Convolutional Sequence Generation for Skeleton-Based Action Synthesis

Sijie Yan* Zhizhong Li* Yuanjun Xiong Huahan Yan
Dahua Lin
Department of Information Engineering, The Chinese University of Hong Kong
{ys016, lz015, dhlin}@ie.cuhk.edu.hk, {bitxiong, huahanyan2}@gmail.com

Abstract

In this work, we aim to generate long actions represented as sequences of skeletons. The generated sequences must demonstrate continuous, meaningful human actions, while maintaining coherence among body parts. Instead of generating skeletons sequentially following an autoregressive model, we propose a framework that generates the entire sequence altogether by transforming from a sequence of latent vectors sampled from a Gaussian process (GP). This framework, named Convolutional Sequence Generation Network (CSGN) [1], jointly models structures in temporal and spatial dimensions. It captures the temporal structure at multiple scales through the GP prior and the temporal convolutions; and establishes the spatial connection between the latent vectors and the skeleton graphs via a novel graph refining scheme. It is noteworthy that CSGN allows bidirectional transforms between the latent and the observed spaces, thus enabling semantic manipulation of the action sequences in various forms. We conducted empirical studies on multiple datasets, including a set of high-quality dancing sequences collected by us. The results show that our framework can produce long action sequences that are coherent across time steps and among body parts.

1. Introduction

When the dancer is stepping, jumping and spinning on the stage, attentions of all audiences are attracted by the stream of the fluent and graceful movements. Building a model that is capable of dancing is as fascinating a task as appreciating the performance itself. In this paper, we aim to generate *long-duration* human actions represented as skeleton sequences, *e.g.* those that cover the entirety of a dance, with hundreds of moves and countless possible combinations.

Skeleton-based action synthesis [2,5,6,9,17,29] is gaining ground in recent years. It is used to aid human-centric video generation [7] and sees potential applications in AR and 3D character animations. Existing tools for generating skeleton sequences [2, 5, 9, 17, 29] are mostly extended from frameworks for *action prediction*, which aim at predicting short sequences of near future given a few seen frames. Therefore they are usually based on autoregressive models such as LSTM [14], GRU [1] and Seq2seq [24]. When applied to generating long-duration action sequences, two inherent limitations of these methods emerge. First, they rely on the assumption of *Markovian dependency* to model temporal relations, *i.e.* the generation of a new frame at time step t depends on the hidden states of a few preceding frames. This makes it nontrivial to express structures at multiple temporal scales. Second, the manner of generating frames one by one along the time dimension creates a barrier for leveraging backward dependencies. In particular, a frame generated at a future time step cannot alter the ones in preceding steps.

We propose the *Convolutional Sequence Generation Network (CSGN)*, a new framework for skeleton-based action

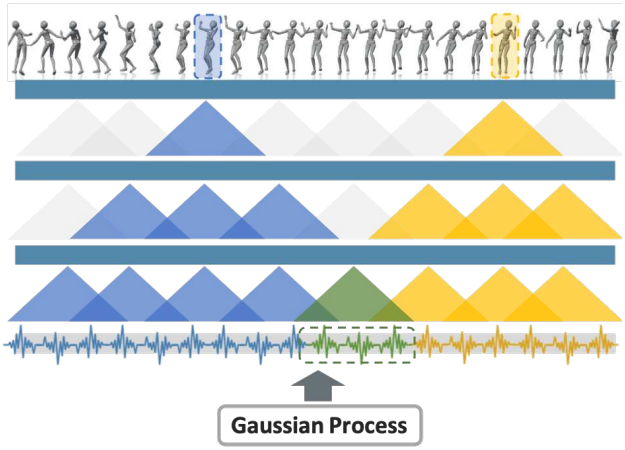

Figure 1. CSGN starts with latent signal sequences sampled from Gaussian process prior and gradually constructs skeleton sequences of the dance move via convolution and upsampling operations. The blue skeleton and yellow skeleton share part of latent signals (shown in green), resulting in a temporal dependency between them. The relation for longer duration is brought by the Gaussian process.

*The two authors contributed equally.
[1]Codes and data at *https://github.com/yysijie/CSGN*.

generation. Unlike autoregressive models, CSGN transforms a sequence of latent vectors drawn from a fixed Gaussian process to a sequence that follows the data random process. Specifically, the sampled latent vectors are transformed into a sequence of skeletons through a convolutional network in a layer-by-layer manner. This network interleaves spatial-temporal graph convolution layers with spatial-temporal graph upsampling operators, thus incorporating temporal and spatial relations in multiple scales gradually. As shown in Figure 1, the resultant skeleton at each step depends upon a sub-sequence of the input, and the corresponding sub-sequences for neighboring skeletons overlap. This ensures the generated sequence is temporally coherent across a short range of time steps. Additionally, the relationships in long temporal ranges is established by the Gaussian process.

To enable conditional sampling, we devise an encoding network along with the generative network introduced above. The encoding network transforms an observed sequence of skeletons back into a sequence of latent vectors by reducing the spatial resolution of the skeleton via *graph coarsening*. This two-way transformation allows CSGN to manipulate action sequences in various ways, *e.g*. complementing, forecasting, and semantic editing.

CSGN is evaluated on two datasets: *NTU-RGB+D* [22], a real-world action dataset, and *MikuDance*, a new dataset collected by us. The former comprises short action sequences obtained from Kinect sensors [28]. The latter contains 201 pieces of long dancing sequences. Each piece lasts for 3–5 minutes, summing up to 10 hours in total. We conducted quantitative and qualitative assessments of the CSGN in tasks of generating action sequences and performing action manipulations. Experiments show that CSGN overcomes the limitation of existing autoregressive approaches and is able to generates long action sequences in high quality.

2. Related Work

Action prediction. Previous work on action prediction mostly relies on autoregressive models. Fragkiadaki *et al*. [9] propose the *Encoder-Recurrent-Decoder (ERD)* model, incorporating encoder and decoder networks before and after recurrent units. The accumulation of errors is diminished by its denoising component. Martinez *et al*. [17] propose a framework based on Seq2seq [24], which predicts velocities rather than positions of joints. Li *et al*. [29] propose the Ac-LSTM, which mixes synthesized frames with observed frames in training, thus enhancing the model's capability for error correction. Butepage *et al*. [5] propose to encode a series of previous frames to a latent representation, and decode future sequences therefrom. For all these methods, frames are generated step by step, with new ones depending on previously generated frames. Our framework, instead, generates the entire sequence directly by convolution, which naturally captures the temporal structure at multiple scales.

Action synthesis. Generative Adversarial Network [10], a popular paradigm for generative models, has been applied to human action generation [2]. HP-GAN [2] combines the Seq2seq framework with GAN for motion prediction, where a Seq2Seq model is used as the generator and a fully connected network is used as the discriminator. Unlike HP-GAN, we formulate both the generator and the discriminator as graph convolutional networks. Cai *et al*. [6] propose a two-stage GAN for skeleton motion generation, where the first stage learns to generate spatial signals of pose, and the second stage generates temporal signals represented as latent vector sequences. Our model jointly models spatial-temporal signals using spatial-temporal graph convolutions.

Graph convolution. Graph neural networks have received increasing attention [16, 27]. There are two major types of graph convolutions. *Spectral graph convolution* utilizes the convolution theorem and operates on the spectral domain via graph Laplacians [4, 8, 12]. *Spatial graph convolution* directly operates on the vertices and their neighbors [18, 27]. Graph convolution was used for action recognition in [27]. In this work, we use graph convolution to capture the spatial structure of skeletons. Moreover, to allow the computation at different resolutions of a graph, we introduce the graph coarsening and refining operations.

3. Convolutional Sequence Generation

Inspired by the observation that an action is continuously composed of short elemental motions, *e.g*. jumping, turning, kicking, we propose *Convolutional Sequence Generation Network (CSGN)*, a network architecture that generates an action sequence by transforming a sequence of latent vectors.

As shown in Figure 2, to generate an action sequence of length T, CSGN first samples a sequence of *latent vectors* from a Gaussian process, which is expected to contain the abstract and slowly changing motion signals. This latent sequence can be represented by a tensor of shape $(C_0, 1, T_0)$ which contains T_0 latent vectors of dimension C_0. As each latent vector is assumed to contain information over multiple time steps, we let $T_0 < T$. Subsequently, it hierarchically decodes the latent sequence via a series of *blocks*, each comprised of spatial-temporal graph convolutions, into an action sequence. The action sequence is a spatial-temporal graph of shape (C, V, T), where T is the number of time steps, V is the number of vertices of the skeleton graph with each corresponds to a joint, and C is the dimension of the data vector associated with each vertex, which may include information like positions, orientations and rotations.

Each block of the transformation above takes as input a spatial-temporal graph with shape (C_k, V_k, T_k), denoted by $\mathcal{F}_k$, and outputs a feature graph $\mathcal{F}_{k+1}$ at the next level,

$$\mathcal{F}_{k+1} = h\left(\mathrm{Conv}_{st}(\mathrm{Up}_{st}(\mathcal{F}_k))\right). \tag{1}$$

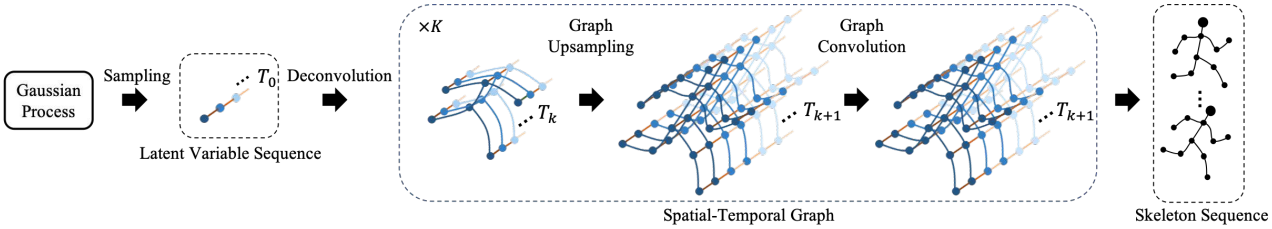

Figure 2. The overall pipeline of CSGN. Starting with a latent vector sequence sampled from Gaussian processes, CSGN gradually increases the spatial and temporal resolutions of the graph. In the end, the latent signals are transformed into a sequence of human skeletons.

This block carries out the computation in three steps: $\mathcal{F}_k$ is first upsampled by a spatial-temporal graph upsampling operator Up_{st}, which is introduced in Section 3.2, and then transformed by a spatial-temporal graph convolution Conv_{st}. Finally, a nonlinear activation function h is applied. Here, the upsampling operator increases both the spatial and the temporal resolution with $V_{k+1} \geq V_k$ and $T_{k+1} = 2T_k$. The subsequent convolution fuses the features over a spatial-temporal neighborhood into the output. The last block of the network outputs the 3D locations of the human skeleton joints at every time step, thus forming an action sequence.

3.1. Latent Vector Sequence Generation

The CSGN model starts by sampling a sequence of T_0 *latent vectors* $\mathcal{F}_0 = (\mathbf{z}_t)_{t=0}^{T_0}$. As mentioned, these vectors will be gradually upsampled to a sequence of skeletons of T frames, with $T_{k+1} = 2T_k$ for each upsampling operation. For a CSGN with K levels of upsampling, we have $T = T_0 \cdot 2^K$. We use the *Gaussian process* [20] prior to generate the latent vector sequences, where the zero-mean Gaussian process with Radial Basis Function kernel (RBF) is adopted to enforce long-term temporal correlations. Particularly, the Gaussian process has the form of $(z_t^{(c)})_t \sim GP(0, \kappa)$, where the covariance function κ is defined as

$$\kappa(t, t') = \exp\left(-\frac{|t - t'|^2}{2\sigma_c^2}\right). \tag{2}$$

Here, $z_t^{(c)}$ is the c-th component of $\mathbf{z}_t$. The intuition is that latent variables closer in time are more correlated than those faraway. We assume that different channels of the latent space are independent, and thus they respectively constitute independent Gaussian processes. The parameter σ_c defines the *characteristic length-scale* for each channel.

Figure 3 shows several sequences sampled from Gaussian processes of different σ_c. It shows that a larger σ_c allows latent vectors at distant time steps to be strongly correlated, while a smaller σ_c yields signals with frequent fluctuations. To simultaneously model a variety of temporal interaction of different scales, we set $\sigma_c = (c/C_0)\bar{\sigma}$ to span a spectrum of values. Here, $\bar{\sigma}$ is a hyperparameter that defines the *base scale*, which will be determined empirically in Section 6.

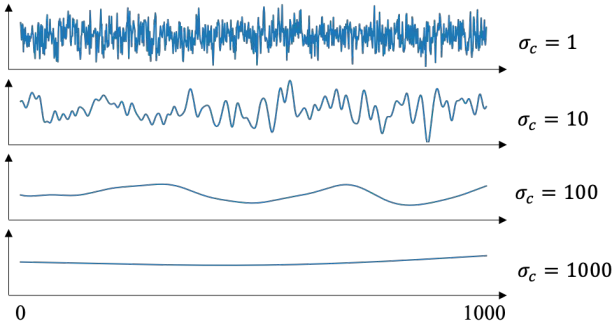

Figure 3. Gaussian processes with different characteristic length-scale σ_c. Larger σ_c correlates distant latent variables stronger.

Block	Operations	MikuDance	NTU RGB+D
In	Noise from Gaussian process	$(1024,\ 1,\ \frac{T}{16})$	$(1024,\ 1,\ \frac{T}{16})$
0	$\mathrm{LR} \circ \mathrm{BN} \circ \mathrm{Conv}_{st}$	$(\ 512,\ 5,\ \frac{T}{16})$	$(\ 512,\ 5,\ \frac{T}{16})$
1	$\mathrm{LR} \circ \mathrm{BN} \circ \mathrm{Conv}_{st} \circ \mathrm{Up}_t$	$(\ 256,\ 5,\ \frac{T}{8})$	$(\ 256,\ 5,\ \frac{T}{8})$
2	$\mathrm{LR} \circ \mathrm{BN} \circ \mathrm{Conv}_{st} \circ \mathrm{Up}_{st}$	$(\ 128, 20,\ \frac{T}{4})$	$(\ 128, 11,\ \frac{T}{4})$
3	$\mathrm{LR} \circ \mathrm{BN} \circ \mathrm{Conv}_{st} \circ \mathrm{Up}_t$	$(\ 64, 20,\ \frac{T}{2})$	$(\ 64, 11,\ \frac{T}{2})$
4	$\mathrm{LR} \circ \mathrm{BN} \circ \mathrm{Conv}_{st} \circ \mathrm{Up}_{st}$	$(\ 32, 46,\ T)$	$(\ 32, 25,\ T)$
Out	$\tanh \circ \mathrm{Conv}_{st}$	$(\ 4, 46,\ T)$	$(\ 3, 25,\ T)$

Table 1. Design of the generator network, where LR is shorthand for LeakyReLU, operator Conv_{st} is the graph convolution, operator Up_{st} is the upsampling operator, and operator Up_t only upsamples the temporal dimension and keep graphical dimension intact.

3.2. Sequence Generation from Latent Vectors

Starting with a sampled latent vector sequence, CSGN gradually increases the resolutions in spatial and temporal dimensions with its generator network. The architecture of the generator is outlined in Table 1, where the sizes of spatial-temporal graphs are denoted as (C, V, T). To generate high-quality skeletons, we introduce two graph operations: *graph upsampling* and *spatially-varying graph convolution*, which will be illustrated in the following.

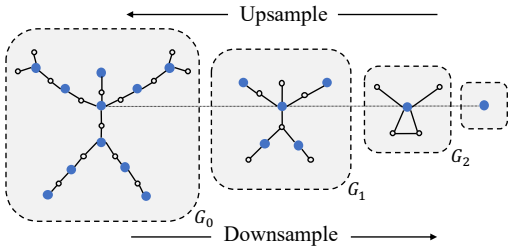

Figure 4. Graph pyramid for the NTU-RGB+D dataset. G_0 is the original skeleton graph. Graph pyramids provide a unambiguous guidance for performing graph upsampling and downsampling.

Graph upsampling/downsampling via graph pyramid. When building the generation model, we need to transform the latent vector sequence, which has no spatial extent, into the final output of spatial-temporal graph. One proven idea in image generation is to perform gradual upsampling [19], which increases the resolution and adds details in a layer-by-layer manner. However, as skeleton graphs are not regular grids, it is nontrivial to devise a path which we can follow to perform gradual upsampling. We solve this problem by introducing graph pyramids. A graph pyramid is constructed from a full resolution skeleton graph with $V_0 = V$ vertices. We perform graph subsampling by removing vertices as many as possible, and simultaneously ensuring that each removed vertex has at least one neighbor staying in the graph. We repeat this operation until there is only one vertex left. The result is a graph pyramid, where the i-th level of graph G_i is the output of the i-th iteration of subsampling. An example graph pyramid built on the skeleton graph of NTU-RGB+D is shown in Figure 4. By retracing the path from the top of the pyramid, the single vertex graph, to the bottom, the full skeleton graph, we now can easily obtain a path to perform gradual upsampling. To implement one graph upsampling operation, we first embed the coarser graph into the finer graph according to the pyramid and then assign values to any new vertex by averaging the values of its neighboring vertices. We further integrate the graph upsampling with the temporal upsampling to complete the operator Up_{st}. Likewise, the discriminator model or the inverse mapping model which need to transform skeleton graphs back to scalers or latent vectors can follow the pyramid from the bottom to the top as a path to perform gradual downsampling.

Spatially-varying graph convolution. Among the many variants of graph convolutions, spatial graph convolution groups neighboring vertices into different partitions and *shares weights* for vertices in the same partition. When combined with the temporal convolution jointly, the resultant spatial-temporal graph convolution forms an effective tool to model spatial-temporal information in skeleton-based action recognition [27]. In this work, we adopt this architecture in action sequence generation. As an extension of spatial graph convolution, the spatially-varying graph convolution further exploits the topology of the graph via localized operations. Compared to standard graph convolutions, the parameters of locally connected networks are *not shared* across vertices. The extra degree of freedom better respects the heterogeneity of different body parts in the skeleton graph.

3.3. Generative Adversarial Training of CSGN

The CSGN is trained via generative adversarial learning as in [10, 11]. Specifically, the generator G of the CSGN transforms a sequence of latent vectors into a sequence of skeleton graphs. Here the input to G is a vector sequence sampled from a Gaussian process instead of a random vector with independent components like in standard GANs. The training also involves a discriminator D, which is used to discriminate between the generated sequences from the real ones, thus providing feedbacks to the generator. Like G, the discriminator D also consists of a series of core blocks. It incorporates a spatial-temporal graph convolution Conv_{st}, a coarsening operator Coar_{st}, and a nonlinear activation h, as

$$\mathcal{F}_{k+1} = h(\mathrm{Coar}_{st}(\mathrm{Conv}_{st}(\mathcal{F}_k))). \tag{3}$$

The coarsening on the graph aspect follow the downsampling path described in Section 3.2. With the use of the coarsening operation, the temporal and spatial resolution is gradually reduced. The output of the last block will be fed to a fully-connected classifier, to produce a probability score. Some examples of generation is shown in Figure 6.

4. Action Manipulation

In practice, users often desire to control the generation process by setting certain conditions. One can a) provide an initial segment; b) give a series of disconnected segments at certain time-steps, and ask the system to fill in the rest. CSGN enables this by introducing a two-way transformation between the latent vectors and the skeleton sequences. Formally, it is formulated as *conditional sampling* (Figure 5). We first transform the conditioning sequences into latent vectors using an inverse mapping network. We then sample the missing time steps in the space of latent vectors. The filled latent vector sequence is finally transformed back to skeleton sequence via generator. According to different use cases, we describe two tasks: a) **probabilistic prediction**, where a skeleton sequence of the first few time steps is provided as condition, and b) **action completion**, where a few disjoint time ranges are provided as conditions.

Suppose that the data sequence $\mathcal{F}$ is partially observed. Denote the observed parts by $\mathcal{F}^K$ and the rest by $\mathcal{F}^U$, where K and U are subsets of time steps. Then the conditional sampling process can be expressed as

$$\mathcal{F} = G(\mathcal{F}_0), \text{ where } \mathcal{F}_0 \sim p(\mathcal{F}_0|\mathcal{F}^K). \tag{4}$$

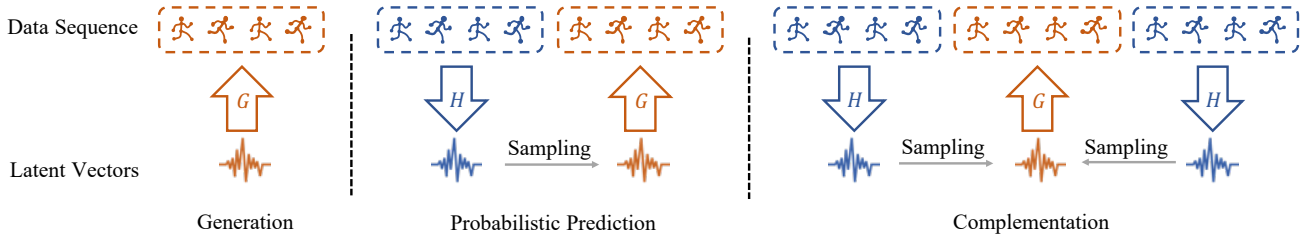

Figure 5. The pipeline of three tasks. Blue color denotes the conditional frames or latent vectors, while orange color denotes the sampled frames or latent vectors. Generator G and inverse mapping network H build a bridge between skeleton sequences and latent variables.

As G involves multiple layers of nonlinear computation, it is difficult to give the posterior distribution $p(\mathcal{F}_0|\mathcal{F}^K)$ exactly. Here, we resort to an approximated factorization, where

$$p(\mathcal{F}_0|\mathcal{F}^K) \approx p(\mathcal{F}_0^U|\mathcal{F}_0^K) \cdot p(\mathcal{F}_0^K|\mathcal{F}^K). \tag{5}$$

Accordingly, the conditional sampling can be carried out in three steps: (1) First sample the corresponding part in the latent space $\mathcal{F}_0^K$ conditioned on the partial observation. This step is done via an inverse mapping network H. The architecture of H is designed to be an opposite of G, where the core blocks have the form as in Equation (3). Following the practice of Cycle-GAN, we train the inverse mapping H using the cycle consistency loss. (2) Then derive the remaining part of the latent sequence by drawing $\mathcal{F}_0^U \sim p(\mathcal{F}_0^U|\mathcal{F}_0^K)$. (3) Finally generate the sequence of skeletons by $\mathcal{F}^U = G(\mathcal{F}_0^U)$. Examples of applying this process to action completion are shown in Figure 7.

5. MikuDance Dataset

Our goal is to generate long-duration action sequences, However, there lacks a large-scale, high-quality skeleton-based dataset with abundant long motions. For example, the *CMU Graphics Lab Motion Capture Database* [25] has 88 short dance clips ranging from several seconds to tens of seconds; and the *Panoptic* dataset [15] contains 25 minutes of dancing data from two actresses. We collected a new large-scale dancing action dataset named the *MikuDance*. MikuDance is built from handcrafted dancing performances from the MikuMikuDance (MMD) [26] animation program. MMD allows users to import 3D models into a virtual space that can be animated. Each piece contains the full animation of a song dance, with an average length of 4823 frames at 30 fps. We collected 201 distinct dances, totaling 10 hours with 1M frames. The skeleton data contains 46 joints, including joints of the ten fingers. In addition, the rotational information of bones are provided in the form of quaternions. With all these, the data is capable of driving a full-fledged 3D character model. We use this dataset to evaluate the capability of CSGN in generating long action sequences.

6. Experiment

In this section, we evaluate the performance of CSGN for human action generation. First we describe the quantitative metrics for measuring the generation quality and diversity. Then we compare CSGN with baseline models on two benchmark datasets: *NTU-RGB-D* and the newly built *MikuDance* dataset. Finally we provide ablation study of the components in CSGN to investigate their strengths and limitations.

NTU-RGB+D [23] is a daily action dataset with plenty of short-term skeleton sequences. Data are represented by 3D locations (X, Y, Z) of 25 body joints. It contains 56000 clips from 60 action categories. The coordinate data are captured by the Kinect [28] depth sensor. It is worth to note that the noisy depth signals set an unreasonable upper bound on the generation quality. To circumvent this problem, we preprocess the skeleton data by removing outlier values and smoothing the data along the time dimension with Savitzky-Golay [21] and median filters. The training and evaluation are then performed on this cleaned version of the dataset.

MikuDance is built by us using hand-crafted dancing data. It features very long action sequences with 201 dancing pieces and 1M frames. Quaternions are used to record the relative rotation of each bone w.r.t. its parent joint.

Implementation detail. All our models, both for NTU-RGB-D and MikuDance, and unless specified otherwise, are trained by WGAN-GP [11], using the Adam optimizer with betas $(0.5, 0.999)$ and learning rate 0.0002. The gradient penalty is 10, and we let the gradient norm regress to 0.1. Base scale $\bar{\sigma}$ is 100. Models for NTU-RGB-D are trained with 50 epochs, and those for MikuDance have 500 epochs.

6.1. Evaluation Metrics

Long-duration skeleton action generation is a relatively new task. It is important to have well-defined quantitative assessments for the generation quality of models. Two well-known metrics in image generation are the *Inception Score* (IS) [3], which feeds generated samples to a classification

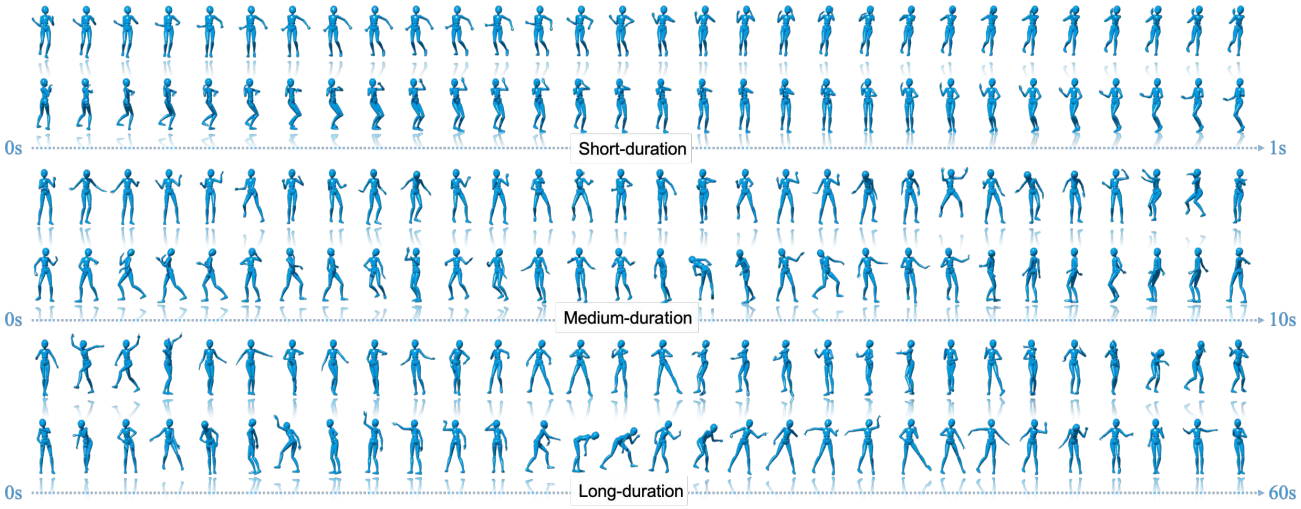

Figure 6. **Generation**. The skeleton sequences synthesized by CSGN on MikuDance. The duration of three rows from top to bottom are $1s$, $15s$ and $60s$. Our model has both smoothness in short-term synthesis and diversity in the long-duration generation.

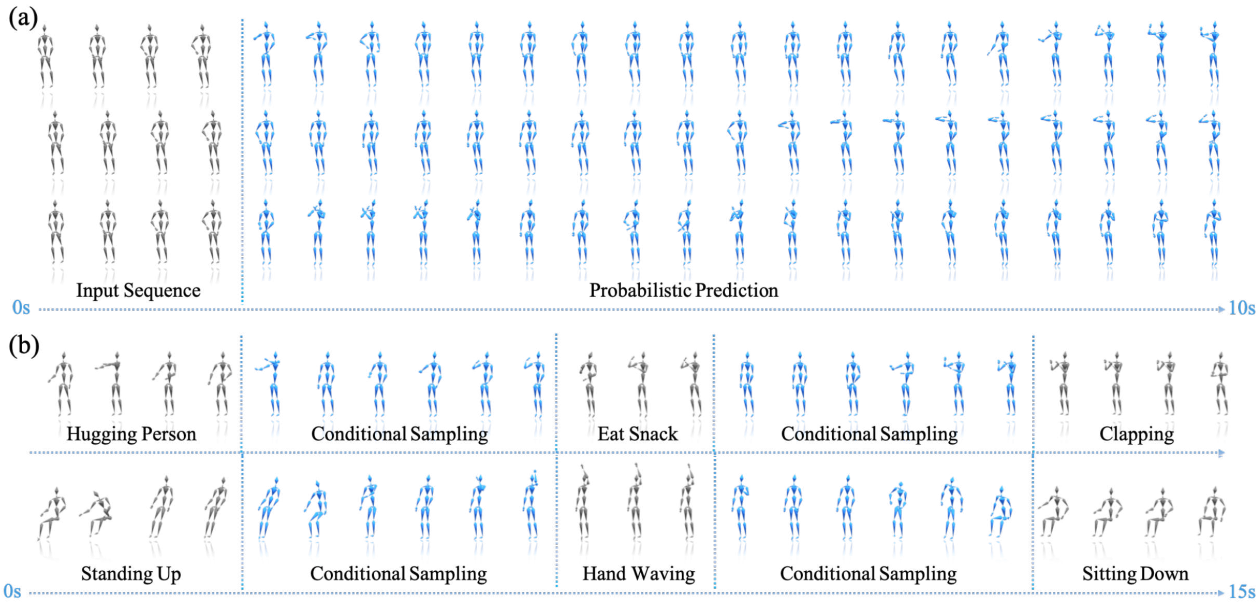

Figure 7. (a) **Probabilistic prediction** and (b) **completion** of daily actions on NTU-RGB+D. Gray poses are the given conditional sequences, while the blue skeleton sequences are the synthesized frames conditioned on the inputs.

model and analyze their output probabilities over all classes; and *Fréchet Inception Distance* (FID) [13], which measures the distance between the statistics of real and synthesized data on feature space. We extend these two basic metrics to evaluate on the spatial-temporal sequences. We use ST-GCN [27] to build action classifiers and train them on a different data splits w.r.t. generators. In MikuDance, each dance is cut into two halves for generation and classification respectively. In NTU-RGB+D, we use the train set of cross-subject benchmark for generation, and the validation set for classification. The top-1 and top-5 classification accuracies for MikuDance are 51.76% and 75.44%, and those for NTU-RGB+D are 60.52% and 86.83%, respectively. The input to the classification model is fixed to 32 frames. Below we describe three extensions to IS and FID. To compute them, we would generate $N = 1000$ long-duration skeleton sequences $\mathcal{F}^n, n = 1, \ldots, N$, and cut each sequence into $M = 1000$ short snippets $\mathcal{F}^{nm}$ of length 32.

Mean ensemble FID/IS evaluates the overall generation quality and diversity *across different sequences*, which is inspired by the ensemble average in stochastic processes. Fix an index $m \in \{1, 2, \ldots, M\}$, we can compute FID/IS on the set of snippets at the m-th position of all sequences. Mean ensemble FID/IS is then obtained by averaging the above FID/IS values over all position indices. Take the mean ensemble FID, denoted as FID^e, as an example, we have

$$\text{FID}^e = \frac{1}{M} \sum_m \text{FID}\Big(\{\mathcal{F}^{nm}\}_{n=1,\ldots,N}\Big). \tag{6}$$

Mean time FID/IS. The generation quality and diversity *within each sequence* is also critical for applications. If the generated sequence is static poses or just looping a simple action, it is hard for ensemble scores to detect them because different samples may still have different poses or motions. Inspired by the time average technique in stochastic process, we introduce the mean time FID/IS scores to characterize the generation quality within each generated sequence. We first obtain the time FID/IS scores for each generated sequence by computing the basic metrics on all snippets within each sequence. Then we average the scores across all sequences. Take the mean time FID, denoted as FID^t, as an example,

$$\text{FID}^t = \frac{1}{N} \sum_n \text{FID}\Big(\{\mathcal{F}^{nm}\}_{m=1,\ldots,M}\Big). \tag{7}$$

Short FID/IS is defined as the ensemble FID/IS at snippet index $m = 1$. Some baseline methods were designed for short-duration generation/prediction tasks, so we explicitly use this metric to measure the quality of the starting part of the generated sequences. It has the following form,

$$\text{FID}^s = \text{FID}\Big(\{\mathcal{F}^{n1}\}_{n=1,\ldots,N}\Big). \tag{8}$$

6.2. Comparing Generation Quality

We compare the generation qualities of CSGN and other baseline models on the MikuDance and NTU-RGB+D [22] datasets. The compared methods include two autoregressive models: ERD [9] and acLSTM [29], and two GAN-based models: HP-GAN [2] and Two-Stage [6]. Except for the Two-Stage [6], all compared methods are built upon RNN modules, which can generate arbitrarily long sequences. The quantitative results, measured by mean ensemble FID/IS and mean time FID/IS, are summarized in Table 2.

Short sequence generation. Since most existing action generative models were designed for short-term generation, we first examine the short sequence generative ability of them before diving into the analysis of long-duration action generation. The results are shown in the *Short* columns in Table 2. CSGN outperforms other baselines on both two datasets, even though it is not specifically tuned for short sequence generation.

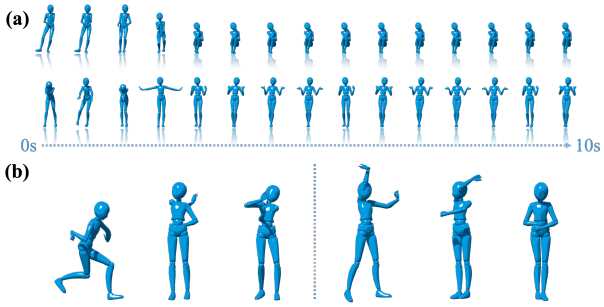

Figure 8. (a) Failed case in ERD. The top skeleton gradually freezes to a contemplation posture. The bottom skeleton waves its hands ceaselessly. (b) Unnatural pose in CSGN. Skeletons on the left twist their bodies in physically impossible ways. The right ones demonstrate arms piercing through heads and bodies.

Long-duration generation. The long-duration generation quality is indicated by the *Ensemble* and *Time* columns in Table 2. CSGN outperforms baselines on most benchmarks with a large margin. There are three notable observations. First, the gap between CSGN and two autoregressive models, *i.e.* ERD [9] and acLSTM [29], is significantly enlarged in the mean time FID/IS. This implies that similar contents may arise repeatedly in a sequence from the autoregressive models. As shown in Figure 8(a), ERD may lose creativity once it generate a static or reciprocating action. Second, the mean ensemble FID/IS of CSGN, the average of ensemble scores over time, is very close to short-term FID/IS scores. This suggests that CSGN can stably synthesize along the time dimension. Third, of all methods, the mean ensemble FID/IS scores are always better than mean temporal FID/IS. One possible explanation is the temporal dependency within a generated sequence. Snippets in a same sequence tend to have correlations rather than being independent, which disadvantages the within sequence diversity metric.

Note that although CSGN achieves remarkable improvements in generation quality in quantitative evaluation, it may sometimes generate unnatural poses as shown in Figure 8 (b). Similar cases happen in other GAN-based baselines.

6.3. Ablation Study

Effectiveness of Gaussian processes. To study the impact of the Gaussian process, we begin with two special cases. According to Equation (2), when the length scale $\sigma_c = 0$, the Gaussian process degrades to independent Gaussian noises, and there is no connection between two time positions in a latent sequence. In comparison, when $\sigma_c = \infty$, the latent vectors are constant over time, where temporal variations are being suppressed. Results in Table 3 show that these two

	NTU-RGB+D						MikuDance					
	FID			IS			FID			IS		
	Short	Ensemble	Time	Short	Ensemble	Time	Short	Ensemble	Time	Short	Ensemble	Time
ERD	46.24	41.60	94.90	5.11	5.48	1.18	54.37	47.49	352.1	15.70	**22.32**	1.06
acLSTM	69.86	78.41	82.00	3.41	3.24	3.08	46.31	50.96	229.5	15.28	16.69	3.09
HP-GAN	46.31	46.31	112.2	4.97	4.97	1.09	422.0	422.0	282.3	3.58	3.58	1.36
Two-Stage	87.91	—	—	1.06	—	—	420.4	—	—	2.42	—	—
CSGN	**6.03**	**5.86**	**8.80**	**15.40**	**15.39**	**12.71**	**22.60**	**23.49**	**34.74**	**20.71**	20.32	**18.78**

Table 2. Human action generation on MikuDance and NTU-RGB+D. We compare the FID (lower is better) and IS (higher is better) score of our CSGN with baseline models. The *Short* scores estimate the short-term generative ability; the *Ensemble* scores measure the quality and diversity between generated samples; while the *Time* scores evaluate the variance of short snippets within a single long sequence.

		NTU-RGB+D		MikuDance	
	Length Scale	FID^e	FID^t	FID^e	FID^t
Singular	0	12.60	13.35	1016.87	1013.23
	∞	74.65	102.64	652.48	756.85
Single	1	11.77	12.36	57.70	58.55
	10	14.64	16.16	31.97	32.45
	100	427.06	427.65	1552.46	1525.24
	1000	114.38	215.51	1546.10	1526.41
Spectrum	$0 \sim 1$	10.65	11.43	1017.67	997.78
	$0 \sim 10$	7.16	**8.04**	30.63	**31.96**
	$0 \sim 100$	**5.86**	8.80	**23.49**	34.74
	$0 \sim 1000$	6.19	18.73	26.00	73.89

Table 3. The FID scores (lower the better) on two datasets with different length-scale values and varied spectrum ranges.

special cases lead to poor generation qualities, especially on the MikuDance dataset, where the action sequences are much longer. Further, we fix the length scales σ_c to one single value between the two extreme cases, which is shown at the *Single* entry in Table 3. In NTU-RGB+D, small length scale ($\sigma_c = 1$) achieves smaller FID; while in MikuDance, larger length scale ($\sigma_c = 10$) performs better. If the value ($\sigma_c \geq 100$) is too large, the model fails to train. Finally, we test the spectrum of length scales, as mentioned in Section 3.1. The entry *Spectrum* in Table 3 compares different variation ranges of length scales. We observe that using a spectrum of Gaussian processes brings considerable improvements to the model. According to this study, we choose the varied length scale version with base scale $\bar{\sigma} = 100$ for other experiments.

Graph operations. In Table 4, we started with a baseline of 1D temporal convolution, which outputs a vector of size $3V$ representing the concatenated joint coordinates. This model performs fully-connected operations on the spatial dimension and ignores the body part connectivity between joints. We denote it as *Temp-Conv*. Then we add the graph convolution to the model by directly upsamples the latent vectors to the full resolution of skeleton graphs. This setting is denoted as *ST-GCN* and *ST-SGCN* according to whether they are using spatially varying graph convolution. Then we add the upsampling operators described in Section 3.2.

				NTU-RGB+D			MikuDance		
	GC	UP	SV	Param	FID^e	FID^t	Param	FID^e	FID^t
Temp-Conv				27.2M	22.25	23.60	27.3M	43.19	50.30
ST-GCN	✓			20.1M	8.72	11.73	36.6M	35.83	42.98
ST-SGCN	✓		✓	26.2M	101.2	102.1	48.2M	1194	1192
CSGN-GC	✓	✓		9.4M	31.51	33.71	9.4M	31.20	38.21
CSGN	✓	✓	✓	14.8M	5.86	8.80	17.3M	**23.49**	**34.74**
CSGN-Narrow	✓	✓	✓	**5.7M**	**5.13**	**7.75**	**6.3M**	26.58	36.89

Table 4. Design decisions in graph modeling. Graph convolution/upsampling is denoted as *GC*/*UP*, and spatially varying graph convolution is denoted as *SV*. The number of parameters is shown in column *Param*. *CSGN-Narrow* halves the channels of *CSGN*.

From the results we can observe that the graph convolution and gradual upsampling is important for generating high quality samples. Furthermore, the spatially varying graph convolution is powerful only when used together with the gradual upsampling, leading to the best generation quality. We also halve the channels of CSGN. With less parameters, CSGN still outperforms other baselines.

7. Conclusion

In this work, we propose a convolution based approach to tackle the problem of human action synthesis. The model directly transforms a sequence of noises from a Gaussian process to a data sequence that follows some unknown data random process. Gaussian processes enforce long-term temporal correlations and spatial-temporal graph convolutions construct fine motions. It overcomes the major limitations of previous models. To evaluate the model, we design two quantitative metrics for the sequence generation task and provide a new dataset with 10 hours of dance animation data. On the two large scale skeleton-based human action datasets, we observe a remarkable improvement of generation quality in both quantitative and qualitative evaluations.

Acknowledgement This work is supported by the Collaborative Research Grant from SenseTime (CUHK Agreement No. TS1610626 & No. TS1712093), and the General Research Fund (No. 14209217 & No. 14203518).

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Emad Barsoum, John Kender, and Zicheng Liu. HP-GAN: Probabilistic 3d human motion prediction via GAN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1418–1427, 2018.

[3] Ali Borji. Pros and cons of GAN evaluation measures. *Computer Vision and Image Understanding*, 179:41–65, 2019.

[4] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.

[5] Judith Bütepage, Michael J Black, Danica Kragic, and Hedvig Kjellström. Deep representation learning for human motion prediction and classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 2017. IEEE, 2017.

[6] Haoye Cai, Chunyan Bai, Yu-Wing Tai, and Chi-Keung Tang. Deep video generation, prediction and completion of human action sequences. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 366–382, 2018.

[7] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. *arXiv preprint arXiv:1808.07371*, 2018.

[8] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[9] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[11] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[12] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[13] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] Hanbyul Joo, Tomas Simon, Xulong Li, Hao Liu, Lei Tan, Lin Gui, Sean Banerjee, Timothy Scott Godisart, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social interaction capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[16] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[17] Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4674–4683. IEEE, 2017.

[18] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023, 2016.

[19] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[20] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[21] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.

[22] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1010–1019, 2016.

[23] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[24] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[25] The Motion Capture Lab. Cmu graphics lab motion capture database. *http://mocap.cs.cmu.edu/*, 2018.

[26] Wikipedia. MikuMikuDance. *https://en.wikipedia.org/w/index.php?title=MikuMikuDance*, 2018.

[27] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*, 2018.

[28] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE MultiMedia*, 19(2):4–10, Apr. 2012.

[29] Yi Zhou, Zimo Li, Shuangjiu Xiao, Chong He, Zeng Huang, and Hao Li. Auto-conditioned recurrent networks for extended complex human motion synthesis. In *International Conference on Learning Representations*, 2018.