT to translate the music feature and source pose codes to the future pose codes in Section 3.3. Furthermore, to achieve synchronized alignment between generated motion tempos and music beats, we propose actor critic learning on the motion GPT with our newly designed beat-align rewards in Section 3.5. The generated pose code sequences are finally decoded to fluent 3D dance by VQ-VAE decoders.

3.1 3D Pose VQ-VAE with Choreographic Memory

Dance positions, *i.e.*, the meaningful poses in dancing movements, are the basic constituents of a piece of dance. The process of choreography can be regarded as the combinations and connections of dance positions. Although dances vary greatly in style or speed, they share common dance positions. Instead of indicating fixed units of dance with plenty of manual effort, our goal is to summarize such dance positions into a rich and reusable codebook in an unsupervised manner, such that any piece of dance $P \in \mathbb{R}^{T\times(J\times3)}$, where T is the time length and J is the number of human joints, can be represented by a sequence of codebook elements $\mathbf{e_q} \in \mathbb{R}^{T'\times C}$, where $T' = T/d$, d is the temporal down-sampling rate, and C is the channel dimension of features.

To collect distinctive pose codes and reconstruct them back to represented dancing sequence efficiently, we design a 3D pose

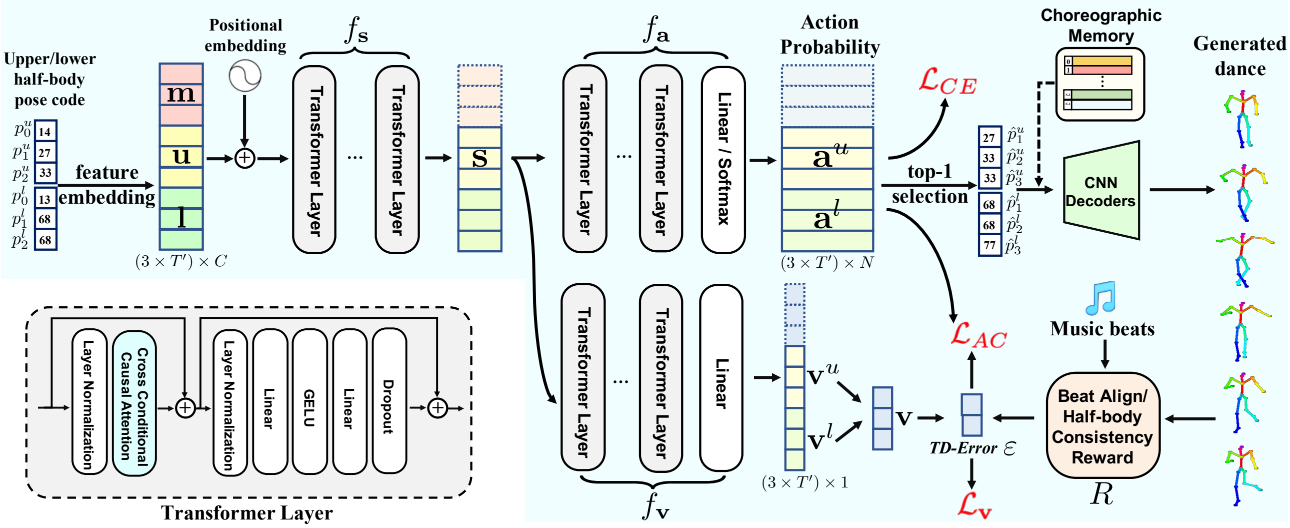


Fig. 5: **Actor-Critic GPT.** The GPT is learned to sequentially translate the source pose codes $[p_t^u, p_t^l]$ of upper-and-lower half bodies along with music features $\mathbf{m}$ to the targeted future pose codes $[\hat{p}_{t+1}^u, \hat{p}_{t+1}^l]$. The parameters of the networks are learned via cross-entropy loss $\mathcal{L}_{CE}$ with ground truth and actor-critic loss $\mathcal{L}_{AC}$.

VQ-VAE as shown in Figure 4. In this scheme, we first adopt a 1D temporal convolution network E to encode the 3D joint sequence P to context-aware features $\mathbf{e} \in \mathbb{R}^{T' \times C}$.

Then, we quantize $\mathbf{e}$ by substituting each temporal feature $\mathbf{e}_i$ to its closest codebook element $\mathbf{z}_j$ as

$$\mathbf{e}_{\mathbf{q},i} = \arg\min_{\mathbf{z}_j \in \mathcal{Z}} \|\mathbf{e}_i - \mathbf{z}_j\|. \quad (1)$$

Finally, we decode the quantized features $\mathbf{e_q}$ via a CNN D_P and reconstruct the dance movement $\hat{P}$.

Compositional Human Pose Representation. In order to represent a larger range of motions by training on limited dance data, we train independent 3D pose VQ-VAEs and learn two separate codebooks $\mathcal{Z}^u$ and $\mathcal{Z}^l$ for the upper and lower half bodies, respectively, such that we can combine different upper-lower code pairs to enlarge the range of dance positions that the learned codebooks can cover. Meanwhile, to avoid encoding confusion caused by a global shift of joints (*e.g.*, the same motion may be encoded to different features when it is at different locations), we normalize the absolute locations of input P, *i.e.*, setting the root joints (hips) to be 0. To realize the overall movement, we add a separate decoder branch D_V, which predicts the global movement velocity $\hat{V} \in \mathbb{R}^{T \times 3}$ according to pose codes of the lower half body, where $\hat{V}_t$ represents the shift of root joint between the $(t+1)$-th and the t-th frames.

Learning Stable 3D Pose VQ-VAEs. The pose encoder E and decoder D_P are simultaneously learned with the codebook via the following loss function:

$$\mathcal{L}_{VQ} = \mathcal{L}_{rec}(\hat{P}, P) + \|\text{sg}[\mathbf{e}] - \mathbf{e_q}\| + \beta\|\mathbf{e} - \text{sg}[\mathbf{e_q}]\|. \quad (2)$$

The global velocity decoder branch is learned thereafter by fixing the parameters of other parts of VQ-VAE via loss function $\mathcal{L}_{rec}(\hat{V}, V)$, where V is the ground truth global velocity. $\mathcal{L}_{rec}$ is the reconstruction loss that constrains the predicted 3D joint sequence to ground truth. In this loss, we regress not only the original 3D points of joints, but also the velocities and accelerations of movements:

$$\mathcal{L}_{rec}(\hat{P}, P) = \|\hat{P} - P\|_1 + \alpha_1\|\hat{P}' - P'\|_1 + \alpha_2\|\hat{P}'' - P''\|_1, \quad (3)$$

where P' and P'' represent the 1^{st}-order (velocity) and 2^{nd}-order (acceleration) partial derivatives of 3D joint sequence P on time, while α_1 and α_2 are trade-off weights. Experimental results show that "velocity-and-acceleration" loss items play essential roles in preventing jitters in generated dance. (See Section 4.2.) The second part of $\mathcal{L}_{VQ}$ is the "codebook loss" to learn codebook entries, where sg$[\cdot]$ denotes "stop gradient" [42], while the third part is the "commitment loss" with trade off β [39], [41]. Since the quantization operation of Equation 2 is not differentiable, to train the whole networks end to end, the back-propagation of this operation is achieved by simply passing the gradient of $\mathbf{e_q}$ to $\mathbf{e}$. The training process with $\mathcal{L}_{VQ}$ achieves two learning objectives. First, it involves learning an encoder-decoder pair that can effectively represent the motion sequence to temporal deep features and reconstruct it from the feature space, using the unsupervised loss $\mathcal{L}_{rec}$, akin to a typical autoencoder. Simultaneously, a group of principle motion components in deep feature space are summarized to codebook $\mathcal{Z} = \{\mathbf{z}_i\}$, *i.e.* the choreographic memory, during the training process. This is achieved through the combined effect of the "codebook loss" and the "commit loss", which encourage the encoding feature $\mathbf{e}$ to converge towards its most similar codebook element $\mathbf{z}_i$, and vice versa. Over repeated iterations of the training data, similar encoding features are clustered together to form typical and reusable components that effectively represent the high-quality dance motion domain.

The learned choreographic memory codes are interpretable. After the training process of pose VQ-VAEs, each quantized feature in the codebook is decoded into a unique dance position. And any permutation and combination of codes can be decoded into a piece of fluent movement based on corresponding dance positions. (See Section 4.3.)

3.2 A Hybrid Training Strategy for 3D Rotation

Generating dance in 3D joint rotation format rather than 3D positions has several advantages, such as the ability to drive avatar animation without additional inverse kinetics operations and the avoidance of mapping errors like morphological distortions

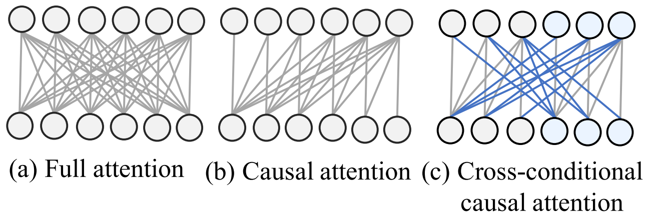


Fig. 6: **Different types of attention layers.** The proposed cross-conditional causal attention realizes causal inferences intra (gray lines) and inter (blue lines) different kinds of components (gray and blue circles). Two kinds of components are shown here for concision, but three (music, upper, lower bodies) are in reality.

(see Figure 3). However, as we will demonstrate below, it is not feasible to directly train VQ-VAE on rotation data using the method described in Section 3.1.

If we look closely at the quantization step (described in Equation 2) of the pose VQ-VAE, we can see that an input motion feature, $\mathbf{e}$, is transformed into a template feature, $\mathbf{z}$, with the smallest distance to $\mathbf{e}$ in the codebook. In other words, motion features that are similar tend to be clustered into the same pose code in VQ-VAE. Therefore, when training on 3D joint position data, spatially similar dancing components will be clustered together, while those that are visually distinct will be summarized into different pose codes. As a result, even though the learning process is unsupervised, VQ-VAE is still able to construct spatially diverse and standard dancing units from 3D position data, which contributes to the high performance of Bailando.

The distance between poses, however, is not the same in 3D physical space when expressed in 3D rotation format. Poses that appear to change significantly in 3D position space may have a small distance in the 3D rotation format. One reason for this is that the rotation format, such as SMPL, uses a hierarchical structure of joint angles, meaning that the rotation angle of each joint only represents the local rotation of its inherited bones relative to that joint. As a result, poses that only change a few joints (such as bending over or spinning in place) have only a small portion that is different, and tend to be clustered into the same pose code when training the VQ-VAE, even though they can be significantly different in 3D space. This leads to these spatially distinct dancing components being summarized into a single dancing unit, reducing the diversity of motions that the choreographic memory can represent. Meanwhile, according to Equation 7, the pose codes will regress to an average of the clustered movements. Since VQ-VAE may summarize distinct poses into a single cluster, the averaged motion manifold is not guaranteed to represent a spatially standard dancing movement. These factors lead to poor performance of the pose VQ-VAE when trained solely on rotation data, and therefore result in low-quality choreographic results, as shown in Section 4.2.

Hybrid Training Strategy. To address the issue of inconsistent distances in the rotation data, we use 3D joint positions as a prior that ensures the spatial diversity and quality of the learned dancing units in the choreographic memory. Specifically, we train the full pose VQ-VAE using the following steps: (1) we train the pose VQ-VAE, including the pose encoder E, the codebook $\mathcal{Z}$, and the pose decoder D, using 3D joint position data P. (2) We freeze the encoder and codebook and train the global velocity decoder D_V to predict the global shift V. (3) We freeze E and $\mathcal{Z}$, and train the rotation angle decoder D_A by reconstructing the 3D rotation angles A from the 3D positions P. We use the rotation matrix of SMPL joint angles, as it has been shown to be more stable in Li *et al.* [7]. Mapping pose codes of 3D joint positions is beneficial for two reasons: 1) a pose code represents a 3D dancing pose, and 2) a 3D joint position motion sequence P corresponds to that in joint angles A. Under this training strategy, the learned VQ-VAE maintains spatial quality while synthesizing dance movements in rotation format (see Section 4).

Learning a Stable Joint Angle Decoder. When training the joint angle decoder D_A, we use the following loss function:

$$\mathcal{L}_A = \|\hat{A} - A\|_1 + \alpha_1\|\hat{A}' - A'\|_1 + \alpha_2\|\hat{A}'' - A''\|_1, \qquad (4)$$

where $\hat{A}$ is the reconstructed 3D joint rotation angles and A is the ground truth. In our experiments, we observe that the constraints of the first-order and second-order derivatives (*i.e.*, A' and A'') are still effective to suppress the motion jitters when applied through training on 3D rotation data (see Section 4.2).

3.3 Cross-Conditional Motion GPT

Now that we can represent any piece of dance by a sequence of quantized position codes, the dance generation task is then reframed to select proper codes from codebook $\mathcal{Z}$ for future actions according to given music and existing movements. For any target time t, we estimate the probability of every $\mathbf{z}_i \in \mathcal{Z}$ and select the one with the largest possibility as the predicted pose code $\hat{\boldsymbol{p}}_t$. Since we model the upper and lower half bodies separately, in order to keep the coherence of the composed body and to avoid the asynchronous situation (*e.g.*, the direction of the upper half is opposite to that of the lower), the prediction of the future action should be cross-conditioned between existing upper and lower movements to make the most of mutual information:

$$\begin{cases} \hat{p}^u_t &= \arg\max_k \mathbb{P}(\mathbf{z}^u_k | \mathbf{m}_{1\ldots t}, p^u_{0\ldots t-1}, p^l_{0\ldots t-1}) \\ \hat{p}^l_t &= \arg\max_k \mathbb{P}(\mathbf{z}^l_k | \mathbf{m}_{1\ldots t}, p^u_{0\ldots t-1}, p^l_{0\ldots t-1}) \end{cases} \qquad (5)$$

We introduce the powerful GPT model [11] to estimate the action probabilities as shown in Figure 5. Given a dance position code sequence with length of T', we first embed the upper and lower pose codes to learnable features $\mathbf{u} \in \mathbb{R}^{T' \times C}$ and $\mathbf{l} \in \mathbb{R}^{T' \times C}$, respectively, and concatenate them with music features $\mathbf{m}$ on the temporal dimension. Then, we add a learned positional embedding to this concatenated $(3 \times T') \times C$ tensor and feed it to 12 successive Transformer layers, the structure of which is shown in Figure 5. At last, we employ a linear transform and softmax layer to map the output of Transformer layers to normalized action probability $\mathbf{a} \in \mathbb{R}^{(3 \times T') \times N}$, where N is the size of learned codebook and $\mathbf{a}_{t,i}$ reveals the probability of pose code $\mathbf{z}_i \in \mathcal{Z}$ predicted for time $t+1$. The action probabilities for upper and lower half bodies are indexed as $\mathbf{a}^u_{0:T'-1} = \mathbf{a}_{T':2T'-1}$ and $\mathbf{a}^l_{0:T'-1} = \mathbf{a}_{2T':3T'-1}$, respectively.

In Transformers [43], the attention layer is the core component that determines the computational dependency among sequential elements of data, and is implemented as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{M}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T + \mathbf{M}}{\sqrt{C}}\right)\mathbf{V}, \qquad (6)$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ denote the query, key and value from input, and $\mathbf{M}$ is the mask, which determines the type of attention layers. The most two common types of attention are "full attention" [43] and "causal attention" [43], where the former realizes the intercommunication of input data at all times while the latter only allows the current

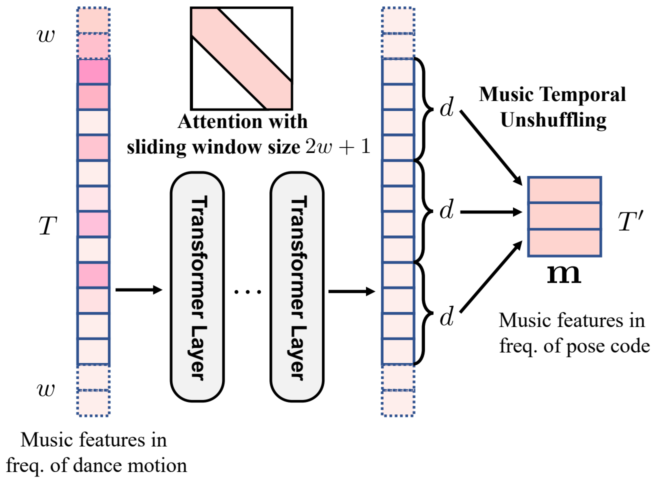

w
T
w
Music features in freq. of dance motion
Attention with sliding window size $2w+1$
Transformer Layer
...
Transformer Layer
d
d
d
Music Temporal Unshuffling
T'
$\mathbf{m}$
Music features in freq. of pose code

Fig. 7: **Contextual Music Encoding.** In this encoding pipeline, music features in 60 fps with a length of $T+2w$ are sampled and fed into a cascade of Transformer layers. The attention of the Transformer layer here aggregates adjacent music features within a $(2w+1)$-wide sliding window. The augmented features are finally downsampled by an unshuffling operation [44] across the temporal dimension.

and previous data to compute the state for the time of interest. As our goal is to infer the future dance position codes, we adopt the causal attention. However, since the generation of upper and lower half bodies are dependent on each other, we cannot realize the inference by just reordering the sequence of input to fit the causality as previous works [39], [41]. Therefore, we propose an attention layer, namely cross-conditional attention, to comply with the causality cross conditioned among features of the music, the upper half and the lower half bodies, where $\mathbf{M}$ is designed to be a 3×3 repeated block matrix with a lower triangular matrix of size T' as its element. As shown in Figure 6, the proposed attention can exchange information of different components, and guarantee that the future information will not be transmitted back to the past.

Learning Motion GPT. The motion GPT is optimized via supervised training with the cross-entropy loss on action probability $\mathbf{a}$:

$$\mathcal{L}_{CE} = \frac{1}{T'}\sum_{t=0}^{T'-1}\sum_{h=u,l}\text{CrossEntropy}\left(\mathbf{a}_t^h, p_{t+1}^h\right). \quad (7)$$

Given a sequence of pose codes $\boldsymbol{p}_{0:T'-1}$ and relevant music features $\mathbf{m}_{1:T'}$ as input, the learned GPT outputs the sequence of actions $\mathbf{a}_{0:T'-1}$ all at once to predict $\boldsymbol{p}_{1:T'}$. This parallel characteristic makes Transformer an ideal model for reinforcement learning [45], [46]. In the following subsection, we adopt the learned motion GPT as a pretrained policy maker and propose a novel actor-critic based finetuning scheme to further improve its performance as complementary to the supervised training above.

3.4 Contextual Music Encoding

Before choreographing based on the quantized dancing units, we need to extract informative features from the input music signals to serve as the condition for the proposed motion GPT. One approach is to sample audio features from the music signals at the same frequency as the pose codes, as in the original Bailando [13]. However, since the conditional motion prediction in Equation 5 only looks ahead one music feature at a time, this sampling method can make the GPT short-sighted when it comes to future music and prone to generating irregularities in the dance movement in the long term, which does not align with the logic of human choreography.

To enhance the motion GPT's understanding of long-term melodies, we use contextual music encoding (CME) to augment music features with context, as shown in Figure 7. Specifically, we first sample audio features at the same frequency as the motion sequence data (60 fps) and obtain features of the same length as the motions. Then, for a piece of music with a length of T, we extend its two ends by a length w from the original music. If we reach the beginning or end of the music, we pad it with zeros. Next, we feed the extended music features to a series of sequential Transformer layers to conduct attention-based context augmentation, where the Transformer layer is the same as the one detailed in Figure 5, except for the attention layer. Since the purpose of CME is to exchange information among contextual melodies rather than to infer the future, the attention here is conducted among temporally adjacent music features within a window. Specifically, the mask $\mathbf{M}$ in CME is a banded matrix with a sliding window size of $2w+1$. Finally, we perform temporal-wise unshuffling [44] on the music features augmented by the Transformer with a downsampling rate of d to obtain the final music features of length T', which is at the same frequency as the pose code.

Our experiments show that the use of Contextual Music Encoding (CME) has allowed the motion GPT to better predict motion that matches the long-term patterns in music. For example, when given a piece of music with a repeated melody, CME helps the motion GPT to generate repetitive motions. In addition, CME also helps to smooth out irregular changes in dance movements and styles within a single dance, resulting in movement that is closer to ground truth in terms of speed and kinetic characteristics. An ablation study is provided in Section 4.2.

3.5 Actor-Critic Learning

While the supervised learning scheme for the motion GPT is straightforward and easy to train, it is intractable to involve further a more flexible constraint of generated dance (*e.g.*, a regularization item that strengthens the consistency of dance beats) to Equation (7), since the supervision target is the code number, which is not differentiable to compute the quantitative constraints on the final dance sequence.

To address this issue and to achieve more accurate synchronized alignment between diverse motion tempos and music beats, we apply actor-critic learning to the motion GPT with a newly-designed reward function. In particular, we regard the first 6 Transformer layers of motion GPT as "state network" $f_{\mathbf{s}}$, and the outputs of $f_{\mathbf{s}}$ are states $\mathbf{s}$ for time 0 to $T'-1$, while the latter 6 Transformer layers along with the linear-softmax layer are regarded as "policy making network" $f_{\mathbf{a}}$, where the actions are computed according to state as $\mathbf{a} = f_{\mathbf{s}}(\mathbf{s})$. Besides, we add a separate three-layer Transformer branch as "critic value network" $f_{\mathbf{v}}$ to estimate the critic values $\mathbf{v}_{0:T'-1} \in \mathbb{R}^{T'\times 1}$ as

$$\mathbf{v} = \mathbf{v}^u + \mathbf{v}^l = f_{\mathbf{v}}(\mathbf{s})_{T':2T'-1} + f_{\mathbf{v}}(\mathbf{s})_{2T':3T'-1}. \quad (8)$$

With the well-defined reward function $R(t) = R(\mathbf{a}_t, \mathbf{s}_t)$, the objective of reinforcement learning is to maximize the expected accumulated rewards:

$$J = \mathbb{E}_\tau\left[\sum_{t=0}^{T'-1} R(t)\right], \quad (9)$$

where $\tau = \{\mathbf{a}_t\}_{t=0}^{T'-1}$ is the trajectories of actions predicted by the policy making network. The learning objective can be reframed as

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[R(\tau)] = \int \pi_\theta(\tau)R(\tau)d\tau, \tag{10}$$

where θ denotes the weights of the policy making network, τ represents a string of actions and $\pi_\theta(\tau)$ is the probability that the policy network predicts to take such actions.

One approach to maximize J is to optimize the network weight θ along the gradient $\nabla_\theta J$ as $\theta \leftarrow \theta + \alpha\nabla_\theta J$. Since

$$\nabla_\theta \pi_\theta = \pi_\theta \frac{\nabla_\theta \pi_\theta}{\pi_\theta} = \pi_\theta \nabla_\theta \log \pi_\theta, \tag{11}$$

we can rewrite $\nabla_\theta J$ into

$$\begin{aligned} \nabla_\theta J &= \int \nabla_\theta \pi_\theta(\tau)R(\tau)d\tau \\ &= \int \pi_\theta \nabla_\theta(\tau)\log \pi_\theta(\tau)R(\tau)d\tau \\ &= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau)R(\tau)\right]. \end{aligned} \tag{12}$$

Note that $\tau = \{\mathbf{a}_t\}_{t=0}^{T'-1}$ is the trajectories of actions predicted on states $\{\mathbf{s}_t\}_{t=0}^{T'-1}$, the probability of trajectory $\pi_\theta(\tau)$ predicted by the policy making network is expanded to be $\prod_{t=0}^{T'-1}\pi_\theta(\mathbf{a}_t, \mathbf{s}_t)$, where $\pi_\theta(\mathbf{a}_t, \mathbf{s}_t)$ is the probability of action $\mathbf{a}_t$ under state $\mathbf{s}_t$. Hence,

$$\nabla_\theta \log \pi_\theta(\tau) = \sum_{t=0}^{T'-1} \nabla_\theta \log \pi_\theta(\mathbf{a}_t, \mathbf{s}_t) \tag{13}$$

and we have

$$\begin{aligned} \nabla_\theta J &= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau)R(\tau)\right] \\ &= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\left(\sum_{t=0}^{T'-1}\nabla_\theta \log \pi_\theta(\mathbf{a}_t, \mathbf{s}_t)\right)\left(\sum_{t=0}^{T'-1} R(\mathbf{a}_t, \mathbf{s}_t)\right)\right] \\ &= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\sum_{t=0}^{T'-1}\nabla_\theta \log \pi_\theta(\mathbf{a}_t, \mathbf{s}_t)\left(\sum_{t'=0}^{T'-1} R(\mathbf{a}_{t'}, \mathbf{s}_{t'})\right)\right]. \end{aligned} \tag{14}$$

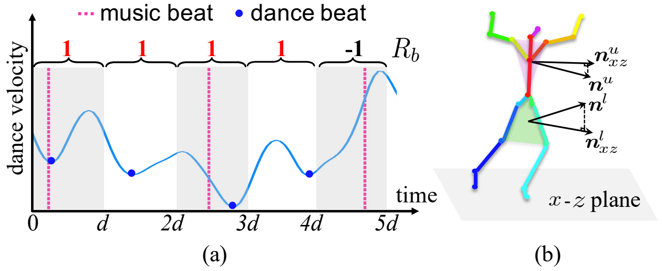
For on-policy reinforcement learning, $\nabla_\theta J$ is estimated on simultaneously sampled sectional trajectories $\{(\mathbf{a}_t^m, \mathbf{s}_t^m)\}_{m=0}^{M-1}$, where M denotes the sampling batch size, such that the equation above is approximated to be

$$\nabla_\theta J \approx \frac{1}{M}\sum_{m=0}^{M-1}\sum_{t=0}^{T'-1}\nabla_\theta \log \pi_\theta(\mathbf{a}_t^m, \mathbf{s}_t^m)\left(\sum_{t'=0}^{T'-1} R(\mathbf{a}_{t'}^m, \mathbf{s}_{t'}^m)\right). \tag{15}$$

Note that the optimization of the policy making network for policy $(\mathbf{a}_t^m, \mathbf{s}_t^m)$ is not expected to be influenced by the past trajectories, *i.e.*, the rewards before t. Therefore, Equation (15) is reframed as

$$\nabla_\theta J \approx \frac{1}{M}\sum_{m=0}^{M-1}\sum_{t=0}^{T'-1}\nabla_\theta \log \pi_\theta(\mathbf{a}_t^m, \mathbf{s}_t^m)\left(\sum_{t'=t}^{T'-1} R(\mathbf{a}_{t'}^m, \mathbf{s}_{t'}^m)\right), \tag{16}$$

where $\sum_{t'=t}^{T'-1} R(\mathbf{a}_{t'}, \mathbf{s}_{t'})$ is the expected "reward to go" under policy $(\mathbf{a}_t, \mathbf{s}_t)$, which is formally named as the Q-value $Q(\mathbf{a}_t, \mathbf{s}_t)$.



(a) (b)

Fig. 8: **Designed rewards.** (a) Beat-align reward penalizes the absence of dance beat for the interval that has music beat. (b) Half-body consistency reward is computed on the angle between normal directions of half bodies to prevent asynchronizations.

To avoid the bias of rewards, *e.g.*, all rewards are positive, the Q-value item in Equation (16) is normalized by an expected "reward to go" on state $\mathbf{s}_t$, *i.e.*, the critic value $\mathbf{v}_t = \mathbb{E}_{\mathbf{a}_t}[Q(\mathbf{a}_t, \mathbf{s}_t)]$, such that

$$\begin{aligned} \nabla_\theta J &\approx \frac{1}{M}\sum_{m=0}^{M-1}\sum_{t=0}^{T'-1}\nabla_\theta \log \pi_\theta(\mathbf{a}_t^m, \mathbf{s}_t^m)\left(Q(\mathbf{a}_t, \mathbf{s}_t) - \mathbf{v}_t\right) \\ &= \frac{1}{M}\sum_{m=0}^{M-1}\sum_{t=0}^{T'-1}\nabla_\theta \log \pi_\theta(\mathbf{a}_t^m, \mathbf{s}_t^m)\left(R(\mathbf{a}_t, \mathbf{s}_t) + \mathbf{v}_{t+1} - \mathbf{v}_t\right). \end{aligned} \tag{17}$$

Examining Equation (17), if the normalized "reward to go" is positive, to increase J, the policy making network will be optimized to enhance the probability of action $\mathbf{a}_t^m$ under state $\mathbf{s}_t^m$. In our proposed framework, the actions are within finite selections of the choreographic memory. Hence, the optimization along with Equation (17) is equivalent to the optimization via a weighted cross-entropy loss on the in-time self-predictions of the policy making network:

$$\begin{aligned} &\mathcal{L}_{AC} = \\ &\frac{1}{T'-1}\sum_{t=0}^{T'-2}\left(\sum_{h=u,l}\text{CrossEntropy}\left(\mathbf{a}_t^h, \hat{p}_{t+1}^h\right)\right)\cdot \text{sg}[\varepsilon_t], \end{aligned} \tag{18}$$

where $\hat{p}_{t+1}^h = \arg\max_i \mathbf{a}_{t,i}^h$ is the pose code number predicted by the policy making network. $\boldsymbol{\varepsilon} \in \mathbb{R}^{(T'-1)\times 1}$ denotes the so-called TD-error calculated as

$$\boldsymbol{\varepsilon}_{0:T'-2} = \mathbf{r}_{0:T'-2} + \text{sg}[\mathbf{v}_{1:T'-1}] - \mathbf{v}_{0:T'-2}, \tag{19}$$

where $\mathbf{r}_t = R(t)$. Meanwhile, the critic value network is optimized by bootstrap training on the difference between $\mathbf{v}_{0:T'-2}$ and $R(\mathbf{a}_t, \mathbf{s}_t) + \mathbf{v}_{1:T'-1}$:

$$\mathcal{L}_{\mathbf{v}} = \frac{1}{T'-1}\|\boldsymbol{\varepsilon}\|_2^2. \tag{20}$$

The computation of actor-critic loss $\mathcal{L}_{AC}$ depends on real-time actions predicted by the motion GPT and the optimization direction is determined on the value of TD-error. When ε_t is positive, the optimization on $\mathcal{L}_{AC}$ will intensify the convergence to predicted code $\hat{p}_{t+1}$, while in the opposite situations, the probability estimated for $\hat{p}_{t+1}$ will be reduced.

The value of TD-error and the learning effectiveness are strongly influenced by the reward function R. In this work, we design a motion-music beat-align reward to generate dance that aligns more accurately with the rhythm of the music. As shown in

TABLE 1: **Quantitative results on AIST++ test set.** The best and runner-up values are bold and underlined, respectively. Among compared methods, "Li *et al.*", DanceNet and FACT are multiplexing the same results of AIST++ benchmark [7], while DanceRevolution [24] is reproduced using officially released code with the optimal settings. † FID$_k$ and DIV$_k$ are fetched from [7] while FID$_g$ and DIV$_g$ are recomputed using the officially updated evaluation code. *The generated dances of "Li *et al.*" are highly jittery making its velocity variation extremely high, which is also reported in [7].

	Motion Quality		Motion Diversity			User Study
Method	FID$_k$ ↓	FID$_g^\dagger$ ↓	Div$_k$ ↑	Div$_g^\dagger$ ↑	Beat Align Score ↑	*Bailando*++ Wins
Ground Truth	17.10	10.60	8.19	7.45	0.2374	66.5%±25.2%
Li *et al.* [30]	86.43	43.46	6.85*	3.32	0.1607	93.0%±13.8%
DanceNet [38]	69.18	25.49	2.86	2.85	0.1430	88.5%±13.5%
DanceRevolution [24]	73.42	25.92	3.52	4.87	0.1950	93.5%± 7.3%
FACT [7]	35.35	22.11	5.94	6.18	0.2209	90.5%±11.2%
Bailando	<u>28.16</u>	**9.62**	<u>7.83</u>	<u>6.34</u>	0.2332	78.0%±18.6%
***Bailando*++**	**17.59**	<u>10.10</u>	**8.64**	**6.50**	**0.2720**	–

Figure 8 (a), the beat-align reward is defined as

$$R_b(t) = \begin{cases} -1, & \exists \textit{ music beat } \wedge \nexists \textit{ dance beats} \in \hat{P}_{td:(t+1)d} \\ 1, & \textit{otherwise,} \end{cases} \tag{21}$$

where $\hat{P}_{0:T-1} = D(\hat{\boldsymbol{p}}_{0:T'-1})$ is the dance motion sequence decoded from predicted dance position codes. Meanwhile, to avoid the compositional asynchronization between upper and lower half bodies during actor-critic learning, we introduce a compositional consistency reward to impose penalties for situations where the upper and lower body are in the opposite direction:

$$R_c(t) = \inf\left\{\hat{R}_c(t)\right\}, t \in [dt, d(t+1)), \tag{22}$$

where

$$\hat{R}_c(t) = \begin{cases} \left\langle \boldsymbol{n}^u_{xz}(t), \boldsymbol{n}^l_{xz}(t) \right\rangle, & \left\langle \boldsymbol{n}^u_{xz}(t), \boldsymbol{n}^l_{xz}(t) \right\rangle < 0 \\ 1, & \textit{otherwise.} \end{cases} \tag{23}$$

Here, $\boldsymbol{n}^u_{xz}(t), \boldsymbol{n}^l_{xz}(t)$ are the normal directions of upper and lower bodies of $\hat{P}_t$ projected to the x-z plane, which is illustrated in Figure 8 (b). The final reward is then a weighted combination of R_b and R_c as $R = \gamma_b R_b + \gamma_c R_c$.

In the finetuning process, we fix the parameters of state network $f_{\mathbf{s}}$, and alternately train the policy making network $f_{\mathbf{a}}$ and the critic value network $f_{\mathbf{v}}$ using the losses introduced above with a small learning rate. After such finetuning, the proposed framework will be further enhanced.

4 Experiments

Dataset. We perform the training and evaluation on the AIST++ dataset proposed in [7], which to our best knowledge is the largest publicly available dataset for paired music and motions. This dataset contains 992 pieces of high-quality 60-FPS 3D pose sequence in SMPL format [15], where 952 are kept for training and 40 are used for evaluation.

Implementation Details. In this work, the choreographic memory codebook size N for both upper and lower bodies is set to 512, while the channel dimension C of encoded features is 512 and the temporal downsampling rate d of encoders is 8. It is worthy to note that $N = 512$ here is based on empirical observation. During the VQ-VAE training, to avoid "index collapse" [39], where some codes are never used in quantization, we track the usage of each code and randomly reset it if it is not used for a period of iterations. In our experiments, larger sizes (1024 and 768) cause frequent resets during training, with considerable fluctuations in the loss, and result in poorly restored motions. When N is 512, the loss decreases consistently, and the Euclidean restoration error of VQ-VAE is small on average (0.027, $\sim 5\%$ of the length from neck to hip) and not deviated ($\sigma = 0.0064$). Therefore, we regard $N = 512$ to be suitable and adequate for the training set of AIST++. While training the VQ-VAEs, dance data are cropped to a length of $T = 240$ (4 seconds) and sampled in a batch size of 32. The commit loss trade-off β in $\mathcal{L}_{VQ}$ is 0.1, while α_1 and α_2 in $\mathcal{L}_{rec}$ and $\mathcal{L}_A$ are set to be 1. Unless specifically mentioned, we use the rotation matrix to represent the 3D joint angles as in [7]. We adopt Adam optimizer [47] with $\beta_1 = 0.9$ and $\beta_2 = 0.99$ to train both pose VQ-VAEs for 500 epochs with learning rate 3×10^{-5}. As to the motion GPT, we follow the structure mirroring minGPT [48], where the channel dimension is 768, and the attention layer is implemented in 12 heads with dropout probability 0.1. The music features are extracted by the public audio processing toolbox Librosa [49], including *mel frequency cepstral coefficients (MFCC)*, *MFCC delta*, *constant-Q chromagram*, *onset strength*, and *tempogram* which are 55-dim in total, and are mapped to the same dimension of GPT by a linear layer after the contextual music encoder (CME). The CME has three Transformer layers with the structure as shown in Figure 5 in our experiments, and the window size parameter w is 44, corresponding to a temporal length of about 1.5 seconds. The block size T' of GPT is set to be 29. While training, the dance sequences are first encoded to pose codes $\boldsymbol{p}$ and sampled to the length of 30, where $\boldsymbol{p}_{0:28}$ are used as input and $\boldsymbol{p}_{1:29}$ are supervision labels. The motion GPT is optimized using Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.99$ for 400 epochs, where the learning rate is initialized as 3×10^{-4} and decayed after 200 epochs with factor 0.1. In the actor-critic finetuning process, we adopt a small learning rate of 1×10^{-5} to learn $f_{\mathbf{a}}$ and $f_{\mathbf{v}}$ for 10 epochs. The reward trade-offs γ_b and γ_c are 5 and 1, respectively. In our experiment, the pose VQ-VAEs and the motion GPT are trained sequentially, and the weights of VQ-VAEs are fixed during the learning process of GPT. The whole framework is learned in four days on one Tesla V100 GPU. During the test, the motion GPT takes a pair of starting pose codes, which can be either manually indicated or randomly sampled, as input and autoregressively generates the motion sequence as long as the target music.

Evaluation Metrics. For quantitative evaluations, we measure the

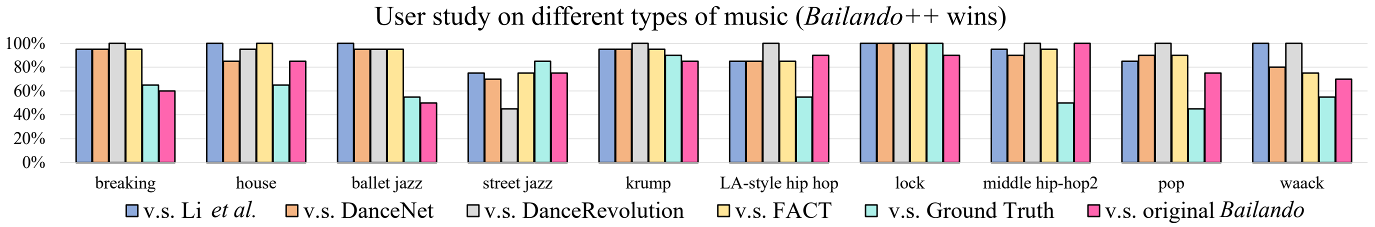


Fig. 9: **The distribution of *Bailando*++ wins on different music types compared to the state of the arts.** Each bar indicates the percentage that our method wins in comparison to the corresponding method.

TABLE 2: **Ablation study on pose VQ-VAE (3D joint position)**

Method	$\text{FID}_k \downarrow$	$\text{FID}_g \downarrow$
Ground Truth	17.10	10.60
w/o. upper/lower	41.21	15.85
w/o. global vel.	70.95	18.52
w/o. vel./acc. loss	30.91	11.87
full pose VQ-VAE (position)	28.23	12.63

generated dance from three perspectives: the quality of generated dances, the diversity of motions and the alignment between the rhythms of music and generated movements. In concrete, for the dance quality, we calculate the Fréchet Inception Distances (FID) [50] between the generated dance and all motion sequences (including training and test data) of the AIST++ dataset on kinetic features [51] (denoted as 'k') and geometric features [52] (denoted as 'g'), which are both extracted using the toolbox of [53]. As to the diversity, we compute the average feature distance of generated movements following [7]. Regarding the alignment between music and generated motions, we calculate the average temporal distance between each music beat and its closest dance beat as the Beat Align Score:

$$\frac{1}{|B^m|}\sum_{t^m \in B^m} \exp\left\{-\frac{\min_{t^d \in B^d} \|t^d - t^m\|^2}{2\sigma^2}\right\}, \tag{24}$$

where B^d and B^m record the time of beats in dance and music, respectively, while σ is normalized parameter which is set to be 3 in our experiment.

4.1 Comparison to Existing Methods

We compare our proposed model to several state-of-the-art methods including Li *et al.* [30], DanceNet [38], DanceRevolution [24] and FACT [7]. For each method, we generate 40 pieces of dances in AIST++ test set, and sample the generated dance sequence with a length of 20 seconds to compute the evaluation metrics mentioned above. We also calculate the quantitative scores for ground truth data in AIST++ test set and compare it to the generated dances.

The quantitative results are shown in Table 1. According to the comparison, our proposed model consistently performs favorably against all the other existing methods on all evaluations. Specifically, *Bailando* improves 7.19 (20%) and 12.49 (56%) than the best compared baseline model FACT on FID_k and FID_g, respectively, and even achieves a better FID_g score than the ground truth (9.62 v.s. 10.60). If we examine the metrics on these two kinds of features, the kinetic feature is defined on motion velocities and energies, which reflect the physical characteristics of dance, while the geometric feature is defined based on multiple man-made templates of movements, which reflects the quality of choreography. The superiority of our method on both dance quality metrics reveals that *Bailando* not only synthesizes more real-like motions than the compared baseline methods, but also achieves outstanding performance in organizing the movements to dance via the proposed actor-critic GPT scheme with learned choreographic memory. Meanwhile, *Bailando* can generate dance with high choreographic diversity instead of converging to few templates, and also achieves improvement on the correlation between music and motion.

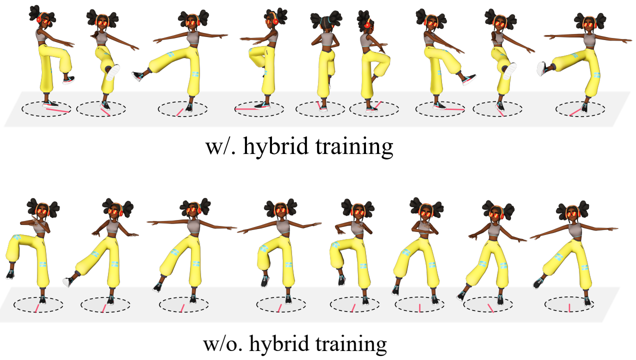
Based on the original *Bailando*, the extended *Bailando*++ can further improves the overall quality of generated dance Specifically, the FID_k score of *Bailando*++ is improved by 10.57 (38%) from *Bailando*, which indicates a significant boost in the quality of synthesized motion. Meanwhile, the two metrics on diversity increase 0.81 (9%) and 0.16 (3%) relatively. Although the FID_g score of *Bailando*++ is slightly worse (0.48, 5%) than that of the original Bailando, it is still better than the ground truth, which indicates that Bailando++ still performs a high quality of choreography. Besides, the generated dance movements of *Bailando*++ are more coherent to the beat of music melodies, with an improvement from 0.2332 to 0.2720 (14%). Visual comparisons can be found in the supplementary video.

User Study. To further understand the actual qualitative performance of our method, we conduct a user study on the dance sequences generated by each compared method (including the original *Bailando*) and the ground truth data in AIST++ test set. The experiment involves 20 participants, aged from 22 to 30 at the time of the study, while eight of them are female. For each participant, we randomly play 60 pairs of comparison videos with a length of around 10 seconds, where each pair contains our result and one competitor's in the same music, and ask the participant to indicate "*which one is dancing better to the music*". The statistics are shown in Table 1. Notably, our method significantly surpasses the compared state-of-art methods with at least 88.5% winning rate. Even in comparison to the ground truth, 66.5% of our generated dance is voted as the better in average. For the null hypothesis "*there is no difference between the results of Bailando++ and **", the p-value is smaller than 0.00001 as * represents every compared method and is 0.010038 as to ground truth, which suggests this hypothesis can be safely rejected for these comparisons.

The detailed distribution of the user study results on various music types is shown in Figure 9. The results of our user study indicate that our method, Bailando++, performs well across different types of dance, with a particularly strong performance in the lock dance category. The participants also noted that the generated dances were more "stable to the rhythm" and had "higher

TABLE 3: **Ablation study on downsampling rate (d) in pose VQ-VAE (3D joint position)**

downsample rate d	FID$_k$ ↓	FID$_g$ ↓
8	28.23	12.63
16	30.84	12.79
64	51.54	18.42



w/. hybrid training

w/o. hybrid training

Fig. 10: **Visualization on the effectiveness of hybrid training.** Here we show the dance generation result of "foettes on pointe" (spinning in place) with and without the hybrid training strategy when trained on the 3D joint rotation data. If learned without the hybrid training, the agent can only kick awkwardly and cannot turn around. The red lines indicate the orientations of the agent.

diversity" compared to the state of the art. In comparison to the original *Bailando*, *Bailando*++ was preferred by 78.0% of the participants, demonstrating the significant improvement in visual performance achieved through the use of contextual musical encoding and the hybrid training strategy.

4.2 Ablation Studies

We conduct ablation studies on the pose VQ-VAEs and the motion GPT, respectively. The quantitative scores are shown in Table 2, 4, 3, 5, and 6. The visual comparisons of this study can be found in the supplementary video.

Pose VQ-VAE. We separately study the pose VQ-VAEs for the two formats of synthesized motion. For pose VQ-VAE in 3D joint positions, we explore the effectiveness of the following components: (1) the up-lower half body separation, (2) the global velocity prediction branch, and (3) the velocity-and-acceleration loss used in $\mathcal{L}_{rec}$. We train three variant models without each of the three components, respectively. The motion quality measured for VQ-VAEs is on reconstructed results of ground truth of AIST++ test set. As shown in Table 2, the FID$_k$ and FID$_g$ values for variant "w/o. upper/lower" become worse by 12.98 (46%) and 3.22 (25%), respectively. The VQ-VAE trained on the whole body cannot reconstruct the dancing pose of the test set effectively. Therefore, the separate representations of upper-lower half bodies are necessary to enlarge the range of poses that the choreographic memory can cover. As to the global velocity branch, the motion quality scores of "w/o. global vel." sharply drops 42.72 (151%) and 5.89 (47%), respectively, which shows the isolated velocity prediction is critical for representing the dance movement. For "w/o. vel./acc. loss" variant, the FID$_k$ is worsened by 2.68. Although the FID$_g$ value of "w/o. vel./acc. loss" is slightly improved by 0.76, the model produces strong motion jitters without adopting vel./acc loss for training in the supplementary video. Besides, we explore the influence of the downsampling rate d in VQ-VAE on the quality of synthesized motions. As shown in Table 3, it becomes harder for the VQ-VAE to represent the dance sequence when using larger downsampling rates, which leads us to choose $d = 8$ as a proper setting in our experiments.

TABLE 4: **Ablation study on pose VQ-VAE (3D joint rotation)**

Method	FID$_k$ ↓	FID$_g$ ↓
Ground Truth	17.10	10.60
pose VQ-VAE (3D position)	28.23	12.63
w/o. hybrid training (rec.)	38.07	12.23
rotation matrix→rotation 6D	30.77	12.56
w/o. vel./acc. loss	31.34	12.63
full pose VQ-VAE (rotation)	29.36	12.51

As to the model for 3D joint rotations, we focus on the modules that can boost the VQ-VAE to reach a comparable performance as 3D joint positions while synthesizing motion in the rotation format. We explore the contribution of (1) the proposed hybrid training strategy, (2) the representation of joint rotational angles, and (3) the velocity-and-acceleration loss under $\mathcal{L}_A$. As shown in Table 4, if the pose VQ-VAE is not trained through the proposed hybrid strategy, the FID$_k$ score drops sharply by 8.71 (30%). This poor base on motion representation and reconstruction directly leads to the low performance of subsequent choreographic results of the GPT, which can be referred from "w/o. hybrid training (cho.)" in Table 6. Lacking of the hybrid training strategy causes non-standard dance movements. As shown in Figure 10, for the model without hybrid training strategy, the agent fails to turn around when doing "foettes on pointe" but only kicks awkwardly (also see supplementary video), while the one with hybrid training can perform normally, suggesting the necessity of the proposed hybrid training strategy for robust motion synthesis in 3D rotation format. Besides, we compare the performances of two kinds of representation of 3D joint rotations: the rotation matrix and "rotation 6D" [54]. The "rotation 6D" is the first two rows of the 3×3 rotation matrix and is shown to be more continuous for motion representation than the rotation matrix in [54]. However, our experiment shows that training the VQ-VAE in rotation 6D worsens the quality of synthesized motion, where the FID$_k$ and FID$_g$ scores drop 1.41 and 0.05, respectively. This phenomenon is also consistent with that observed in [7], where the rotation matrix is prone to produce stabler performance in this task. A visual comparison of the results of the two representations can refer to the supplementary video. As to the "vel./acc. loss" that suppresses the jitters in 3D joint positions, we find it also works in the training of joint rotations. As shown in Table. 4, if training without this constraint, the two FID scores drop 1.98 and 0.12, respectively, while apparent jitters will occur in output motions of the pose VQ-VAE. This experiment shows that constraining the first (velocity) and second derivatives (acceleration) of the synthesized pose sequence is a generalized way to reduce the generated jitters in generated movements for both 3D position and 3D rotation formats.

Motion GPT. Similar to the VQ-VAE, we explore each module's contribution to the proposed actor-critic GPT for 3D positions and 3D rotations, respectively. For 3D positions, first, we explore the effect of quantized choreography memory by training a variant

TABLE 5: **Ablation study on motion GPT (3D joint position).**

Method	FID$_k$ ↓	FID$_g$ ↓	BAS ↑
w/o. quantization	42.71	147.28	–
w/o. cross-cond. att.	37.41	15.52	–
w/o. CME w/o. actor critic	28.75	11.82	0.2245
w/o. CME (original *Bailando*)	28.16	9.62	0.2332
w/o. actor critic	20.89	10.18	0.2109
full actor-critic GPT (position)	17.66	11.30	0.2597

TABLE 6: **Ablation study on motion GPT (3D joint rotation).**

Method	FID$_k$ ↓	FID$_g$ ↓	BAS ↑
w/o. hybrid training (cho.)	39.09	13.66	–
w/o. CME	28.29	12.71	0.2437
w/o. actor critic	20.88	10.09	0.2264
full actor-critic GPT (rotation)	17.59	10.10	0.2720

GPT directly regress to the encoding features of 3D joint sequence via an L_2 Loss. As shown in Table 5, the FID$_g$ drops 135.41 for variant “w/o. quantization” (compared to “w/o. CME w/o. actor critic”, same below), while the generated dance sequences contain frequent jitters in vision, which shows the quantization of dancing positions is essential to our proposed framework. Second, to test the effectiveness of the proposed cross-conditional causal attention, we substitute it to causal attention, and train two motion GPTs for upper and lower half bodies separately. The motion quality scores of “w/o. cross-cond. att.” drop 8.66 (30%) and 3.70 (31%) (compared to “w/o. CME w/o. actor critic”), respectively. The main reason for the poor performance is that the generated dances of contain frequent asynchronization of upper and lower half bodies, while the proposed cross-conditional attention layer can effectively prevent such situations via the interaction of information between the half bodies.

Next, we evaluate the effectiveness of contextual music encoding by training a variant of the model that directly samples music features into the same frequency as pose codes and feeds them into the GPT without using CME. As shown in Table 5, the FID$_k$ score for this variant is significantly worse than the full actor-critic GPT for 3D positions, with a 59% decrease to 10.50. In contrast, the full actor-critic GPT performs close to the ground truth, with a FID$_k$ score of 17.66 compared to 17.10. The kinetic feature, which consists of motion velocities and energies, is affected by this change, indicating that contextual augmentation of music features can improve the GPT's ability to generate movements that are more physically realistic. In addition, we observe that CME leads the GPT to produce smoother movements by reducing irregular motion changes and choreographing repetitive movements for repeated melodies (see supplementary video). These results demonstrate the critical role of long-term understanding of music in synthesizing human-like movements in dance generation. At last, we compare the motion quality and music-motion consistency between the model with (denoted as “full actor-critic GPT”) and without actor-critic finetuning (denoted as “w/o. actor critic”). After the actor-critic learning, the beat-align score (BAS) of motion GPT increases from 0.2109 to 0.2597, proving the effectiveness of the reinforcement learning scheme with the proposed beat-align reward. Meanwhile, by constraining the consistency with music, the actor-critic finetuning process can also enhance the motion quality on choreography and saliently improves the FID$_k$ score by 3.23 (15%).

For 3D rotations, first, we study the influence of the hybrid training strategy on the choreography of motion GPT. Specifically, we train a GPT based on the learned quantized codebook of “w/o. hybrid training (rec.)” in Table 4. As shown in Table 6, without hybrid training, the motion quality scores of choreographic results degrade to 39.09 and 13.66, respectively, with significant drops of 21.50 (122%) and 3.56 (25%) from the GPT trained with full pose VQ-VAE; even worse, *Bailando* will degenerate to the point where it is worse than FACT [7] (FID$_k$: 39.09 *v.s.* 35.35). Moreover, When training motion GPT, dance motion sequences need to be encoded and quantized to code numbers as the regression target of GPT. This is because the 3D rotation format has spatial ambiguity, as discussed in Section 3.2, which can cause different poses to be clustered in one code, making it harder for the GPT to learn the laws of choreography from the codebook summarized from 3D rotation data. This results in a relatively lower FID$_g$ compared to the complete model (13.66 *v.s.* 10.10). Therefore, the proposed hybrid training strategy is necessary not only for the reconstruction quality of the pose VQ-VAE, but also for the final choreographed dance of the motion GPT In addition, we also separately test the effects of contextual music encoding and actor-critic learning on the GPT for 3D rotation outputs, following the experiments for 3D positions. As shown in Table 6, when CME was removed, FID$_k$ and FID$_g$ dropped by 10.70 (60%) and 2.61 (26%), respectively, while omitting actor-critic learning resulted in a 0.0456 (17%) lower BAS. These results are similar to the conclusions drawn from the corresponding ablation experiments. Therefore, we can draw the same conclusions on the usefulness of these two parts as those for 3D positions.

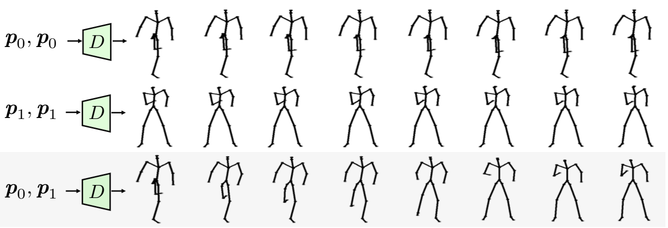


Fig. 11: **Interpretability of choreographic memory code.** The sequence of a single code is decoded to a static pose, while the sequence of two various codes is decoded to a smooth transition between two poses, which means each code represents a dancing-style pose and the decoder links poses of different codes to movements.

4.3 Interpretability of Choreographic Memory

In this work, we propose to summarize meaningful dancing units into the codebook via pose VQ-VAE in an unsupervised manner. To understand what kind of dance unit is learned in the choreographic memory, we visualize the latent codes and find each code represents a unique 3D dancing-style pose. As revealed in Figure 11, the first and the second rows are 3D poses decoded from $\boldsymbol{p}_0 = [4, 4]$ and $\boldsymbol{p}_1 = [5, 5]$, respectively, where the former one is doing right leg lifting and the latter is right bicep curl. The decoded pose will keep static for repeating codes, and will make a smooth transition between postures of different codes. As shown in the third row of Figure 11, the decoded 3D poses of $[\boldsymbol{p}_0, \boldsymbol{p}_1]$

starts with the posture of $\boldsymbol{p}_0$, while gradually putting down the leg and blending the arm towards the pose of $\boldsymbol{p}_1$. Furthermore, for an arbitrary combination of learned choreographic memory codes, the decoders can synthesize fluent movement based on the represented dance positions, which can be observed in the supplementary video. With such characteristics, the choreography process becomes interpretable in proposed *Bailando* as a process of selecting and sorting the quantized dance positions from the learned choreographic memories, instead of a black box as in most previous works.

4.4 Comparison to EDGE [37]

We conduct a comparative experiment to EDGE, a denoising-diffusion-based dance generation pipeline. We follow the same experimental setting as all other compared methods, where we generate 40 dance sequences in AIST++ test set using the *official* code [37], and sample each of them to a length of 20 seconds. When computing the local minimum of motion magnitude as needed by BAS, since the output frame rate of EDGE is 30 fps, we duplicate its frames to 60 fps to align with the same hyper-parameter in Equation 24 as other methods. The FID$_k$, FID$_g$, DIV$_k$, DIV$_g$ and BAS of EDGE are 77.56, 55.12, 6.55, 2.20 and 0.2488, respectively. We also conduct a separate user study to compare the performance of *Bailando*++ and EDGE by 25 participants. The wining rate of *Bailando*++ is 56.4% in average, with 19.2% of standard deviation and $[48.3\%, 64.5\%]$ of the 95% confidence interval. Based on the experiment, *Bailando*++ achieves higher quantitative scores while performing a comparable visual quality based on the user study. A visual comparison can be found in the supplementary video.

5 Discussion and Conclusion

In this paper, we address the spatial and temporal challenges of 3D dance generation by proposing a novel framework named *Bailando*++, which is composed of a choreographic memory to address the spatial constraint by encoding and quantizing dancing-style poses, and an actor-critic GPT to realize the temporal coherency with music that translates and aligns various motion tempos and music beats. Experiments on the standard benchmark (*i.e.*, AIST++ dataset) along with user studies show that *Bailando* achieves state-of-the-art performance both qualitatively and quantitatively.

Quantitative Metrics *vs* Subjective Evaluations. Although EDGE does not achieve satisfactory quantitative scores (FID and Div), it demonstrates significantly more appealing performance compared to other state-of-the-art methods in the subjective user study, which is also acknowledged in [37]. To analyze this phenomenon, we provide a discussion on these two metrics and their limitations that result in inconsistencies with subjective evaluations. First, FID is a commonly used metric in generative tasks to quantify the dissimilarity between the distribution of generated data and the real data. When examining the distribution of AIST++, a significant portion of the dance sequences exhibits an organized choreography consisting of repeated basic motion units *within* each sequence, while maintaining distinct motion patterns across different sequences. Given this characteristic, achieving satisfactory FID scores on AIST++ necessitates that the generated motion also possesses distinct and unique inter-sequence motion patterns. From a generative standpoint, this requirement aligns with the expectation that generated results should adhere to the same distribution as the source training data. However, if a model tends to choreograph in freestyle, a dance may incorporate hybrid incomplete motion units and casual movements within one sequence. Such choreography can present intense and vibrant movements, which gives an advantage in the pairwise comparisons during the user study, but will make the distribution of generated motion deviate from that of AIST++ data, since each sequence lacks a distinct pattern after taking an average of features in a long term. To address this limitation of solely relying on FID, future works can consider incorporating metrics derived from subjective user studies, as discussed in [37], to provide a more comprehensive evaluation. Besides FID, we compute the Div (Diversity) metric in this work, following the settings of AIST++. Specifically, we extract motion features for each frame within the first 20 seconds of each generated sequence and calculate the average over this duration to represent the feature of the sequence. In contrast, the metrics in [7] are computed on the first 5 seconds for feature calculation. Using a longer averaging duration emphasizes the need for independence of movements across sequences. If motion patterns appear frequently across sequences, even if the intra-sequence movements appear diverse, the inter-sequence features over the longer duration can become similar, resulting in lower Div scores. In our experiments, EDGE achieves a Div$_k$ score of 6.55. However, if we only consider the first 5 seconds of motion, as done in [37], this value would significantly increase to 10.32. Therefore, the current metric setting prioritizes dissimilarity between sequences over a longer duration, aiming to capture distinct inter-sequence characteristics.

Limitation on Music in the Wild. Although our method can be extended to dances with music in the wild, as demonstrated in the supplementary video, addressing the domain gap between AIST++ music and wild ones requires on-policy reinforcement tuning to achieve satisfactory results. Integrating a robust music encoder originally trained on wild music, such as Jukebox [39] into the system can serve as a strong acoustic prior for broader applications, as shown in [37]

Acknowledgement. This study is supported under the RIE2020 Industry Alignment Fund Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner(s). This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-PhD/2021-01-031[T]). This work is partially supported by the NTU NAP grant.

References

[1] “Mixamo,” https://www.mixamo.com/. 1, 2

[2] O. Alemi, J. Françoise, and P. Pasquier, “Groovenet: Real-time music-driven dance movement generation using artificial neural networks,” *Networks*, vol. 8, no. 17, p. 26, 2017. 1, 3, 4

[3] H.-K. Kao and L. Su, “Temporally guided music-to-body-movement generation,” *ACM MM*, 2020. 1

[4] T. Tang, J. Jia, and H. Mao, “Dance with melody: An lstm-autoencoder approach to music-oriented dance synthesis,” *ACM MM*, 2018. 1, 3, 4

[5] S. Ginosar, A. Bar, G. Kohavi, C. Chan, A. Owens, and J. Malik, “Learning individual styles of conversational gesture,” *CVPR*, 2019. 1

[6] H. Ahn, J. Kim, K. Kim, and S. Oh, “Generative autoregressive networks for 3d dancing move synthesis from music,” *IEEE Robot. and Automat. Letters*, vol. 5, pp. 3501–3508, 2020. 1

[7] R. Li, S. Yang, D. A. Ross, and A. Kanazawa, “Ai choreographer: Music conditioned 3d dance generation with aist++,” in *ICCV*, 2021. 1, 3, 4, 6, 9, 10, 11, 12, 13

[8] Z. Ye, H. Wu, J. Jia, Y. Bu, W. Chen, F. Meng, and Y. Wang, “Choreonet: Towards music to dance synthesis with choreographic action unit,” *ACM MM*, 2020. 1

[9] C. Kang, Z. Tan, J. Lei, S.-H. Zhang, Y.-C. Guo, W. Zhang, and S.-M. Hu, “Choreomaster: Choreography-oriented music-driven dance synthesis,” in *SIGGRAPH*, 2021. 1, 3

[10] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *NeurIPS*, 2017. 1

[11] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, 2019. 2, 6

[12] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *NeurIPS*, 2000. 2

[13] L. Siyao, W. Yu, T. Gu, C. Lin, Q. Wang, C. Qian, C. C. Loy, and Z. Liu, "Bailando: 3d dance generation by actor-critic gpt with choreographic memory," in *CVPR*, 2022. 2, 3, 7

[14] A. Aristidou, J. Lasenby, Y. Chrysanthou, and A. Shamir, "Inverse kinematics techniques in computer graphics: A survey," in *Computer graphics forum*, 2018. 2

[15] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: A skinned multi-person linear model," *SIGGRAPH Asia*, vol. 34, no. 6, pp. 248:1–248:16, Oct. 2015. 2, 4, 9

[16] A. Lamouret and M. van de Panne, "Motion synthesis by example," in *Comput. Animat. and Simulat.*, 1996. 3

[17] O. Arikan and D. A. Forsyth, "Interactive motion generation from examples," *ACM TOG*, vol. 21, no. 3, pp. 483–490, 2002. 3

[18] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," in *SIGGRAPH*, 2008. 3

[19] J. Lee, J. Chai, P. S. Reitsma, J. K. Hodgins, and N. S. Pollard, "Interactive control of avatars animated with human motion data," in *Annual Conf. on Comput. Graph. and Interactive Tech.*, 2002. 3

[20] M. Lee, K. Lee, and J. Park, "Music similarity-based approach to generating dance motion sequence," *Multimedia tools and applications*, vol. 62, no. 3, pp. 895–912, 2013. 3

[21] S. Fukayama and M. Goto, "Music content driven automated choreography with beat-wise motion connectivity constraints," *SMC*, pp. 177–183, 2015. 3

[22] D. Holden, J. Saito, and T. Komura, "A deep learning framework for character motion synthesis and editing," *ACM TOG*, vol. 35, no. 4, pp. 1–11, 2016. 3

[23] N. Yalta, S. Watanabe, K. Nakadai, and T. Ogata, "Weakly-supervised deep recurrent neural networks for basic dance step generation," in *IJCNN*, 2019. 3

[24] R. Huang, H. Hu, W. Wu, K. Sawada, M. Zhang, and D. Jiang, "Dance revolution: Long-term dance generation with music via curriculum learning," in *ICLR*, 2021. 3, 9, 10

[25] S. Yan, Z. Li, Y. Xiong, H. Yan, and D. Lin, "Convolutional sequence generation for skeleton-based action synthesis," in *ICCV*, 2019. 3

[26] X. Ren, H. Li, Z. Huang, and Q. Chen, "Self-supervised dance video synthesis conditioned on music," in *ACM MM*, 2020. 3

[27] J. P. K. Ferreira, T. M. Coutinho, T. L. Gomes, J. F. Neto, R. Azevedo, R. Martins, and E. R. Nascimento, "Learning to dance: A graph convolutional adversarial network to generate realistic dance motions from audio," *Comput. Graph.*, vol. 94, pp. 11–21, 2021. 3

[28] H.-Y. Lee, X. Yang, M.-Y. Liu, T.-C. Wang, Y.-D. Lu, M.-H. Yang, and J. Kautz, "Dancing to music," in *NeurIPS*, 2019. 3

[29] G. Sun, Y. Wong, Z. Cheng, M. S. Kankanhalli, W. Geng, and X. Li, "Deepdance: music-to-dance motion choreography with adversarial learning," *TMM*, vol. 23, pp. 497–509, 2020. 3

[30] J. Li, Y. Yin, H. Chu, Y. Zhou, T. Wang, S. Fidler, and H. Li, "Learning to generate diverse dance motions with transformer," *ArXiv*, vol. abs/2008.08171, 2020. 3, 9, 10

[31] B. Li, Y. Zhao, and L. Sheng, "Danceformer: Music conditioned 3d dance generation with parametric motion transformer," in *AAAI*, 2022. 3

[32] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 3

[33] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NeurIPS*, 2014. 3

[34] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *ArXiv*, vol. abs/1706.03762, 2017. 3

[35] G. Tevet, S. Raab, B. Gordon, Y. Shafir, D. Cohen-Or, and A. H. Bermano, "Human motion diffusion model," *arXiv preprint arXiv:2209.14916*, 2022. 3

[36] M. Zhang, Z. Cai, L. Pan, F. Hong, X. Guo, L. Yang, and Z. Liu, "Motiondiffuse: Text-driven human motion generation with diffusion model," *arXiv preprint arXiv:2208.15001*, 2022. 3

[37] J. Tseng, R. Castellon, and C. K. Liu, "Edge: editable dance generation from music," in *CVPR*, 2023. 3, 13

[38] W. Zhuang, C. Wang, S.-Y. Xia, J. Chai, and Y. Wang, "Music2dance: Music-driven dance generation using wavenet," *ArXiv*, vol. abs/2002.03761, 2020. 4, 9, 10

[39] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020. 4, 5, 7, 9, 13

[40] W. Yan, Y. Zhang, P. Abbeel, and A. Srinivas, "Videogpt: Video generation using vq-vae and transformers," *arXiv preprint arXiv:2104.10157*, 2021. 4

[41] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *CVPR*, 2021. 4, 5, 7

[42] X. Chen and K. He, "Exploring simple siamese representation learning," in *CVPR*, 2021. 5

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, vol. 30. Curran Associates, Inc., 2017. 6

[44] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *CVPR*, 2016. 7

[45] M. Janner, Q. Li, and S. Levine, "Reinforcement learning as one big sequence modeling problem," in *NeurIPS*, 2021. 7

[46] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *arXiv preprint arXiv:2106.01345*, 2021. 7

[47] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2014. 9

[48] A. Karpathy, https://github.com/karpathy/minGPT, 2020. 9

[49] Y. Jin, J. Zhang, M. Li, Y. Tian, H. Zhu, and Z. Fang, "Towards the automatic anime characters creation with generative adversarial networks," *arXiv preprint arXiv:1708.05509*, 2017. 9

[50] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *NeurIPS*, vol. 30, 2017. 10

[51] K. Onuma, C. Faloutsos, and J. K. Hodgins, "Fmdistance: A fast and effective distance function for motion capture data." in *Eurographics*, 2008, pp. 83–86. 10

[52] M. Müller, T. Röder, and M. Clausen, "Efficient content-based retrieval of motion capture data," in *SIGGRAPH*, 2005, pp. 677–685. 10

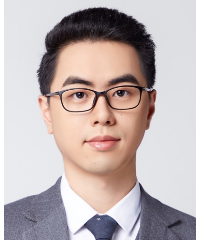
[53] D. Gopinath and J. Won, "fairmotion - tools to load, process and visualize motion capture data," Github, 2020. [Online]. Available: https://github.com/facebookresearch/fairmotion 10

[54] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *CVPR*, 2019. 11

Li Siyao is a PhD student at MMLab of Nanyang Technological University, Singapore, advised by Prof. Chen Change Loy. Before that, he was a full-time researcher in SenseTime Research. He was awarded the Singapore AISG scholarship in 2022. His current interests include AI techniques for content creation.

Weijiang Yu is a Ph.D. student from the School of Computer Science and Engineering, Sun Yat-sen University, advised by Nong Xiao. He has closely collaborated with Prof. Bernard Ghanem in the KAUST and Prof. Eric P. Xing in the CMU. His research interests mainly include interactive vision system, structural representation learning, and multimodality like visual question answering, visual commonsense reasoning, video captioning and understanding. His current interest is to explore (Transformer-based) pretrained models for cognitive reasoning in video-based multimodality.

Tianpei Gu received the M.Eng. degree from University of California, Los Angeles and B.S degree from University of Maryland, College Park. He is currently a Research Engineer at Lexica. His research is in Computer Vision, especially generative models, diffusion models and human motion.



Chunze Lin is currently a Senior Research Scientist at SenseTime, ARFace group. Chunze received his M.Eng. degree from Tsinghua University in 2019, advised by Professor Jiwen Lu and Professor Jie Zhou. He received M.Eng. degree from Ecole Centrale de Nantes, France in 2019 as a dual degree program. His research interests are computer vision, deep learning, image stylization, 3D face reconstruction, pedestrian detection and face alignment.

Quan Wang received the B.S. degree in electronic engineering from Tsinghua University, China. He is currently a research scientist at SenseTime. His research interests include computer vision and 3D reconstruction.

Chen Qian received the BEng degree from the Institute of Interdisciplinary Information Science, Tsinghua University, in 2012, and the MPhil degree from the Department of Information Engineering, the Chinese University of Hong Kong, in 2014. He is currently working at SenseTime as a research director. His research interests include human-related computer vision and machine learning problems.

Chen Change Loy (Senior Member, IEEE) is currently a Nanyang Associate Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. He received the PhD degree in computer science from the Queen Mary University of London, in 2010. Prior to joining NTU, he served as a research assistant professor with the Department of Information Engineering, The Chinese University of Hong Kong, from 2013 to 2018. His research interests include computer vision and deep learning with a focus on image/video restoration and enhancement, generative tasks, and representation learning. He serves as an associate editor of the IEEE Transactions on Pattern Analysis and Machine Intelligence and the International Journal of Computer Vision. He also serves/served as an Area Chair of top conferences such as ICCV, CVPR and ECCV.

Ziwei Liu (Member, IEEE) is currently a Nanyang Assistant Professor at Nanyang Technological University, Singapore. His research revolves around computer vision, machine learning and computer graphics. He has published extensively on top-tier conferences and journals in relevant fields, including CVPR, ICCV, ECCV, NeurIPS, ICLR, ICML, TPAMI, TOG and Nature - Machine Intelligence. He is the recipient of Microsoft Young Fellowship, Hong Kong PhD Fellowship, ICCV Young Researcher Award, HKSTP Best Paper Award and WAIC Yunfan Award. He serves as an Area Chair of CVPR, ICCV, NeurIPS and ICLR, as well as an Associate Editor of IJCV.