A new dance generation task from start to end

Xiao Xu\textsuperscript{1,2,3} and Lijuan Duan\textsuperscript{1,2,3*}

\textsuperscript{1}Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
\textsuperscript{2}Beijing Key Laboratory of Trusted Computing, Beijing 100124, China
\textsuperscript{3}National Engineering Laboratory for Critical Technologies of Information Security Classified Protection, Beijing 100124, China

* Corresponding author’s e-mail: ljduan@bjut.edu.cn

Abstract. The dance generation task is an interesting task. The general task is to predict the next dance sequence through a given initial frame, or to convert from music to dance through audio information. However, these works cannot control the direction of dance generation and lack suitable ground truth to evaluate the generated results. In our work, we give the start frame and the end frame, which can effectively solve the above problems, and can also be understood as a standardized new task. Our experimental results show that through a two-stream CNN network, we can well complete the new task that we proposed.

1. Introduction

Dance is an art of human movement. It takes professional movements as the main means of expression, and focuses on the inner spiritual world of people which is difficult to express in words or other artistic expressions. Generally, it has delicate emotions, deep thoughts or clear relationships while the movements must be refined, organized, and beautiful. As an ordinary person, dance choreography is a time-consuming and laborious job because it requires professional background and economic foundation. Therefore, how to use the deep learning method to realize the automatic generation of dance through the inherent connection of dance movements becomes very meaningful.

The previous dance generation work is mainly to control the quality of generated movements through learning models or statistical distribution of parameters. For example, some works use Hidden Markov Models to complete the synthesis of dance movements \cite{1, 2}; some works use Linear Dynamic System (LDS) to learn and produce dance movements \cite{3}; the learning content represented as step charts that encode the timing and position of steps and use LSTMs to generate a new step chart \cite{4}; using Long Short Term Memory (LSTM) and Recurrent Neural Networks (RNNs) to learn and generate choreography \cite{5}. On the basis of these works, some subsequent works introduce deep learning and audio information to form a mapping of music to dance task. Some works align the audio with the skeleton information, using LSTM and FC layers to construct the generating network \cite{6}; taking advantage of the low parameters of the convolution layer to construct multi-frame dance prediction by music \cite{7, 8}, and so on.

The current dance generation task needs to consider the diversity of dance generation. However, because of the uncertainty of the prediction results, it is difficult to evaluate the stability of the generated results and whether it conforms to the normal dance style in detail. In our work, we give a start frame and an end frame, so that based on the prompts of the two, there can be a specific direction...
of generation, and the dance generation task can be completed under the standardized requirements. Of course, with the above two frames, we combine the middle frames with them to form the ground truth (GT) required for evaluation. It should be noted that in a short period of time, there can be a corresponding dance GT. But if the duration is long, measuring the relationship between generation results and GT will limit the consideration of the diversity of dance, which will become meaningless.

We are inspired by some works of Super Resolution [9,10,15]. We fill the input two frames to the output size, and the new input can be regarded as a low-resolution input in Super Resolution. The actual dance information between the two frames is the target output, which is GT. Then CNN is used to learn the changes of intermediate movements, and finally we can get a dance segment of the required time length.

The main contributions of our work are as follow:

- A new dance generation task is proposed, this task is more directional which can better understand related tasks and evaluate experimental results.
- Based on the two-stream CNN dance generation model, our results are well reflected in our evaluation metric tMPJPE.

2. Methodology

An overview of the proposed model is shown in Figure 1.

![Figure 1. Two-stream CNN dance generation model.](image)

2.1. Data processing

Our input is the start frame and the end frame, the size of the data in each frame is 23×3. We turn it to 69 dimensions (skeleton size). Then initialize and supplement in the middle of the two frames, the input of one stream is filled with the start frame in the middle, and the other stream is filled with the end frame. After that we can get two new inputs, and their sizes are both skeleton size×time, time here can be understood as the number of frames that need to be supplemented in between.

2.2. Feature learning based on CNN

Our dance generation model consists of twelve convolutional layers. The number of kernels of the first layer and the last two layers are 32, 32, 1 while each other layer is 64. The kernel size and the stride for each convolutional layer is 3 and 1. Except for the last layer, each other convolutional layer follows by a batch normalization layer (BN) and a ReLU layer. After ten layers of convolution, we concentrate the two outputs and input the merged result to the subsequent network. Formally, each layer is expressed as an operation Fi:
\[ F_i(X) = \max \left( 0, BN_{\gamma_i, \beta_i}(W_i \ast X + B_i) \right) \] (1)

Where \( X \) refers the input, \( i \) refers \( i \)-th layer, \( i \in \{1, 2, \ldots, 11\} \); \( \max(0, \cdot) \) refers function \( \text{ReLU} \); \( \gamma \) and \( \beta \) are the parameters to be learned by BN; \( W \ast X + B \) refers convolution operation, \( W \) and \( B \) represent the filters and biases respectively.

It should be noted that share weight is performed on two streams. On the one hand, we consider that the number of parameters is reduced. On the other hand, the learning of the two inputs is not much different, so share weight can satisfy the joint learning of the two streams.

The reason why we do not use LSTM [14] in feature learning is that, first, LSTM has much more parameters than CNN to achieve the same output dimension. Second, CNN cannot limit the size of the input, which is convenient for us to obtain results of different durations, while LSTM fixes the input and is only suitable for a fixed duration.

2.3. Loss function
This is a typical regression task, we use Mean Squared Error (MSE) as the loss function:

\[ \text{Loss} = \frac{1}{n} \sum_{i=1}^{n} \| F(X_i) - Y_i \|^2 \] (2)

This is achieved through minimizing the loss between the output \( F(X) \) and the corresponding GT \( Y \), where \( n \) is the number of training samples.

2.4. Evaluation metric
We consider using MPJPE (per joint position error, [11]) as the evaluation metric of our experiments which is usually used in the 3D Pose Estimation task. It mainly calculates the Euclidean distance between ground truth and prediction first, and then gets mean of per joint position error for all joints. However, our task has time dimension information, so adding the calculation of time on the basis of MPJPE, our metric expression becomes:

\[ t\text{MPJPE} = \frac{d(w, v)}{k \times t} \] (3)

Among them, \( w \) is the output, \( v \) is the target, \( d(\cdot, \cdot) \) is the Euclidean distance, \( k \) is the number of joint points, and \( t \) is the time parameter. Our metric can be understood as MPJPE per unit time.

It should be noted that our metric focuses on the accuracy of the joint position and is only suitable for short durations. If it takes a long time, the result may be different from GT, but the movements are in line with the dance requirements, so that the evaluation will be inaccurate. Therefore, do not use our metric to evaluate long dance sequences.

3. Experiments

3.1. Dataset and implementation details
We use the datasets in the paper [12]. The datasets contain four types of dance (Cha-Cha, Rumba, Tango and Waltz), total approximately 907,000 frames with 25 fps, all of which are 3D skeleton information with 23 key points per frame. We choose the Cha-Cha, Rumba and Tango dance as the data of our experiments, with about 19500, 27300, 42000 frames, respectively.

The number of training sets is 17000×3, and the test sets has two scenes of 1s and 3s, so the number is 1000×3×2. The training samples are unified into 1s samples with size of 25×69. For \( t\text{MPJPE}, k = 23 \), the value of \( t \) is equal to the total frames divided by fps, and the unit is s.
We train models for each type of dance. For the optimization, we all employ the Adam [13] solver with a batch of 64, the learning rate in our experiments is 0.0001 and stop our training procedure at 1000 epochs for each dance.

3.2. Visualization of generated dance
In the actual generation, we only need to give the start frame and the end frame, fill in the middle frames according to the required time to construct the network input, and after the network processing, we can get the final generation result.

As shown in Figure 2, we test on different dance test sets to generate 1s dance sequences. Due to limited layout space, we select some of them for display and show the 3D skeleton results every 4 frames.

Based on the results, our model design can meet the needs of our task, similar to GT. For the value of tMPJPE, the smaller the value, the closer to the GT, and there is no deformation of the movements. However, for large values, some movements are deformed, and lack the consistency and fluency of the original movements, which is quite different from the GT.

| Sample1: Cha-cha | output |
|------------------|--------|
| tMPJPE: 8.693    |        |
| GT               |        |

| Sample2: Rumba  | output |
|-----------------|--------|
| tMPJPE: 5.691   |        |
| GT               |        |

| Sample3: Tango  | output |
|-----------------|--------|
| tMPJPE: 12.973  |        |
| GT               |        |

Figure 2. Some of generation results about output and GT.
3.3. Quantitative results

We compared different model design methods under similar parameters as follows:

- **M0**: Only use one-stream CNN network to learn the features, the others are unchanged.
- **M1**: Baseline which we proposed in Methodology.
- **M2**: Use the 5 residual blocks to replace the 10 convolutional layers in the middle of our baseline. The residual block consists of two convolution layers with the same kernel size and number of filters, its output is the sum of the input and the result after two convolution layers. The others are unchanged.

| Dataset  | Model | tMPJPE (frames=25) | tMPJPE (frames=75) |
|----------|-------|---------------------|---------------------|
| Cha-Cha  | M0    | 12.767              | 18.901              |
|          | M1    | **12.069**          | **17.792**          |
|          | M2    | 12.296              | 17.875              |
| Rumba    | M0    | 8.957               | 17.417              |
|          | M1    | **8.106**           | **15.869**          |
|          | M2    | 8.429               | 15.935              |
| Tango    | M0    | 8.798               | 16.738              |
|          | M1    | **8.061**           | **15.228**          |
|          | M2    | 8.283               | 15.255              |

From Table 1, our baseline reflects the best in the evaluation results. In different types of dance, the impact of the model designs is the same: M1 is the best, followed by M2, the worst is M0. In addition, the result of 3s test sets is not as good as 1s. This is because the longer the time, the more difficult it is to control the direction of generation, which is consistent with our previous conjecture.

Compared with one-stream, two-stream obtains better results, mainly because its input to the network becomes more, and the two processing streams learn features from different generation directions. Such a network takes into account the same importance of input, so it can achieve better results. We also use residual learning that can improve performance on general tasks, but our experimental results are not good compared to ordinary convolutional layers. We consider the reason is that the change of each frame is small, and the residual result becomes sensitive to data fluctuations.

Our experiments show that we can use the baseline to generate dance sequences of any length. If need to consider the quality of the generation, we can use the short duration GT and reference to evaluate whether the generated results are accurate. The "accurate" here focuses on measuring whether it conforms to the original dance's movement style and quality.

4. Conclusion

In this paper, we explored a new dance generation task based on the start and end frames, this task can help us control the direction of the generation, and can have GT to evaluate the generated results. On this basis, we have proposed a two-stream CNN network to solve the new task for us, and our experimental results have also performed well on the data sets in our metric. In the future, we will focus on solving the problem of dance movements which are not smooth, in order to get more realistic generation results.

Acknowledgments

This work was supported in part by the Key Project of Beijing Municipal Education Commission (no.KZ201910005008).
References
[1] Matthew Brand and Aaron Hertzmann. 2000. Style machines. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques. 183–192.
[2] F. Ofli, E. Erzin, Y. Yemez, and A. M. Tekalp. 2012. Learn2Dance: Learning Statistical Music-to-Dance Mappings for Choreography Synthesis. IEEE Transactions on Multimedia 14, 3 (June 2012), 747–759.
[3] Yan Li, Tianshu Wang, and Heung-Yeung Shum. 2002. Motion texture: a two-level statistical model for character motion synthesis. In Proceedings of the 29th annual conference on Computer graphics and interactive techniques. ACM, 465–472.
[4] Chris Donahue, Zachary C Lipton, and Julian McAuley. 2017. Dance Dance Convolution. (March 2017). arXiv:1703.06891
[5] Luka Crnkovic-Friis and Louise Crnkovic-Friis. 2016. Generative Choreography using Deep Learning. CoRR abs/1605.06921 (May 2016).
[6] N. Yalta, S. Watanabe, K. Nakadai, and T. Ogata, “Weakly-supervised deep recurrent neural networks for basic dance step generation,” in 2019 International Joint Conference on Neural Networks (IJCNN). IEEE,2019, pp. 1–8.
[7] J. Lee, S. Kim, and K. Lee, “Listen to dance: Music-driven choreography generation using autoregressive encoder-decoder network,” arXiv preprint arXiv:1811.00818, 2018.
[8] H. Ahn, J. Kim, K. Kim, and S. Oh, “Generative autoregressive networks for 3d dancing move synthesis from music,” arXiv preprintarXiv:1911.04069, 2019.
[9] C. Dong, C. C. Loy, K. He. and X. Tang. Image super-resolution using deep convolutional networks, IEEE Transactions on Pattern Analysis and Machine Intelligence. 2015, 38(2):295–307.
[10] J. Kim. J. K. Lee. and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In IEEE Conference on Computer Vision and Pattern Recognition. 2016.
[11] Zhou X, Huang Q, Sun X, et al. Towards 3D Human Pose Estimation in the Wild: a Weakly-supervised Approach[J]. 2017.
[12] T. Tang, J. Jia, and H. Mao, “Dance with melody: An lstm-autoencoder approach to music-oriented dance synthesis,” in 2018 ACM Multimedia Conference on Multimedia Conference. ACM, 2018, pp. 1598–1606.
[13] Kingma D P, Ba J. Adam: A Method for Stochastic Optimization[J]. arXiv: Learning, 2014.
[14] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.
[15] Jeon D S, Baek S H, Choi I, et al. Enhancing the spatial resolution of stereo images using a parallax prior[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 1721-1730.