ABSTRACT

A deep recurrent neural network with audio input is applied to model basic dance steps. The proposed model employs multilayered Long Short-Term Memory (LSTM) layers and convolutional layers to process the audio power spectrum. Then, another deep LSTM layer decodes the target dance sequence. This end-to-end approach has an auto-conditioned decode configuration that reduces accumulation of feedback error. Experimental results demonstrate that, after training using a small dataset, the model generates basic dance steps with low cross entropy and maintains a motion beat F-measure score similar to that of a baseline dancer. In addition, we investigate the use of a contrastive cost function for music-motion regulation. This cost function targets motion direction and maps similarities between music frames. Experimental result demonstrate that the cost function improves the motion beat f-score.

Index Terms—Deep recurrent networks, Contrastive loss, Dance generation

1. INTRODUCTION

Methods to generate human motion are being actively investigated in various domains. Some studies have developed applications that go beyond simply generating dance motion for robots [1, 2], animated computer graphics animated choreographies [3], and video games. Music content-driven motion generation (i.e., generating dance steps for a given piece of music) involves motion as a time-series [4] and non-linear time-dependent mapping between music and motion [5, 6, 7].

Due to its physical nature, dance can be represented as high-dimensional nonlinear time-series [4]. To address this high dimensionality, a factored conditional restricted Boltzmann machine and recurrent neural network (RNN) [8] have been proposed to map audio and motion features and generate a new dance sequence. A generative model [9] has also been implemented to generate a new dance sequence for a solo dancer. However, dance generation requires significant computational capabilities or large datasets. In addition, dance generation is constrained by the trained data. Dancing involves significant changes in motion that occur at regular intervals, i.e., a motion beat [5, 6, 7]. When dancing to music, the music and motion beat should be synchronized. In previous studies [9, 8], generating motion required detailed information about the music.

We propose a deep learning model to generate basic dance steps synchronized to the music’s rhythm. The proposed end-to-end model generates large motion sequences with precision similar to that of a human dancer. Following a previous study [10], the proposed model employs multilayered Long Short-Term Memory (LSTM) layers and convolutional layers to encode the audio power spectrum. The convolutional layers reduce the frequency variation of the input audio, and the LSTM layers model time sequence features. We employ another deep LSTM layer with an auto-conditioned configuration [11] to decode the motion. This configuration enables the model to handle a longer dance sequence with low noise accumulation, which is fed back into the network. In addition, we use a contrastive cost function [12] for music-motion regulation to ensure alignment between the motion and the music beat. The contrastive cost function is a measure of similarity between the given inputs, it minimizes the distance of the input patterns in case that the inputs were similar; otherwise, the distance is maximized. This cost function enables training with small number of samples and avoids the need of pre-training, therefore, reduces the need of larger computational capabilities. The cost function uses motion direction as a target and maps similarities between its inputs. In this study, the inputs are audio features from the encoder. This increases the precision of the motion beat with regard to the music beat and avoids the use of additional label information or annotations from the music. The proposed model demonstrates improved music-motion regulation.

The primary contributions of this study are summarized as follows:

• We use a deep RNN (DRNN) and a contrastive cost function to generate long motion sequences. The contrastive cost function improves the alignment between the music and motion beat, is end-to-end trainable, and reduces the need for additional annotations or labeled data (Section 2). We describe the training setup and feature extraction in Section 3.4.

• We evaluate the motion beat and the cross entropy of the generated dance relative to the trained music (Section 4). We demonstrate that the proposed approach increases the precision of the motion beat along with the music beat and models basic dance steps with lower cross entropy.

• Conclusions and suggestions for potential future enhancements of the proposed model are given in (Section 5).

2. PROPOSED FRAMEWORK

An overview of the proposed system is shown Figure 1.

2.1. Deep Recurrent Neural Network

Mapping high-dimensional sequences, such as motion, is a challenging task for deep neural networks (DNN) [3] because such sequences are not constrained to a fixed size. In addition, to generate motion from music, the proposed model must map highly non-linear representations between music and motion [8]. In time signal modeling [13], DRNNs implemented with LSTM layers have shown remarkable performance and stable training when deeper networks...
are employed. Furthermore, using stacked convolutional layers in a DRNN has demonstrated promising results for speech recognition tasks [10]. This unified framework is referred to as a CLDNN. In this framework, the convolutional layers reduce the spectral variation of the input sound, and the LSTM layers perform time signal modeling. To construct the proposed model, we consider a DRNN with LSTM layers separated into two blocks [13]: one to reduce the music input sequence (encoder) and another for the motion output sequence (decoder). This configuration can handle non-fixed dimensional signals, such as motion, and avoids performance degradation due to the long-term dependency of RNNs.

The input to the network is the power spectrum from the audio represented as $x_{1:n} = (x_t \in \mathbb{R}^b | t = 1, \ldots, n)$ with $n$ frames and $b$ frequency bins, and the ground truth sequence is represented as $y_{1:n} = (y(t) \in \mathbb{R}^j | t = 1, \ldots, n)$ with $n$ frames and $j$ joint axes. The following equations show the relations of the motion modeling:

$$g(x_t) = \text{LSTM}^{le}(x'_t), \quad (1)$$

$$m_{t+1} = \text{LSTM}^{ld}(g(x_t)), \quad (2)$$

where $g(x_t)$ is the output processed by the encoder with $le$ layers, and $x'_t$ is the output from the convolutional layers. Network output $m_{t+1}$ is processed from the current input and the previous states of the decoder with $ld$ layers (Fig. 2 left). Then, $m_{t+1}$ is a l2-norm model.

However, with time-series data (such as dance data), the model may freeze or the output may diverge from the target due to accumulated feedback errors. To address these issues, the output of the decoder sets its value to consider autoregressive noise accumulation by including the previous generated step in the motion generation process.

2.2. Auto-conditioned Decoder

A conventional method uses as input the ground truth of the given sequence as the input to train sequences with RNN models. During evaluations, the model that was accustomed to the ground truth in the training process may freeze or diverge from the target due to the accumulation of slight differences between the trained and the self-generated sequence.

The auto-conditioned LSTM layer handles errors accumulated during sequence generation by conditioning the network using its own output during training. Thus, the network can handle large sequences from a single input, maintain accuracy, and mitigate error accumulation.

In a previous study [11] that employed an auto-conditioned LSTM layer for complex motion synthesis, the conditioned LSTM layer was trained by shifting the input from the generated motion with the ground truth motion after fixed repetitive steps. In the proposed method, we only employ ground truth motion at the beginning of the training sequence as a target (Fig. 2). By modifying Eq. 2, the generated output is expressed as follows.

$$m_{t+1} = \text{LSTM}^{ld}([g(x_t), y'_t]) \quad (3)$$

where

$$y'_t = \begin{cases} y_t & \text{if } t = 0 \\ m_t & \text{otherwise.} \end{cases} \quad (4)$$

And the error of the motion is calculated by using a mean squared error (MSE) cost function, which is be denoted as:

$$L_{\text{MSE}} = \frac{1}{k} \sum_{i=1}^{k} (y_{t+1} - m_{t+1})^2 \quad (5)$$

where $k$ is the training batch size, $y(t + 1)$ is the ground truth and $m(t + 1)$ is the generated motion as.

We employ a vector of zeros in our evaluations as the input of the first step followed by the self-generated output to generate the dance until the music stops.

2.3. Music-motion Alignment Regulation

The motion beat is defined as significant changes in movement at regular moments, and a previous study [6] reported that motion-beat frames occur when the direction of the movement changes; thus, the
motion beat occurs when the speed drops to zero. Furthermore, harmony is a fundamental criterion when dancing to music. Therefore, the music and motion beat should be synchronized.

For basic dance steps, the repetitions of dance steps are given by a repetitive music beat, where the direction of the movement changes drastically (Fig. 3). To avoid using additional information, we employed the previous definition to formalize the extracted music features will be different compared to the previous frame (i.e., \( g(x_1) \neq g(x_{t-1}) \)) when a beat occurs; otherwise, it may keep a similar dimension (i.e., \( g(x_1) \approx g(x_{t-1}) \)).

For regulation, we employ a contrastive cost function \([12]\) that can map a similarity metric for the given features. To employ contrastive loss, we extract the standard deviation (SD) of the ground truth motion at each frame and compare it to the next frame. Then, we assign a label equal to 1 when the motion maintains its direction; otherwise, the label is 0.

At \( t \):

\[
\begin{align*}
    d_t &= SD(y_t) \\
    s_t &= \text{sign}(d_t - d_{t-1}),
\end{align*}
\]

and at \( t + 1 \):

\[
    s_{t+1} = \text{sign}(d_{t+1} - d_t).
\]

Then, labels \( d \) are expressed as follows:

\[
    d = \begin{cases} 
    1 & \text{if } s_{t+1} = s_t, \\
    0 & \text{otherwise}
    \end{cases}
\]

The contrastive cost function at frame \( t + 1 \) is expressed as:

\[
    \mathcal{L}_{\text{contrastive}} = \frac{1}{2}(d_{t+1}g^2 + (1 - d_{t+1}) \max(1 - g, 0)^2)
\]

where \( g = \|g(x_{t+1}) - g(x_t)\|^2 \).

Finally, the cost function of the model is formulated as follows:

\[
    \mathcal{L}_y = \mathcal{L}_{\text{MSE}}[y_{t+1}, m_{t+1}] + \max(\mathcal{L}_{\text{contrastive}}[g_{t+1}, g_t], 0).
\]

In this manner, we synchronize the music beat to the motion beat without requiring additional annotations or further information for the music beat. Figure 4 shows how the features from the output encoder behave after being trained by contrastive loss. The principal component analysis (PCA) features of the encoder output have a repetitive pattern, and, in our experiment, they move in an elliptical shape and group the music beats at specific areas.

2.4. Model Description

The DRNN topology employed in our experiments comprised a CLDNN encoder and a deep recurrent decoder (Fig. 1). The CLDNN architecture follows a similar configuration as that reported in the literature [10].

The input audio features are reduced by four convolutional layers, each followed by batch normalization and exponential linear unit activation. Then, three LSTM layers with 500 units each and a fully-connected layer with 65 dimensions complete the encoder structure.

The input to the decoder consists of the previous motion frame (71 dimensions) and the output of the decoder (65 dimensions) with a width of 136 dimensions. The decoder also comprises three LSTM layers with 500 units each and a fully-connected layer with 71 dimensions.

For music-motion control, we add the contrastive cost function after calculating the next step and the mean squared error cost function.

3. EXPERIMENTS

In this study, our experiments were conducted to 1) improve motion generation with weakly supervised learning and 2) examine the effect that music with different characteristics has on the results.

3.1. Data

We prepared three datasets due to a lack of available datasets that include dance synchronized to music. We restricted the data to small samples using different music genres with different rhythms.

**Hip hop bounce**: This dataset comprises two hip hop music tracks with a repetitive lateral bouncing step to match the rhythm. Each track is three minutes long on average at 80 to 95 beats per second (bpm).

**Salsa**: This dataset comprises seven salsa tracks (four minutes long on average). All tracks include vocals and rhythms between 95 to 130 bpm. This dataset includes a lateral salsa dance step during instrumental moments and a front-back salsa step during vocal elements.

**Mixed**: This dataset comprises 13 music tracks with and without vocal elements (six genres: salsa, bachata, ballad, hip hop, rock and bossa nova) with an average length of three minutes. Depending on the genre, each track includes up to two dance steps.

3.2. Audio Feature Extraction

Each audio file was sampled at 16 KHz. Then, we extracted the power features as follows.

- To synchronize the audio with the motion frame, we extracted a slice of 534 samples (33 ms) of the corresponding position. This extracted slice was converted to a short-time Fourier transform (STFT) frames of 160 samples (10 ms) with a shift of 80 samples (5 ms).
- From the STFT frames, we used the power information, which was normalized between -0.9 and 0.9 on the W frequency bin axis.
- We stacked the H frames; thus, the input of the network was a \( 1 \times W \times H \)-dimensional file.

3.3. Motion Representation

For each audio track, we employed the rotations and root translation captured by a single Kinect v2 device at a regular rate of 30 frames per second. Then, the captured motion was post-processed and synchronized with the audio data using a previously proposed motion beat algorithm [6].

- From the captured data in hierarchical translation-rotation format, we processed the spatial information (i.e., translation) of the body in a vector \((x, y, z)\) in meters and a 17-rotation vector in quaternions denoted \((q_x, q_y, q_z, q_w)\).
- We normalized each vector component using the maximum value of each component in the range of \(-0.9\) to 0.9. The resultant vector (71 dimension) was the target of the neural network.

Note that we did not apply any filtering or denoising method to maintain the noisy nature of the motion.
Table 1. F-score and cross entropy of bounce dataset

| Method                  | Hip Hop Clean* | White* noise | Claps | Crowd | Other genres | Entropy |
|-------------------------|---------------|--------------|-------|-------|-------------|---------|
| Madmom (Music beat)     | 89.18         | -            | -     | -     | -           | 80.13   |
| Marsyas (Music beat)    | 54.11         | -            | -     | -     | -           | 48.89   |
| Dancer (baseline)       | 62.31         | -            | -     | -     | -           | 54.49   |
| S2S                     | 55.98         | 46.33        | 50.56 | 54.11 | 32.95       | 35.84   |
| S2S-MC                  | 64.90         | 55.58        | 60.62 | 56.37 | 37.69       | 34.63   |

Table 2. F-score of salsa dataset

| Method                  | Clean* | White* | Claps | Crowd |
|-------------------------|--------|--------|-------|-------|
| Madmom                  | 51.62  | -      | -     | -     |
| Marsyas                 | 23.38  | -      | -     | -     |
| Dancer (baseline)       | 52.82  | -      | -     | -     |
| S2S                     | 53.79  | 52.88  | 52.76 | 51.98 |
| S2S-MC                  | 53.96  | 53.09  | 53.61 | 52.48 |

Table 3. F-score of mixed genres

| Method                  | Bachata* | Ballad* | Bossa* | Nova | Rock* | Hip Hop* | Salsa* |
|-------------------------|----------|---------|--------|------|-------|----------|--------|
| Dancer (baseline)       | 62.35    | 52.07   | 45.02  | 62.72| 55.84 | 53.86    |
| S2S                     | 66.72    | 49.92   | 46.19  | 60.06| 64.30 | 52.31    |
| S2S-MC                  | 56.63    | 48.48   | 40.91  | 64.87| 63.85 | 53.71    |

3.4. Training procedure

The models were trained for 15 epochs using each dataset and the CHAINER framework [14] as an optimization tool. Each training epoch took an average of 60 minutes.

The models were trained using an NVIDIA GTX TITAN graphic processing unit. For the optimization, we employed the ADAM solver [15] with a training mini-batch of 50 files and white noise added to the gradient. Each training batch employed sequences of 150 steps.

4. RESULTS

4.1. Metrics

A quantitative evaluation of the models was performed using the f-score. We followed the literature [6] to extract the motion beat of the generated dances. As can be seen, the proposed models demonstrate better performance than MARSYAS, and S2SMC outperformed S2S in the evaluations that used clean and noisy data for training. However, both models did not perform well when a different music track or music from different genres were used as the input. In addition, the proposed models did not outperform a model trained to only process the music beat (i.e., MADMOM). S2SMC demonstrated lower cross entropy than S2S, which means that the dance generated by S2SMC was similar to the trained dance.

Table 2 shows the f-score for the music tracks trained using the salsa dataset. Both models show better performance than the dancer when tested under the same training conditions, and S2SMC shows better performance than S2S under all conditions. Note that the size of the dataset influences the performance of the models, and we employed a larger dataset compared to the previous experiment.

Table 3 shows the results of the mixed genre dataset. As can be seen, diverse results were obtained for each model. The proposed methods cannot outperform the baseline, whereas S2S outperformed S2SMC for most genres. The main reason for this difference in the results is due to the complexity of the dataset and the variety of dance steps relative to the number of music samples; thus, the model could not group the beat correctly (Fig. 5).

5. CONCLUSION

In this paper, we have presented an optimization technique for weakly supervised deep recurrent neural networks for dance generation tasks. The proposed model was trained end-to-end and
performed better than using only a mean squared cost function. We have demonstrated that the models can generate correlated motion patterns with a similar motion-beat f-score to that of a dancer and lower cross entropy. In addition, due to low forwarding time (approximately 12 ms), the models could be used for real-time tasks. The models show low training time and can be trained from scratch.

The proposed model demonstrates reliable performance for motion generation. However, the motion pattern is affected by the diversity of the trained patterns and is constrained to the given dataset. This issue will be the focus of future experiments.

6. REFERENCES

[1] Joao Lobato Oliveira, Gokhan Ince, Keisuke Nakamura, Kazuhiro Nakadai, Hiroshi G Okuno, Fabien Gouyon, and Luis Paulo Reis, “Beat tracking for interactive dancing robots,” International Journal of Humanoid Robotics, vol. 12, no. 04, pp. 1550023, 2015.

[2] J. J. Aucouturier, K. Ikeuchi, H. Hirukawa, S. Nakaoka, T. Shiratori, S. Kudoh, F. Kanehiro, T. Ogata, H. Kozima, H. G. Okuno, M. P. Michalowski, Y. Ogai, T. Ikegami, K. Kosuge, T. Takeda, and Y. Hirata, “Cheek to chip: Dancing robots and ai’s future,” IEEE Intelligent Systems, vol. 23, no. 2, pp. 74–84, March 2008.

[3] Satoru Fukayama and Masataka Goto, “Music Content Driven Automated Choreography with Beat-wise Motion Connectivity Constraints,” in Proceedings of SMC, 2015.

[4] Zhe Gan, Chunyuan Li, Ricardo Henao, David Carlson, and Lawrence Carin, “Deep Temporal Sigmoid Belief Networks for Sequence Modeling,” Advances in Neural Information Processing Systems, pp. 1–9, 2015.

[5] Tae-hoon Kim, Sang Il Park, and Sung Yong Shin, “Rhythmic-motion synthesis based on motion-beat analysis,” in ACM SIGGRAPH 2003 Papers, New York, NY, USA, 2003, SIGGRAPH '03, pp. 392–401, ACM.

[6] C. Ho, W. T. Tsai, K. S. Lin, and H. H. Chen, “Extraction and alignment evaluation of motion beats for street dance,” in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, May 2013, pp. 2429–2433.

[7] W. T. Chu and S. Y. Tsai, “Rhythm of motion extraction and rhythm-based cross-media alignment for dance videos,” IEEE Transactions on Multimedia, vol. 14, no. 1, pp. 129–141, Feb 2012.

[8] Omid Alemi, Jules François, and Philippe Pasquier, “GrooveNet: Real-Time Music-Driven Dance Movement Generation using Artificial Neural Networks *,” vol. 6, no. August, 2017.

[9] Luka Crnkovic-Friis and Louise Crnkovic-Friis, “Generative choreography using deep learning,” CoRR, vol. abs/1605.06921, 2016.

[10] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, “Convolutional, long short-term memory, fully connected deep neural networks,” in 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), April 2015, pp. 4580–4584.

[11] Zimo Li, Yi Zhou, Shuangjiu Xiao, Chong He, and Hao Li, “Auto-conditioned LSTM network for extended complex human motion synthesis,” CoRR, vol. abs/1707.05363, 2017.