Liu F, Chen DL, Zhou RZ *et al.* Self-supervised music motion synchronization learning for music-driven conducting motion generation. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 37(3): 539–558 May 2022. DOI 10.1007/s11390-022-2030-z

Self-Supervised Music Motion Synchronization Learning for Music-Driven Conducting Motion Generation

Fan Liu[1] (刘 凡), *Member*, *CCF*, *IEEE*, De-Long Chen[1,*] (陈德龙), Rui-Zhi Zhou[1] (周睿志)
Sai Yang[2] (杨 赛), and Feng Xu[1] (许 峰), *Member*, *CCF*

[1] *College of Computer and Information, Hohai University, Nanjing 211100, China*
[2] *School of Electrical Engineering, Nantong University, Nantong 226019, China*

E-mail: {fanliu, chendelong, zhouruizhi}@hhu.edu.cn; yangsai@ntu.edu.cn; xufeng@hhu.edu.cn

Received November 19, 2021; accepted March 10, 2022.

Abstract The correlation between music and human motion has attracted widespread research attention. Although recent studies have successfully generated motion for singers, dancers, and musicians, few have explored motion generation for orchestral conductors. The generation of music-driven conducting motion should consider not only the basic music beats, but also mid-level music structures, high-level music semantic expressions, and hints for different parts of orchestras (strings, woodwind, etc.). However, most existing conducting motion generation methods rely heavily on human-designed rules, which significantly limits the quality of generated motion. Therefore, we propose a novel Music Motion Synchronized Generative Adversarial Network (M^2S-GAN), which generates motions according to the automatically learned music representations. More specifically, M^2S-GAN is a cross-modal generative network comprising four components: 1) a music encoder that encodes the music signal; 2) a generator that generates conducting motion from the music codes; 3) a motion encoder that encodes the motion; 4) a discriminator that differentiates the real and generated motions. These four components respectively imitate four key aspects of human conductors: understanding music, interpreting music, precision and elegance. The music and motion encoders are first jointly trained by a self-supervised contrastive loss, and can thus help to facilitate the music motion synchronization during the following adversarial learning process. To verify the effectiveness of our method, we construct a large-scale dataset, named ConductorMotion100, which consists of unprecedented 100 hours of conducting motion data. Extensive experiments on ConductorMotion100 demonstrate the effectiveness of M^2S-GAN. Our proposed approach outperforms various comparison methods both quantitatively and qualitatively. Through visualization, we show that our approach can generate plausible, diverse, and music-synchronized conducting motion.

Keywords self-supervised learning, generative adversarial network, human motion generation

1 Introduction

Music and human motions are inherently correlated. When singing, playing musical instruments, or dancing to music, humans naturally follow the rhythmic dynamics and the emotions conveyed by the music. Although computational analysis of music has been investigated for decades, the relationship between music and motion is an interdisciplinary research area that has emerged only relatively recently. With the development of generative techniques, methods capable of automatically generating musical motion have been widely explored. Recently, researchers have successfully generated dance motions [1, 2] or instrument-playing motions [3, 4] from music. However, only a few prior studies have focused on the generation of conductors' motions.

Conductors, the soul of an orchestra, perform elegantly and charmingly in every concert. Their motions are extensively planned and rehearsed before

Regular Paper
Special Section on Self-Learning with Deep Neural Networks
This work was partially funded by the Natural Science Foundation of Jiangsu Province of China under Grant No. BK20191298 and the National Natural Science Foundation of China under Grant No. 61902110.
*Corresponding Author

each performance. However, music-driven conducting motion generation has attracted less research attention than the generation of dancing, instrument playing, and singing motions. Moreover, most existing approaches [5–11] are based on predefined rules, which limits the diversity and realism of the results. Wang *et al.* [12] proposed a kernel-based hidden Markov model (KHMM) to predict conducting motion, but its rhythm adaptability and computational efficiency are poor. To the best of our knowledge, thus far, no other learning-based conducting motion generation models have been developed.

The scarcity of learning-based conducting motion generation research can be attributed to the significant associated challenges. Unlike other cross-modal generation tasks [13–16], conducting motion not only conveys basic beat information, but also contains articulatory information (legato, staccato, etc.), hints towards different parts of the orchestra (strings, woodwind, etc.), and information about the emotion conveyed by the music. Because the motion of conductors has both low-level music texture dependencies and high-level music structure dependencies, the task of music-driven conducting motion generation has a difficulty comparable to generating instrument-playing motion and dance motion at the same time. Moreover, the generation is also inherently ill-posed because of the distinctive styles of different conductors. For the same piece of music, the motions performed by different conductors may be highly distinct. As will be shown in our experiments, using standard L1 or L2 loss to regress the motion will fail to learn the one-to-many mapping and produce over-smoothed results.

To tackle these challenges, we bring the advances of recent multimodal self-supervised learning [17] to this task. As shown in Fig. 1, we integrate two self-supervised learning approaches, contrastive learning and generative learning, into a unified two-stage framework. In the contrastive stage, we devise a music motion synchronization (M^2S) learning task and design a two-branch network, Music Motion Synchronization Network (M^2S-Net), to learn rich and aligned music and motion representations in a self-supervised manner. In the subsequent generative stage, the previously learned music representations provide semantic information for the motion generator, while the motion representations are used to calculate a proposed perceptual training metric named sync loss. Finally, the proposed Music Motion Synchronization-Based Generative Adversarial Network (M^2S-GAN) is trained jointly with sync loss and Wasserstein distance based adversarial loss [18, 19], such that both music motion synchronization and motion realism are guaranteed.

In addition, we find that existing conducting motion datasets are too small to train a generative deep learning model. Thus, we collect and construct a large-scale conducting motion dataset. Based on advanced object detection [20] and pose estimation [21] techniques, we efficiently extract conducting motion data from online video sources. The constructed dataset, named ConductorMotion100, has 100 hours of conducting motion data and aligned Mel spectrograms. Its scale significantly exceeds that of existing conducting motion datasets. We conduct extensive experiments on ConductorMotion100, finding that the impressive performance achieved by our proposed approach demonstrates its effectiveness.

The main contributions of this paper can be summarized as follows.

1) We propose a novel two-stage model for music-driven conducting motion generation. The model first learns music and motion feature representations in a

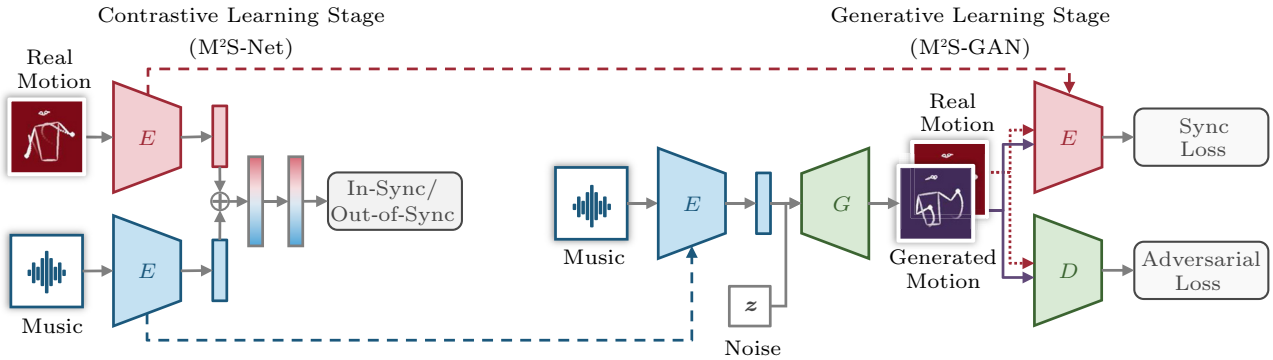


Fig.1. High-level illustration of the proposed two-stage approach. The contrastive learning and generative learning stages are bridged by transferring learned music and motion encoders, as indicated by the dotted lines. In-sync/out-of-sync is a binary cross-entropy loss function used to determine whether music and action are synchronized.

self-supervised manner, and then feeds these learned representations into adversarial training procedures to guarantee music motion synchronization.

2) We collect and construct a large-scale conducting motion dataset, ConductorMotion100, based on advanced object detection and pose estimation techniques. ConductorMotion100 contains 100 hours of conducting motion and aligned music data; its scale significantly exceeds that of existing conducting motion datasets.

3) We conduct extensive experiments on ConductorMotion100, using both standard evaluation metrics and several newly designed metrics. The competitive experimental results reveal the effectiveness of the proposed approach.

Both the dataset ConductorMotion100 and the experimental codes are open-sourced①.

2 Related Work

2.1 Music-Driven Conducting Motion Generation

Music-driven conducting motion generation involves the generation of skeleton sequences of an orchestral conductor according to a given piece of music. In 2002, Wang *et al.* [12] designed a kernel-based hidden Markov model (KHMM) and trained it to predict conducting motion from the three-dimensional pitch, loudness, and beat features. However, this model cannot automatically adapt to music tempo; during testing, it requires manual model selection according to the tempo of given music. In addition, the state observation density of KHMM is estimated by the entire sample set, the size of which is set to the scale of the entire training set. Accordingly, this method is computationally inefficient when facing a large-scale training set.

From 2003 to 2008, a group from the Netherlands performed a series of studies with the goal of developing a virtual conductor system [5–9]. They first tried to generate conducting motion based on logic programming and gesture dictionaries [5]. However, this preprogrammed generation cannot be conditioned on music. In [6], the authors improved the system to generate motion from MIDI and enabled it to adjust to tempo change dynamically. The intention of conducting motion (the hints as to tempo, loudness, etc.) was also considered in [7, 8], in which intention generation is based on a retrieval module. However, the construction of the intention database is labor-sensitive. Their project was comprehensively concluded in [9].

The most recent work on this task can be found in [10, 11], where the authors deployed multiple virtual conductors to support the rehearsal of an amateur orchestra. Their model generates different conducting motions for different parts of the orchestra according to the MIDI data. However, in the same way as most other music-driven conducting motion generation approaches, this model is not learning-based. Generally speaking, the above methods, except for [12], rely heavily on human annotation and explicitly defined rules, meaning that the realism and the diversity of the generated motions are greatly limited. By contrast, our approach has no dependencies on prior knowledge. Instead, by learning from massive amounts of training data, it can discover the music-motion relationships on its own. Moreover, since our approach does not require the MIDI data as inputs, it can generate conducting motion from raw audio files (.wav, .mp3, etc.), creating a wider range of application scenarios.

In addition, physical conducting robots have been constructed, with examples including the ASIMO robot from Honda Corporation②, the YuMi robot from ABB company③, and the Conducting Robots project from [22–24]. However we cannot find any associated publications that describe the technical details of how these robots learn to conduct. Dansereau *et al.* [25] designed a particle filter based machine learning algorithm to predict the movement of the conductor's baton in order to resolve the network delay in the context of distributed networked performance. Since their experiments show that the model can accurately forecast the motion for only several seconds, it is not applicable for music-driven conducting motion generation.

2.2 Deep Audio-to-Motion Translation

Music-driven conducting motion generation can be categorized into a broader type of learning task: audio-to-motion translation. Since no deep learning based music-driven conducting motion generation approaches have been developed to date, in this section, we introduce some recent advances of deep audio-to-motion translation: specifically, speech gesture generation and

①https://github.com/ChenDelong1999/VirtualConductor, Mar. 2022.

②https://hondanews.com/en-US/honda-corporate/releases/release-e2b2bddd0aa73108779fe0004c34bd40-hondas-asimo-robot-to-conduct-the-detroit-symphony-orchestra, Mar. 2022.

③https://new.abb.com/news/detail/2069/yumi-taking-the-stage, Mar. 2022.

musical gesture (dance and instrument playing) generation, which share numerous similarities with conducting motion generation.

The most straightforward design choice for deep audio-to-motion translation is to regress the ground-truth motion data using L1 or L2 regression loss. In [26], Yelta used the convolutional neural network (CNN) [27] to extract audio features from the spectrums, and used a long sort-term memory (LSTM) network to learn the temporal relationship. Later, in [28], the authors added contrastive loss to enforce synchronization between the motion and the music. However, this approach resulted in repetitive motion output. Models following a similar CNN-LSTM architecture design are proposed in [3]; however, these approaches use MIDI as input, leading to restricted application scenarios. Many researchers have also tried using fully LSTM models to map music features directly to motion [29–35], but due to the transition of the adopted music features being non-smooth over time, the jittering effect of these LSTM-based models is difficult to eliminate. Moreover, due to error accumulation, LSTM often outputs frozen motion during test time [2, 36, 37]. Comparatively, fully convolutional models yield relatively better results [38, 39]. Recently, Huang *et al.* [36] integrated a transformer [40]-based music encoder with a LSTM motion decoder. Li *et al.* [41] accomplished an impressive music-to-dance generation result with their transformer-based motion decoder. Despite the choice of the model structure, a shared problem encountered by the above methods is that they attempt to model a fixed mapping from audio to motion, which is in fact inherently one-to-many. Faced with such an ill-posed problem, these models tend to yield over-smooth motion.

Therefore, an increasing number of researchers have begun to apply advanced deep generative models, particularly generative adversarial nets (GAN) [42]. These models generally yield better results; however, many of them still preserve the regression loss [1, 37, 43–45]. We believe that the objective of regression loss is in conflict with the adversarial loss: the optimal output of the regression loss is over-smoothed, while the discriminator of GAN can easily learn the over-smooth pattern and provide conflict gradients to the generator. The exceptions are [46] and [47], which develop fully adversarial learning frameworks. However, the model in [46] is designed for talking head orientation generation, which is a simpler task (only two-dimensional prosodic feature inputs and three-dimensional head orientation outputs; no spatial relationships are learned). In [47], the proposed model introduces the additional supervision of the gesture phrase, leading to additional annotation requirements. One possible reason for the preservation of regression loss in other models might be the fact that it is difficult to find a better way to pose a consistency constraint, which could cause the music and generated motion to be semantically coherent and temporally aligned.

3 Data Preparation

Due to the scale of existing conducting motion datasets being insufficient to train a deep generative model, we construct a large-scale conducting motion dataset, named ConductorMotion100, by deploying pose estimation on conductor view videos of concert performance recordings collected from online video platforms. The construction of ConductorMotion100 removes the need for expensive motion-capture equipment and makes full use of massive online video resources. As a result, the scale of ConductorMotion100 has reached an unprecedented length of 100 hours. As shown in Table 1, its scale far exceeds that of existing conducting motion datasets. To facilitate related research, the dataset is made public④.

In the following, we briefly describe the construction of ConductorMotion100. In many of the collected

Table 1. Comparison on the Scale of Conducting Motion Datasets

Year	Dataset	Length (min)
2013	Sarasúa *et al.* [48]	120.0
2013	Dansereau *et al.* [25]	0.5
2014	Sarasúa and Guaus [49]	250.0
2017	Karipidou *et al.* [50]	36.0
2019	Huang *et al.* [51]	180.0
2019	Lemouton *et al.* [52] (IDEA dataset)	56.0
2021	Ours (ConductorMotion100)	6 000.0

④https://github.com/ChenDelong1999/VirtualConductor, Mar. 2022.

videos, there are also some players and audience members. Therefore, we first annotate a small object detection dataset, Concert300, and fine-tune a pre-trained YOLO-V3 [20] to recognize which human is the conductor. Concert300 consists of 300 concert images with bounding boxes denoting the conductor, players, and audiences. Once trained with Concert300, the conductor detection model achieves 0.861 precision and 0.991 recall during testing. On the basis of conductor detection, we estimate 2D pose key points using AlphaPose [21]. Since the motion of the conductor's lower body contains very little useful information and is often occluded or outside of the camera's view, we only preserve 13 2D keypoints of the upper body. The preserved key points are identical to the first 13 key points in the MS COCO format. We normalize the key points into the range of (0,1) and set the average distance between the left and the right shoulder to be 0.2. Next, we centralize the keypoint sequence by moving the midpoint of the left and the right hip to (0.50, 0.75). Finally, all motion data is re-sampled to 30 fps. The Mel-scaled spectrogram is used to represent the music. We extract a 128-bin Mel spectrogram with a hop-length of 256, and then covert the spectrogram to decibel (dB) units. The sampling rate of the derived spectrogram is 86.52 Hz, which we interpolate to 90 Hz to achieve joint music motion encoding.

Formally, the ConductorMotion100 dataset can be described as $\mathcal{D} = \{(\boldsymbol{X}_i, \boldsymbol{Y}_i)\}_{i=1}^{N}$, where $\boldsymbol{X}_i = \{\boldsymbol{x}_t\}_{t=1}^{T_i^x}$ and $\boldsymbol{Y}_i = \{\boldsymbol{y}_t\}_{t=1}^{T_i^y}$ are the i-th Mel spectrogram and conducting motion sequence respectively. Each $\boldsymbol{x}_t$ is a 128-dimension vector, corresponding to the 128 frequency bins of the spectrogram. Each $\boldsymbol{y}_t$ is a single frame of conducting motion. Since the motion is represented as the 2D coordinates of 13 joints, each $\boldsymbol{y}_t$ has 26 dimensions. The spectrogram $\boldsymbol{X}_i$ and motion $\boldsymbol{Y}_i$ are sampled at different sample rates (90 Hz and 30 Hz respectively); therefore, $T_i^x = 3 \times T_i^y$. Going forward, the subscripts i are omitted for simplicity. In Fig. 2, we present the visualization of a sample $(\boldsymbol{X}, \boldsymbol{Y})$ in the ConductorMotion100 dataset; the sample corresponds to the final part of Tchaikovsky's 1812 Overture.

4 Approach

4.1 Motivation

What makes a good conductor? First, the conductor should understand the music, which means knowing the music tempo, emotion, etc. Then, the conductor should know how to interpret the music through their body movement, i.e., translating the musical concepts into motion. The resulting conducting motion should be elegant, which means it must avoid looking unnatural. More importantly, the motion also should be precise to avoid confusing the orchestra. A good music-driven conducting motion generation model should also consider these aspects. To imitate a real conductor, we set up four neural networks, each of which corresponds to one specific requirement respectively.

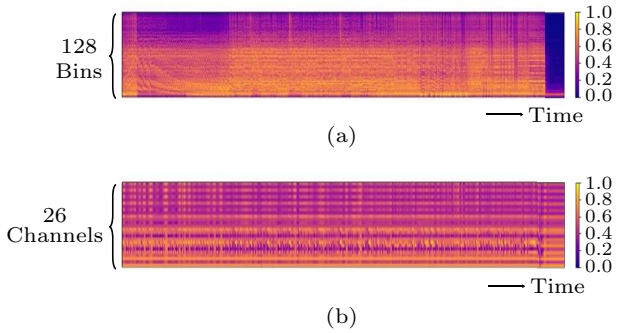


Fig. 2. Visualization of a sample in the ConductorMotion100 dataset. (a) Mel spectrogram. (b) Conducting motion.

1) *Understanding.* A music encoder extracts semantic music features from a given piece of music.

2) *Interpreting.* A generator generates a conducting motion sequence from the music feature sequence.

3) *Elegance.* A discriminator looks at the generated motion and enforces it to be visually similar to the real motion.

4) *Precision.* A motion encoder looks at the generated motion and enforces it to be musically similar to the real motion.

The next problem is how to obtain the weights of these four neural networks. The second and the third requirements are straightforward to realize: the generator and the discriminator can be jointly trained in a standard adversarial learning scheme. However, the first and the fourth requirements are far more difficult: the music encoder should effectively provide high-quality music features to bridge the semantic gap between the music and the motion, while the motion encoder should be able to analyze the motion similarity from a musical perspective (rhythm, emotion, etc.). In other words, we need a music encoder that extracts motion-related features and a motion encoder that extracts music-related features. Here, we find that the multimodal self-supervised learning technique is particularly suitable for obtaining the weights of this pair of encoders. When using them to build up a two-

branch network and applying self-supervised learning, the music encoder and the motion encoder can supervise each other, and subsequently construct a joint feature space that is simultaneously music-related and motion-related.

4.2 Overview of Proposed Approach

Formally, given the ConductorMotion100 dataset $\mathcal{D}$, our goal is to learn a music encoder E_{music} and a generator G to predict a motion sequence from a given Mel spectrogram $\boldsymbol{X}$ and random $\boldsymbol{z}$ sampled from a normal distribution, i.e., $\hat{\boldsymbol{Y}} = G(E_{\text{music}}(\boldsymbol{X}), \boldsymbol{z})$. The generated motion distribution $\mathbb{P}_G$ should approximate the data distribution $\mathbb{P}_{\text{data}}(\boldsymbol{Y})$ and the conditional distribution $\mathbb{P}_c(\boldsymbol{Y}|\boldsymbol{X})$ simultaneously. To learn $\mathbb{P}_{\text{data}}$, we deploy a discriminator $D(\boldsymbol{Y})$ that is trained to distinguish $\mathbb{P}_G$ and $\mathbb{P}_{\text{data}}$ and calculate the adversarial loss. For $\mathbb{P}_c$, we use a motion encoder $E_{\text{motion}}(\boldsymbol{Y})$ to compare the motion features of $\boldsymbol{Y}$ and $\hat{\boldsymbol{Y}}$ and calculate a proposed sync loss. G is then trained to satisfy D and E_{motion} by minimizing the adversarial loss and the sync loss respectively.

To obtain the weights of E_{music} and E_{motion}, we design a preparation stage called the contrastive learning stage, in which a music motion synchronization network (M^{2}S-Net) is trained with a music motion synchronization (M^{2}S) learning task. The M^{2}S-Net consists of the motion encoder $E_{\text{motion}}(\boldsymbol{Y})$, the music encoder $E_{\text{music}}(\boldsymbol{X})$, and fuse layers. In M^{2}S learning, we sample positive (in-sync) and negative (out-of-sync) music motion pairs from $\mathcal{D}$, and then train M^{2}S-Net to predict whether the input pair is positive or negative. Subsequently, the trained E_{music} and E_{motion} are transferred to the generative learning stage, where these two encoders, together with the generator and discriminator, form the Music Motion Synchronized Generative Adversarial Network (M^{2}S-GAN). We use the word "synchronized" here to imply that M^{2}S-GAN requires a preparation stage and that the weights of the transferred encoders are frozen. An overview of the two-stage training procedure can be found in Algorithm 1.

4.3 Network Architectures

As shown in Fig.3, four neural networks are involved in our approach: a music encoder E_{music}, a motion encoder E_{motion}, a generator G, and a discriminator D. In the first contrastive learning stage, E_{music}, E_{motion}, and three fusing layers form M^{2}S-Net. The outputs of the two encoders are concatenated and passed to the three fully-connected fusing layers. A sigmoid activation is applied in the final layer to output a possibility in the range of (0,1), indicating the prediction about whether the given input pair is synchronized. In the following generative learning stage, all four of these networks form M^{2}S-GAN. G generates a motion sequence according to $\boldsymbol{z}$ and the output of E_{music}. The generated motion is then fed to E_{motion} and D for the sync loss and the adversarial loss respectively. In the below, we provide a detailed description of the network architectures.

4.3.1 *Motion Encoder*

E_{motion} should be able to analyze the conducting motion from both spatial and temporal perspectives.

Algorithm 1. Training Procedure of M^{2}S-Net and M^{2}S-GAN

Input: dataset $\mathcal{D} = \{(\boldsymbol{X}_i, \boldsymbol{Y}_i)\}_{i=1}^{N}$; loss function weights λ_{adv}, λ_{sync}, w_{GP};
Output: trained music encoder E_{music}, generator G;

```
/* Contrastive learning stage */
Initialize E_music, E_motion, and f;
while not converge do
    Sample positive/negative pairs (X_i, Y_j) and the corresponding label c_{i,j} from D;
    Predict music motion synchronization: f[E_music(X_i) ⊕ E_motion(Y_j)];
    Update E_music, E_motion, and f with L_{M2S-Net}; // (1)
end
/* Generative learning stage */
Initialize G, D, load network weights of E_music, E_motion;
while not converge do
    Sample music motion pairs (X_i, Y_i) from D;
    Sample z from normal distribution;
    Generate conducting motion: Ŷ = G(E_music(X), z)
    Update G with L_G; // (2)
    Update D with L_D; // (3)
end
```

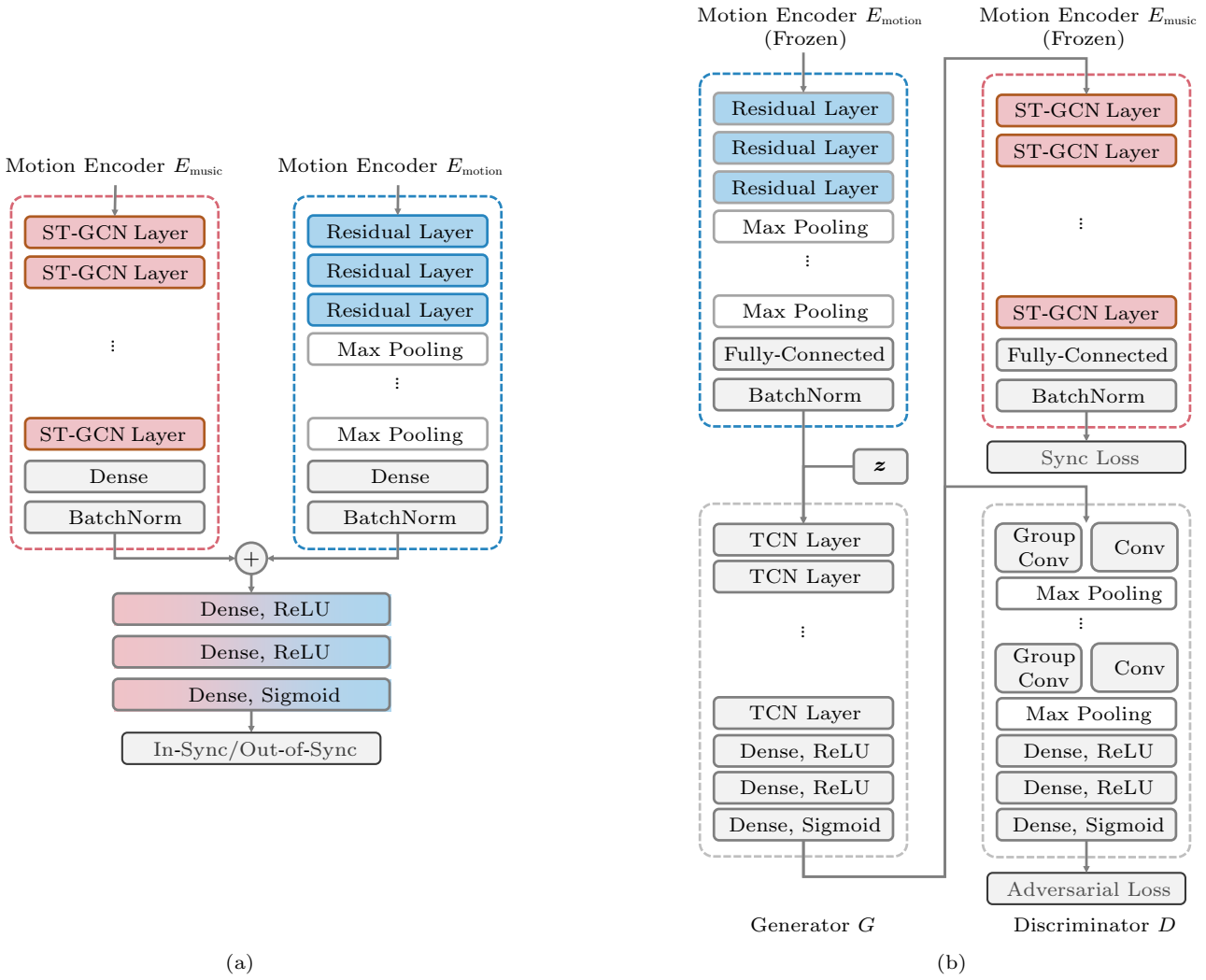


Fig.3. Network structure of M^{2}S-Net and M^{2}S-GAN. In-sync/out-of-sync is a binary cross-entropy loss function used to determine whether music and action are synchronized. (a) M^{2}S-Net. (b) M^{2}S-GAN. Conv: convolutional layer.

Therefore, we use the Spatial-Temporal Graph Convolutional Network (ST-GCN) [53], which was originally designed for human pose recognition. E_{motion} has 10 ST-GCN layers. In each of these layers, graph convolution and temporal convolution extract the spatial and the temporal information respectively, while a 1×1 convolution helps to build up the residual connection. No down-sampling is applied; thus, the sampling rate of the output motion feature is 30 Hz, the same as the input motion. Each convolution layer has 32 channels.

4.3.2 *Music Encoder*

E_{music} extracts music features from the 2D Mel spectrogram. It has three groups of layers, where each group consists of three residual layers and a max-pooling layer. Each residual layer is composed of a 2D convolution with a kernel size of 3×3, batch normalization, ReLU activation, and skip connection. The max-pooling layers down-sample the feature maps on both the frequency and time axes. Only one downsample operation with a factor of 3 is performed in the time axis so that the output music features have the same sampling rate as the motion features. The 2D convolution layers have 16 channels before the ×3 down-sampling and 32 channels after the ×3 down-sampling.

4.3.3 *Generator*

G generates the conducting motion according to the music feature sequence extracted by E_{music} and noise vector sequence sampled from a normal distribution. G is based on a temporal convolution network (TCN) [54], which achieves similar performance to standard LSTM with a much lower computation time. It has six dilated 1D convolution layers with residual connections. Each

layer has 64 channels with a kernel size of 5. Moreover, the original TCN is causal because it was originally designed for a forecasting task; we remove the causal structure to learn bi-directional dependencies. The noise $\boldsymbol{z}$ is an 8-dimensional vector in 1 Hz. Several transpose convolution layers up-sample the noise to 30 fps before feeding it into TCN.

4.3.4 Discriminator

The discriminator D decides whether the motion sequence is real or fake. Note that several GAN-based audio-to-motion translation methods also feed the audio into the discriminator [1,37,43]. Here, however, our D only observes the motions, since we view learning the distribution of real motion $\mathbb{P}_{\text{data}}(\boldsymbol{Y})$ and learning the conditional distribution $\mathbb{P}_c(\boldsymbol{Y}|\boldsymbol{X})$ as two very different tasks. D has a two-branch structure, where one branch learns the dynamics of individual joints by means of group convolution while the other learns the spatial relationships by means of normal 1D convolutions. The motion features extracted by the two branches are then concatenated and passed to the dense layers. We find that the down-sampling is crucial for improving D. The final output of D is a single scalar in 2.5 Hz sampling rate. We also attempted to use an ST-GCN as D, but found that it leads to a very unstable training procedure during experiments.

4.4 Loss Functions

4.4.1 Loss Function for M^{2}S-Net

In the contrastive learning stage, we adopt binary cross-entropy loss to train M^{2}S-Net. The definition of $\mathcal{L}_{\text{M}^2\text{S-Net}}$ is shown in (1); here, $f[\cdot]$ represents the fusing layers of M^{2}S-Net, $\oplus$ denotes the feature concatenation operation, and c_{ij} is the label indicating whether $\boldsymbol{X}_i, \boldsymbol{Y}_j$ is a positive pair or a negative pair. During training, the positive and negative pairs are automatically sampled from the dataset. Positive pairs are naturally synchronized music and motion sequences. Negative pairs are mismatched sequences, of which the details will be introduced in Subsection 4.5.

$$\begin{aligned}&\mathcal{L}_{\text{M}^2\text{S-Net}}\\ &=\sum_{i,j=1}^{M} c_{ij}\log_2\Big(f[E_{\text{music}}(\boldsymbol{X}_i)\oplus E_{\text{motion}}(\boldsymbol{Y}_j)]\Big)+\\ &\quad(1-c_{ij})\log_2\Big(1-f[E_{\text{music}}(\boldsymbol{X}_i)\oplus E_{\text{motion}}(\boldsymbol{Y}_j)]\Big),\end{aligned}\tag{1}$$

where c_{ij} is defined by

$$c_{ij}=\begin{cases}1, & \text{if } i=j,\\ 0, & \text{otherwise.}\end{cases}$$

4.4.2 Loss Function for M^{2}S-GAN

In the generative learning stage, the generator generates the conducting motion by $\hat{\boldsymbol{Y}}=G(E_{\text{music}}(\boldsymbol{X}),\boldsymbol{z})$. Its loss function is shown in (2). The first and the second term are the sync loss and the adversarial loss, respectively, while λ_{sync} and λ_{adv} are their weights respectively. The loss function of the discriminator is shown in (3). The third term is the gradient penalty term of the Wasserstein GAN [19], where $\tilde{\boldsymbol{Y}}$ is obtained via the random linear interpolation between $\hat{\boldsymbol{Y}}$ and $\boldsymbol{Y}$. Note that, here, G and D minimize $\mathcal{L}_G$ and $\mathcal{L}_D$ respectively, but E_{music} and E_{motion} do not participate in the optimization; their weights are directly transferred from the trained M^{2}S-Net and frozen in M^{2}S-GAN.

$$\begin{aligned}\mathcal{L}_G &= \sum_{i=1}^{N}\lambda_{\text{sync}}||E_{\text{motion}}(\hat{\boldsymbol{Y}}_i)-E_{\text{motion}}(\boldsymbol{Y}_i)||_2^2-\\ &\quad\lambda_{\text{adv}}D(\hat{\boldsymbol{Y}}_i),\end{aligned}\tag{2}$$

$$\begin{aligned}\mathcal{L}_D &= \sum_{i=1}^{N}D(\hat{\boldsymbol{Y}}_i)-D(\boldsymbol{Y}_i)+\\ &\quad w_{GP}\mathbb{E}_{\tilde{\boldsymbol{Y}}}||\nabla_{\tilde{\boldsymbol{Y}}}D(\tilde{\boldsymbol{Y}})-1||_2^2.\end{aligned}\tag{3}$$

4.5 Negative Pair Sampling

During M^{2}S learning, the M^{2}S-Net is trained by automatically generated aligned positive pairs, misaligned negative pairs, and their corresponding binary labels c_{ij}. The positive pairs are naturally synchronized music-motion sequences. As noted by Korbar *et al.* [17] and illustrated in Fig.4, three types of negative sampling strategies can be used for negative pairs.

• *Easy Negatives.* Easy negative pairs are selected from different samples (i.e., different pieces of music) within a mini-batch. This strategy is similar to the negatives used in AVC learning [55].

• *Hard Negatives.* Hard negative pairs are selected from the same samples. We force the two sampled pairs to be at least 10 seconds apart.

• *Super-Hard Negatives.* Super-hard negative pairs are also selected from the same samples, but they are sampled by random temporal shifts within the range of 0.5 s to 5 s, which is similar to [56].

We adopt hard negatives for M^{2}S learning. Compared with the AVC-like learning task on unconstrained

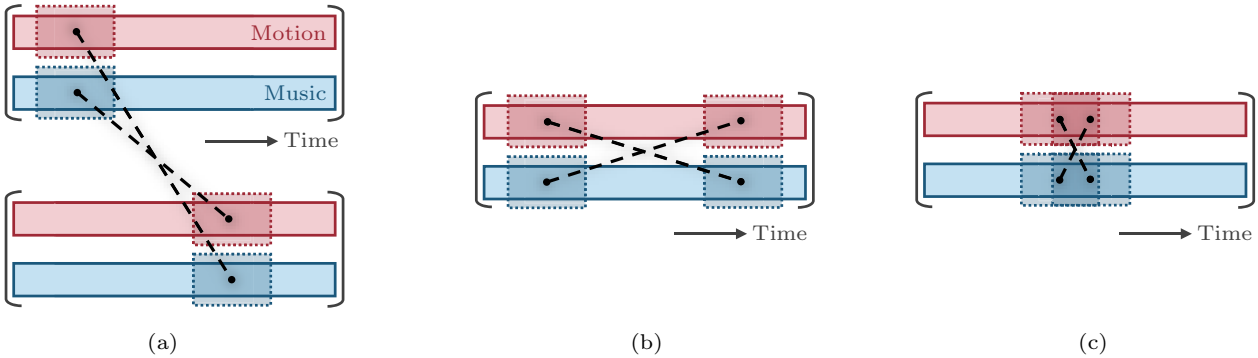


Fig. 4. Illustration of different negative pair sampling strategies. (a) Easy negative pairs. (b) Hard negative pairs. (c) Super-hard negative pairs.

web video data, our M^{2}S learning on ConductorMotion100 is a relatively fine-grained task. We find that the fine-grained nature of this task increases the chance of false negative sampling⑤ in easy negatives, which would have an undesirable effect on the M^{2}S learning process. During training, the easy negatives will enumerate all cross-sample combinations of music and motion, which will encourage the model to isolate different samples and memorize their identities. Hard and super-hard negative sampling cuts off the cross-sample access. However, the super-hard negatives limit the temporal shifts in a small range, meaning that they cannot provide any semantic-related information. Hard negatives have a range of at least 10 seconds, which is able to accommodate semantic differences. In what follows, unless explicitly noted, hard negatives are used for M^{2}S learning.

5 Discussion

5.1 Applying Self-Supervised Learning for Generative Tasks

There are four previous published studies [57–60] that adopt self-supervised learning in generative procedures. Chen *et al.* [57] added a self-supervised loss to the discriminator of a conditional generative adversarial network (CGAN); afterwards Hao *et al.* [58] extended the model to a cross-modal cycle generative adversarial network (CMCGAN). In an audio inpainting study, Zhou *et al.* [59] added a self-supervised loss to provide an audio-visual-related feature. Choi *et al.* [60] first deployed a cross-modal identity matching task, and then transferred the learned features to a CGAN. Although the architectures of these studies [57–60] appear similar to that of our proposed approach, there are three fundamental differences between them.

1) The visual data is represented as a single frame of image in [57,58,60], and they all use only easy negatives. Therefore, the model can only learn the semantic correspondence, while no cross-modal temporal synchronization is learned. By contrast, our M^{2}S-Net takes the sequence of conducting motion as input and learns the temporal synchronization between the conducting motion and the music.

2) In [57, 58, 60], the self-supervision requires predefined labels denoting instrument type [57, 58] or face identity [60]. However, our approach is fully self-supervised: it requires no manual annotation, and instead relies only on the natural synchronization of the motion and the music.

3) The methods in [57–59] have only a single learning stage, where the self-supervised loss is part of the joint loss of the feature encoder [59] or the discriminator [57, 58]. At the beginning of training, these methods are non-optimal in terms of the self-supervised task, which could introduce a certain degree of randomness. For its part, our proposed approach has two learning stages: we first obtain an optimal M^{2}S-Net, and then apply it to M^{2}S-GAN.

5.2 Sync Loss vs Perceptual Loss

Perceptual loss [61] is a popular choice in many ill-posed image manipulation tasks. It refers to using a pre-trained neural network (as the perceptual loss network) to extract features from generated and ground truth samples, then measuring the distance between those deep features. The effectiveness of perceptual loss in solving the over-smoothing problem has been widely

⑤False negative sampling refers to the case in which, in a given sampled negative pair, the sampled music sequence and motion sequence have a very similar rhythm and emotion, but the label indicates they are not synchronized.

validated in the field of computer vision, but it is rarely used in motion generation tasks.

There are three main types of pre-training tasks for the perceptual loss network: classification, reconstruction, and discrimination. The ImageNet pre-trained VGG has become a standard choice of the perceptual loss network. In a study of music-driven dance generation, Ren *et al.* [1] proposed pose perceptual loss, where a motion encoder pre-trained on dance genre classification (distinguishing ballet, pop, and hip-hop dance) was used as the perceptual loss network. However, no such genre labels exist for conducting motion, meaning that classification-based perceptual loss networks are inapplicable. A reconstruction task refers to training an AutoEncoder [62] as a perceptual loss network; however, the features obtained from such a network are not selective, which introduces the problem of regression loss. The third type involves using the feature of a GAN's discriminator to calculate perceptual loss [63]. This discriminator only pays attention to realism-related features; accordingly, it does not measure semantic similarity.

Our proposed sync loss is conceptually similar to perceptual loss. However, there are two important differences between them.

1) The perceptual loss measures the feature distance from multiple layers, but the sync loss only uses the feature of the final layer. We skip this aggregation process, since the training loss of M^{2}S-Net produces an aligned feature space at the end of the two encoders.

2) The relationship between the perceptual loss's pre-training tasks (classification, reconstruction, discrimination) and applied tasks (e.g., neural style transfer) is weak. By contrast, the pre-training task of sync loss (i.e., M^{2}S learning) is strongly related to the generative task. This self-supervised pre-training task enables the sync loss to extract the most music-related motion features, in contrast to category-related features (classification task), information-retained features (reconstruction task), or realism-related features (discrimination task).

6 Experiments

6.1 Implementation Details

All models are implemented in Pytorch⑥ with a single NVIDIA 2080Ti GPU. As suggested in [19], our M^{2}S-GAN is trained by the RMSprop optimizer [64] with a learning rate of 0.000 5. M^{2}S-Net and other comparable models are trained by the Adam optimizer [65] with a learning rate of 0.001. The batch size of the contrastive and generative learning stage is 10 and 3 respectively. Overall, the training of our approach (M^{2}S-Net + M^{2}S-GAN) takes approximately 48 hours. The two stages require a similar training time. The training, validation, and testing sets are split in proportions of 9 : 0.5 : 0.5, resulting in 90 hours of the training set, 5 hours of the validation set, and 5 hours of the testing set.

Contrastive Learning Stage. The positive and negative pairs are sampled from 30 s music-motion pairs. The length of each pair is 10 seconds. Since we determine that the models under hard or super-hard negatives are difficult to train from scratch, the easy negatives are used for pre-training in the first epoch of each model.

Generative Learning Stage. To facilitate the learning of long-term dependencies, the sample length is set to 60 seconds. For hyper-parameters of loss functions, we empirically set $w_{GP} = 10$. Following WGAN-GP [19], in each step, we train the discriminator five times and train the generator once.

6.2 Evaluation Metrics

Since learning music-driven conducting motion generation is a very new task, there are few existing metrics available for measuring the outcomes. In recent years, the objective evaluation of deep generative models has usually been based on the inception score (IS) or the frechet inception distance (FID). Notably these metrics require a feature encoder, which is typically obtained by classification pre-training; however, there are no available class labels for the conducting motions. Therefore, to evaluate the quality of conducting motions more effectively, we propose several new metrics in addition to the existing metrics, as detailed in the below. To provide a better understanding of the characteristics of these metrics, we illustrate the changes in metric values under spatial-temporal perturbation in Fig.5. Spatial perturbation is performed by amplifying the motion by an amplitude scaling factor. Since our motion data is represented by coordinates, to preserve the spatial relationships during perturbation, we first subtract the mean pose before multiplying by the amplitude scaling factor, and then add the mean pose back to the scaled motion. Temporal perturba-

⑥https://pytorch.org, Apr. 2022.

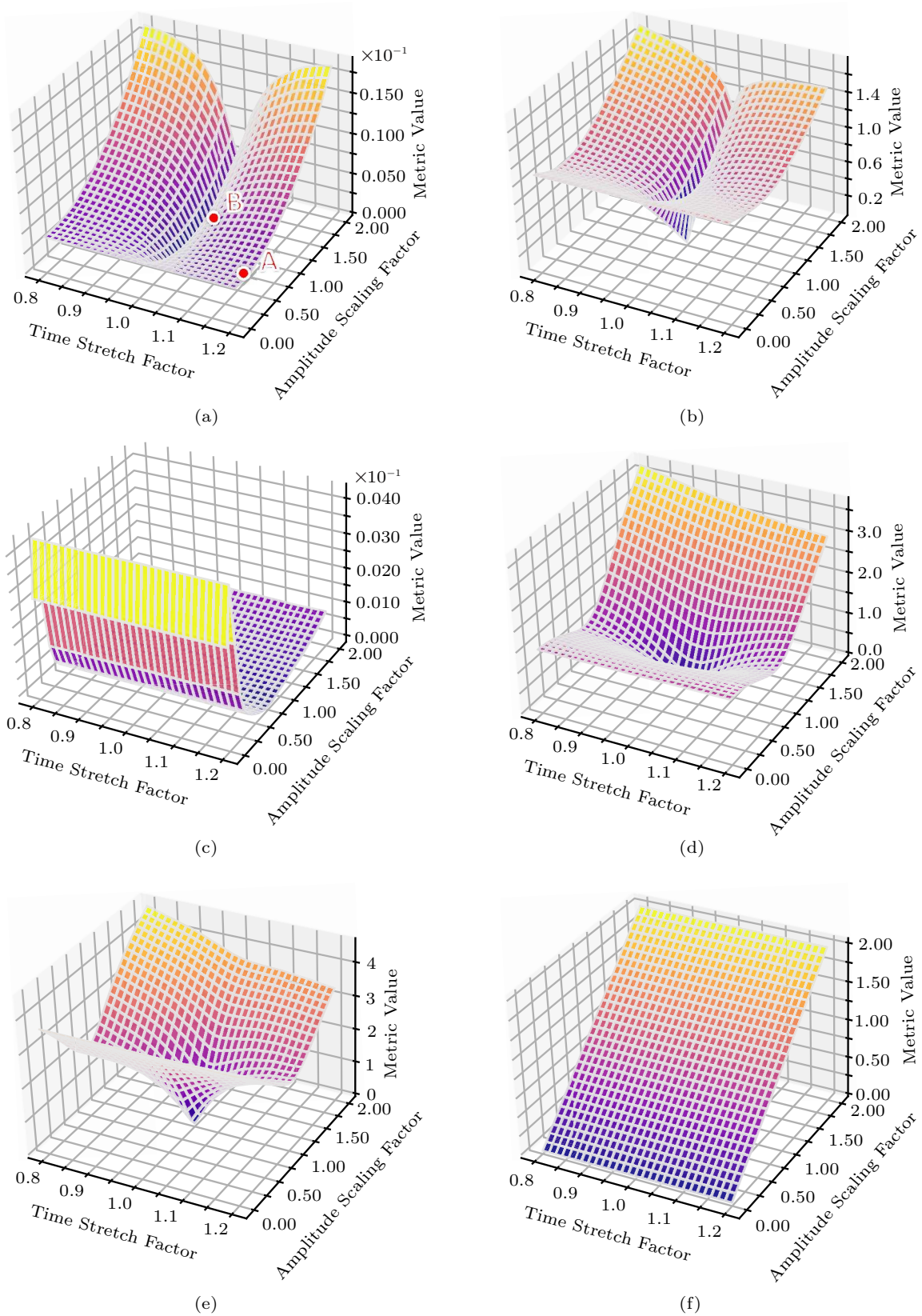


Fig.5. Changes in metric values under spatial-temporal perturbation. (a) Mean squared error. (b) Sync error. (c) Wasserstein distance. (d) Rhythm density error. (e) Strength contour error. (f) Standard deviation percentage.

tion is performed by stretching the motion data along the time axis. We perform the perturbation on 100 random motions and obtain the averaged results.

6.2.1 Mean Squared Error

Mean squared error (MSE) is the most straightforward way to measure how close the generated motion is to the ground truth, and has thus been widely used as an evaluation metric by existing audio-to-motion translation work [3, 4, 29, 30, 32, 43]. The only existing learning-based music-driven conducting motion generation method, KHMM [12], also adopts a mean absolute error (MAE)-like metric. MSE and MAE are similar despite a small difference in their robustness to outliers. Here, following the majority of existing work, we opt to use MSE. However, MSE has a strong preference for small-amplitude motions. As shown in Fig.5, this preference creates the issue of an out-of-sync motion with a very small amplitude (point A) that has a lower MSE than a less out-of-sync motion with normal amplitude (point B). Given ground-truth motion $\boldsymbol{Y} = \{\boldsymbol{y}_t\}_{t=1}^{T^y}$ and generated motion $\hat{\boldsymbol{Y}} = \{\hat{\boldsymbol{y}}_t\}_{t=1}^{T^y}$, MSE is defined as follows:

$$MSE(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = ||\boldsymbol{Y} - \hat{\boldsymbol{Y}}||_2^2. \tag{4}$$

6.2.2 Sync Error

Sync error (SE) is very similar to sync loss. However, it would not be suitable to use the sync loss as an evaluation metric, since our proposed M^{2}S-GAN directly minimizes it. Instead, we first perform M^{2}S learning on the testing set, and then use the learned E_{motion} to extract the motion features. Note that this approach does not result in any data leakage, since E_{motion} trained on the testing set is not involved in the training of M^{2}S-GAN. Compared with MSE, SE does not exhibit preference for over-smoothed motions, as shown in Fig.5. A less out-of-sync motion always has a lower SE, regardless of the motion amplitude. SE is defined as the MSE of motion features:

$$SE(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = ||E_{\text{motion}}(\boldsymbol{Y}) - E_{\text{motion}}(\hat{\boldsymbol{Y}})||_2^2. \tag{5}$$

6.2.3 Wasserstein Distance

Wasserstein distance (W-dis) is the Earth-mover distance between the distribution of real motion $\mathbb{P}_{\text{data}}$ and the generated distribution $\mathbb{P}_G$. A discriminator D is trained with $\mathcal{L}_D$ ((3)) parallel with the generator, even if the generator is not trained with the GAN loss. In Fig.5, we cancel the spatial relationship preservation during spatial perturbation to create variations in motion realism. We can observe that W-dis can recognize the unnaturalness of motion, but it does not change with time perturbation. This demonstrates that W-dis only measures realism and does not pay attention to consistency. W-dis is computed as follows:

$$W\text{-}dis(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = D(\boldsymbol{Y}) - D(\hat{\boldsymbol{Y}}). \tag{6}$$

6.2.4 Rhythm Density Error

We propose rhythm density error (RDE) to measure the frequency-domain similarity between two motions. In Fig.6(a) and Fig.6(b) we plot the spectrogram and power spectral density (PSD) of motions respectively. These motions correspond to the three movements of Mozart's Piano Concerto No.17 in G major, K.453. The three movements are in different tempos, meaning that we can clearly observe the differences between their PSD. However, very low-frequency motions (corresponding to body swing and turning, shown as gray bars) usually have much larger amplitude, which affects the comparison of rhythm components (red bars). Taking 40 BPM as a reasonable lower bound of music tempo, we assume that the motion rhythm component (as indicated by the red arrow) should always have a frequency higher than 0.7 Hz. The definition of RDE is shown in (7). $\boldsymbol{Y}[:, j]$ represents the j-th dimension of $\boldsymbol{Y}$ across time. We use log and a constant $k = 10^7$ to scale RDE to a proper range:

$$RDE(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \log\left(k\left\|\sum_j^{26} \underset{f>0.7\,\text{Hz}}{PSD}(\boldsymbol{Y}[:, j]) - \sum_j^{26} \underset{f>0.7\,\text{Hz}}{PSD}(\hat{\boldsymbol{Y}}[:, j])\right\|_2^2 + 1\right). \tag{7}$$

6.2.5 Strength Contour Error

Motion speed information is widely adopted by approaches aiming to computationally analyze conduction motion [49, 66–68]. Here, we propose strength contour error (SCE) as a measurement of the similarity in intensity of conducting motions. We first calculate the first-order temporal differences of each motion dimension and then sum them up, resulting in a series of summed motion speeds. However, directly comparing the summed motion speed is not robust to local slight misalignment; therefore, we add an average pooling (denoted by "*Pool*") to down-sample the summed motion speed. The kernel size and stride of the pooling are empirically set to 60 (2 seconds) and 30 (1 second) respectively. The result after pooling is referred to as the

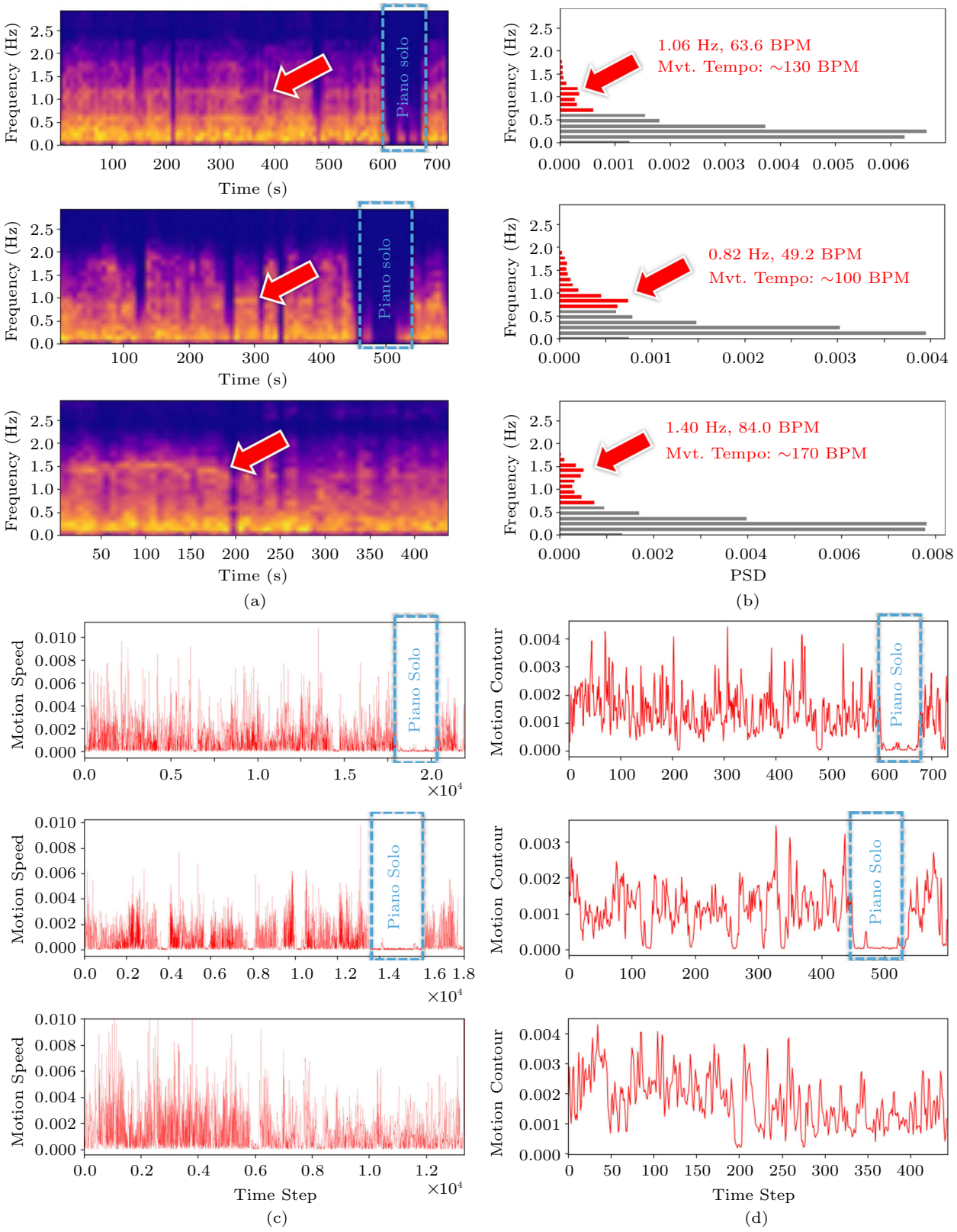


Fig.6. Three movements (Mvt.) of Mozart's Piano Concerto No.17 in G major, K.453. (a) Motion spectrogram. (b) Motion PSD. The red bars in PSD show the frequency bins above the threshold (0.7 Hz). RDE is the difference between those red bars. Red arrows indicate the rhythm component of the conducting motion. We annotate the corresponding motion frequencies, which are very close to half the value of the music tempo. (c) Summed motion speed. (d) Strength contour. SCE is the difference between these strength contours. Both strength contour and summed motion speed can recognize the piano solo section in Mvt.1 and Mvt.2.

strength contour, and SCE is used to compare strength contours. Fig.6(c) and Fig.6(d) present the examples of summed motion speed and strength contour, respectively, from which the piano solo sections can be clearly identified. The definition of SCE is shown in (8); here, $\sum_j^{26} \Delta \boldsymbol{Y}[:, j]$ is the summed motion speed of $\boldsymbol{Y}$. In the same way as for RDE, we add log and a constant $k = 10^7$ to SCE:

$$SCE(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \log\left(k\left\|Pool\left(\sum_j^{26} \Delta \boldsymbol{Y}[:, j]\right) - Pool\left(\sum_j^{26} \Delta \hat{\boldsymbol{Y}}[:, j]\right)\right\|_2^2 + 1\right). \quad (8)$$

6.2.6 *Standard Deviation Percentage*

The standard deviation (SD) measures the averaged intensity of motion change over time. We can compare the SD of the generated motion and ground truth motion to see whether the model has an over-smoothness problem. As shown in Fig.5, standard deviation percentage (SDP) linearly corresponds to the motion amplitude. SDP is defined in (9); here, $\boldsymbol{y}_t$ is the i-th motion frame, T^y is the total number of frames, and $\overline{\boldsymbol{y}}$ is the averaged keypoints position over time. An ideal conducting motion generation model should achieve an SDP of around 100%, while the static motion should have an SDP of 0%.

$$SDP(\boldsymbol{Y}) = \frac{SD(\hat{\boldsymbol{Y}})}{SD(\boldsymbol{Y})} \times 100\%,$$
$$\text{where} \quad SD(\boldsymbol{Y}) = \sqrt{\frac{\sum_{t=1}^{T^y} ||\boldsymbol{y}_t - \overline{\boldsymbol{y}}||_2^2}{T^y - 1}}. \quad (9)$$

6.3 Balancing Sync Loss and Adversarial Loss

We first use the validation set to determine the optimal settings of λ_{sync} and λ_{adv}. Fixing $\lambda_{\text{adv}} = 1$, we test $\lambda_{\text{sync}} = \{0.001, 0.01, 0.02, 0.05, 0.1, 1\}$ and compare the performance on RDE and SCE. The results are shown in Fig.7. We find that $\lambda_{\text{sync}} = 0.05$, $\lambda_{\text{adv}} = 1$ achieve the best performance on both RDE and SCE. According to the change of training sync loss and W-dis, a position either further to the left (larger λ_{adv}, more emphasis on realism) or further to the right (larger λ_{sync}, more emphasis on consistency) would cause one loss term to dominate the other, leading to increase in RDE and SCE. Therefore, we determine that the optimal settings are $\lambda_{\text{sync}} = 0.05$ and $\lambda_{\text{adv}} = 1$. We will use these settings in the following experiments.

6.4 Performance Comparison

Next, we compare the performance of music-driven conducting motion generation on the testing set. As introduced in Subsection 2.1, most existing music-driven conducting motion generation methods [5–11] require the MIDI data as input; by contrast, our approach aims at generating motions from audio input. Moreover, it is difficult to design a fair comparison between these rule-based methods and our learning-based methods. The only learning-based music-driven conducting motion generation model is KHMM [12], first proposed in 2003. However, this method requires manually selecting the appropriate model for the piece of music in question. In addition, their method is computationally inefficient, especially when facing our large-scale ConductorMotion100 dataset. Since none of the existing music-driven conducting motion generation methods are suitable for comparison, we instead select three deep models that

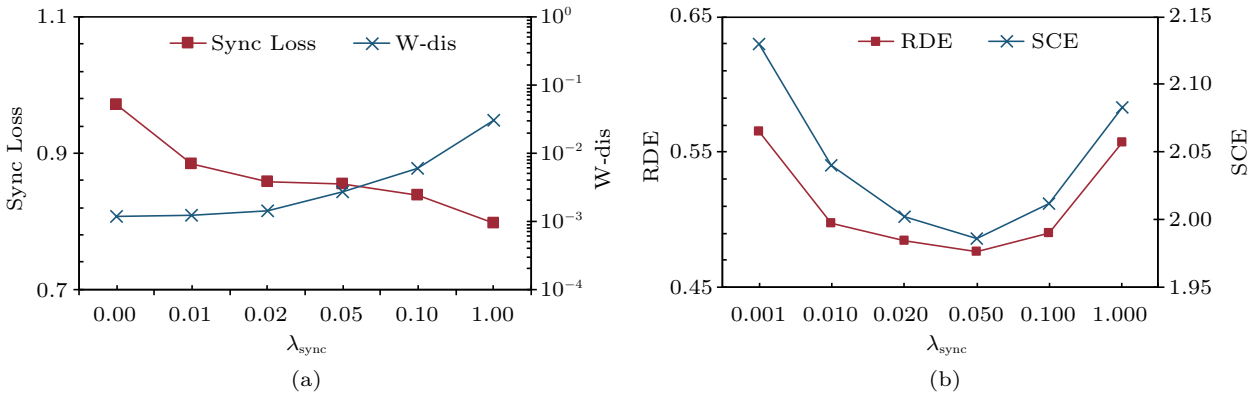


Fig.7. Training sync loss, W-dis, RDE, and SCE under different hyper-parameters. λ_{adv} is fixed to 1. We find that $\lambda_{\text{sync}} = 0.05$ yields the best performance on RDE and SCE. (a) Sync loss and W-dis. (b) RDE and SCE.

are originally designed for other audio-motion translation tasks: music-driven dance generation [28], speech gesture generation [43], and instrument-playing motion generation [32]. As none of these models require any specific prior knowledge, they can be directly trained to generate conducting motion.

1) *Shlizerman et al.* (*LSTM*) [32]. This is an LSTM-based model originally designed for translating music to instrument-playing motions (violin and piano). Following the design in [32], this model consists of a single-layer uni-directional LSTM with 200 hidden units and several fully-connected layers. It takes MFCC as the input. The model is trained with MSE loss.

2) *Yalta et al.* (*CNN-LSTM*) [28]. This model is designed for music-driven dance generation. It uses a CNN to extract music features from the Mel spectrogram, and then uses an LSTM encoder-decoder to learn temporal dependencies and predict motions. A contrastive loss is applied to enforce the feature from the LSTM encoder to be aligned with the motion. This model is trained with MSE loss and contrastive loss.

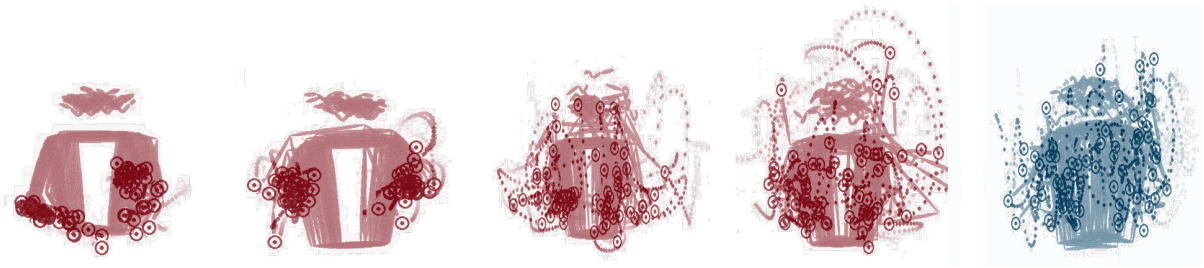
3) *Ginosar et al.* (*GAN*) [43]. This comparison is a hybrid method that combines the loss function of [43] and the music feature used by KHMM [12]. Specifically, the loss function is a joint loss composed of an adversarial loss and L1 loss. We add a gradient penalty term to the adversarial loss to improve the training stability. The music feature comprises three-dimensional pitch, loudness, and beat features. The beat data is modified to sawtooth the wave-like data, as in [12].

The performances of the above comparison methods and our proposed M^{2}S-GAN are listed in Table 2. Bolded ones indicate the best results. As the table shows, our M^{2}S-GAN outperforms all comparison methods on SE, RDE, SCE, and W-dis. The superior results on SE, RDE, and SCE indicate that M^{2}S-GAN can model the music-motion relationships the most accurately. M^{2}S-GAN's advantages on SDP and W-dis indicate that it generates the most realistic conducting motion. Notably, M^{2}S-GAN does not achieve the lowest MSE. Since MSE has a preference for over-smoothed motion, it cannot accurately reflect the motion realism or consistency. The low MSE of MSE loss-based comparisons LSTM [32] and CNN-LSTM [28] is caused by low SDP rather than high motion realism or consistency.

In Fig.8, we present the motion distribution generated from the first movement of Beethoven's Symphony No. 5 in C minor, Op. 67 for qualitative comparison. The motions are plotted in 0.1 fps, and the length of the hand trajectory is 30 time steps, i.e., 1 second. We also show the distribution of the ground truth motion. It can be seen that the first two MSE loss-based comparison methods suffer greatly from over-smoothing problems. Comparatively, the motion distributions generated by GAN-based approaches (GAN [43] and Our M^{2}S-GAN) look much closer to the real motion. Their SD and SDP also reflect this point. Comparing these two GAN-based approaches, we find that the motion generated by our M^{2}S-GAN conforms far more closely with the music. However, it is difficult to

Table 2. Performances of Music-Driven Conducting Motion Generation

Model	MSE ($\times 10^3$)	SE	W-dis ($\times 10^3$)	RDE	SCE	SDP (%)
Shlizerman *et al.*, LSTM [32]	3.50	1.301	87.470 0	0.973 9	2.511	38.98
Yelta *et al.*, CNN-LSTM [28]	**3.08**	0.911	50.850 0	0.991 1	2.482	27.11
Ginosar *et al.*, GAN [43]	6.60	1.371	29.980 0	0.943 7	2.864	97.93
Ours, M^{2}S-GAN	5.40	**0.883**	**1.426 4**	**0.049 0**	**2.046**	**99.62**



(a) (b) (c) (d) (e)

Fig.8. Conducting motion distribution of Beethoven's Symphony No.5. (a) LSTM [32], SD = 0.017 32, SDP = 36.98%. (b) CNN-LSTM [28], SD = 0.011 06, SDP = 27.11%. (c) GAN [43], SD = 0.043 51, SDP = 97.93%. (d) M^{2}S-GAN, SD = 0.043 51, SDP = 99.62%. (e) Real, SD = 0.044 43, SDP = 100%.

demonstrate this point in figures. Therefore, we create a demo video comparing the motion generated by these approaches. We also include a Turing test in the video. The video can be found online[7].

6.5 Impact of Different Negative Pairs

As addressed in Subsection 4.5, hard negatives should have the advantage over easy and super-hard negatives for M^{2}S learning. Here, we demonstrate this point by experiment: specifically, we train M^{2}S-Net using different types of negatives and compare their testing accuracies. The results are shown in Table 3. Bolded ones indicate the best results. Impressively, the model trained with hard negatives achieves the best performance on all three testing negatives. This indicates that hard negatives enable M^{2}S-Net to learn both semantic correlation (easy negatives) and temporal synchronization (super-hard negatives). Moreover, as shown in Fig.9, the training of easy negatives is very unstable compared with hard and super-hard negatives. We speculate that the most likely scenario is as follows: under easy negatives, M^{2}S-Net sometimes seeks to learn the semantic correspondence, and sometimes seeks to isolate the identity of each sample. In many other multimodal self-supervised learning tasks, the easy negatives can be used as a strong baseline. Herein, however, the experiments demonstrate that easy negatives are not suitable for M^{2}S learning. Moreover, in [17], Korbar *et al.* found that super-hard negatives were very difficult to optimize. In our experiment, although the super-hard negatives under-perform hard negatives in M^{2}S learning, the training process under super-hard negatives is reasonably smooth, as shown in Fig.9.

6.6 Impact of Training Set Scale

We next conduct another experiment to demonstrate the necessity of discarding the MSE loss: we train the MSE model and our proposed M^{2}S-GAN using different scales of the training set. Specifically, the full training set of ConductorMotion100 is 90 hours; here, we train these two models with {1, 4, 8, 16, 32, 64, 90} hours, and compare their performances. The results are shown in Fig.10. From the training of the MSE model (blue dotted line) we can see that under a very small training set, the MSE model can fit the data very well. As the training set scale increases, however, the capacity of the MSE model becomes incapable of memorizing so much data, meaning that the MSE begins to rise. At the same time, SDP drops dramatically, showing that the model is shrinking the motion amplitude. Comparatively, the SDP of our M^{2}S-GAN (red lines) does not change with the training set scale, remaining stable at around 100%. Since M^{2}S-GAN does not seek to regress the ground truth motion, it has a much higher MSE than the MSE model.

Table 3. Performance of M^{2}S Learning with Different Types of Negatives

Negative Pair	Training Accuracy (%)	Testing Accuracy		
		Easy (%)	Hard (%)	Super-Hard (%)
Easy	75.14	67.56	57.90	53.01
Hard	68.79	**72.60**	**67.83**	**62.03**
Super-hard	61.79	65.71	63.53	61.27

7 Discussions

Unlike most previously published rule-based music-driven conducting motion methods, our approach requires no prior knowledge of orchestra conducting. The

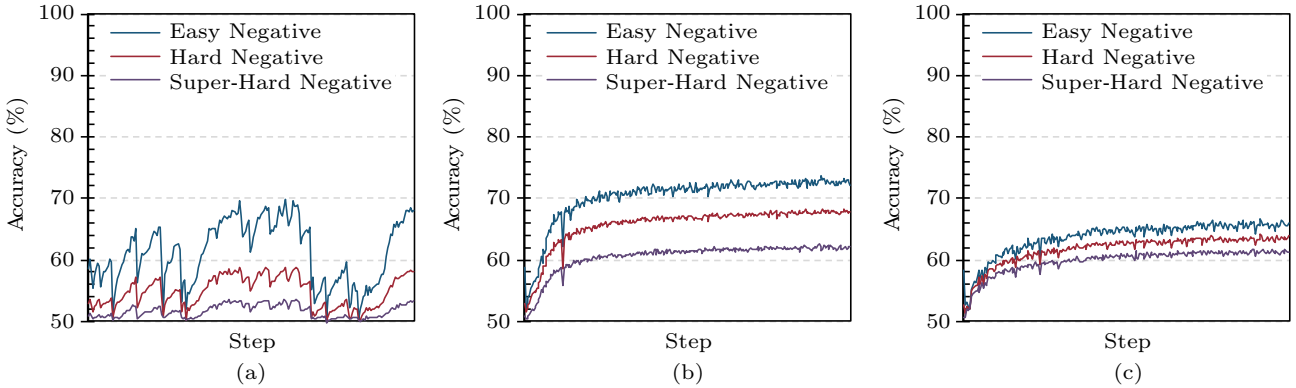


Fig.9. Change of testing accuracy during the training under (a) easy, (b) hard and (c) super-hard negative sampling.

[7]https://github.com/ChenDelong1999/VirtualConductor, Mar. 2022.

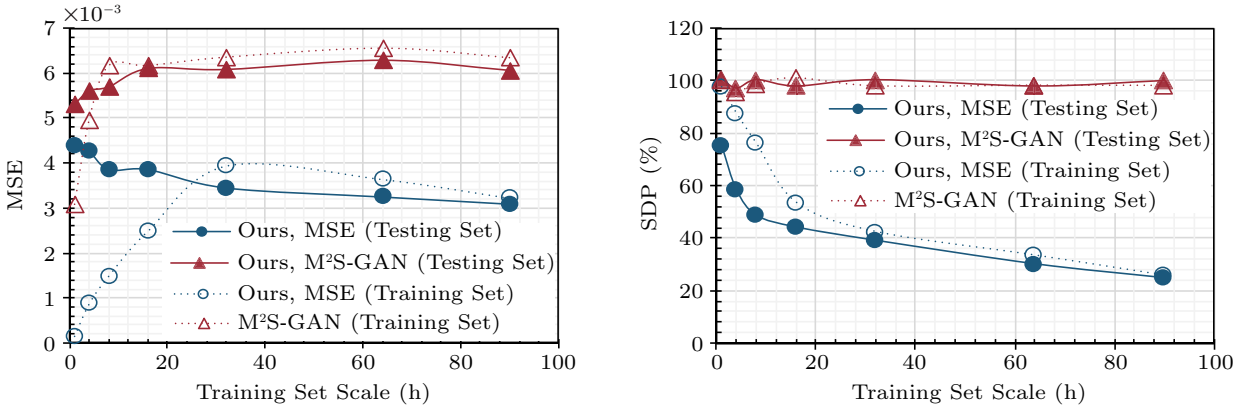


Fig.10. Change of MSE and SDP across different training set scales.

knowledge of the music motion relationship is learned from the large dataset by the model itself. As a result, our approach is highly extendable. We will continue to explore the potential of applying our approach to other tasks (e.g., dance generation, instrument playing motion generation, and talking head generation) in the future.

We further note that our proposed M^{2}S learning task in the contrastive learning stage is the first attempt to apply multimodal self-supervised learning to music and motion. Importantly, we discover that the "easy negatives" that work well in the video domain are not suitable for learning music motion synchronization. During our experiments, we find that the "easy negatives" make the model training process unstable, a finding that we attribute to the increased chance of false negative sampling. However, the exact boundary between the "easy negatives" that hinder model convergence and the "hard negatives" that do not have this problem remains unclear. We leave the investigation of this point to future work.

In addition, we construct a large-scale conducting motion dataset, ConductorMotion100, which contains an unprecedented 100 hours of conducting motion and corresponding music data. The ConductorMotion100 dataset enables M^{2}S-GAN to learn rich music semantics. Since the scale of ConductorMotion100 is also larger than many datasets for music information retrieval (MIR) tasks, in future, we will also validate the effectiveness of using it as a pretraining dataset for MIR tasks, such as beat tracking and tempo estimation.

8 Conclusions

In this paper, we revisited the task of music-driven conducting motion generation, which has received minimal research attention over the years. In a departure from most previous methods, we proposed a deep learning based method that can automatically learn the temporal relationship between the music and the motion, removing the need to rely on human-designed rules. To avoid using regression loss, which leads to over-smoothed results, we designed a two-stage framework consisting of a contrastive learning stage and a generative learning stage. An M^{2}S-Net is first trained with a self-supervised loss, and then an M^{2}S-GAN is trained with adversarial loss and a proposed sync loss, which measure the realism and the perceptual similarity between the real motion and the generated motion respectively. On our collected ConductorMotion100 dataset, the proposed method achieved 0.049 RDE and 2.046 SCE, outperforming all the compared methods. Through the visualization of generated motion, we demonstrated that our method can generate plausible, diverse, and music-synchronized conducting motion.

References

[1] Ren X, Li H, Huang Z, Chen Q. Self-supervised dance video synthesis conditioned on music. In *Proc. the 28th ACM International Conference on Multimedia*, October 2020, pp.46-54. DOI: 10.1145/3394171.3413932.

[2] Lee H, Yang X, Liu M, Wang T, Lu Y, Yang M, Kautz J. Dancing to music. In *Proc. the Annual Conference on Neural Information Processing Systems*, December 2019, pp.3581-3591.

[3] Li B, Maezawa A, Duan Z. Skeleton plays piano: Online generation of pianist body movements from MIDI performance. In *Proc. the 19th International Society for Music Information Retrieval Conference*, September 2018, pp.218-224.

[4] Kao H, Su L. Temporally guided music-to-body-movement generation. In *Proc. the 28th ACM International Confe-*

rence on Multimedia, October 2020, pp.147-155. DOI: 10.1145/3394171.3413848.

[5] Ruttkay Z, Huang Z, Eliens A. The conductor: Gestures for embodied agents with logic programming. In *Proc. the Joint Annual ERCIM/CoLogNet International Workshop on Constraint and Logic Programming*, June 30-July 2, 2003, pp.9-16. DOI: 10.1007/978-3-540-24662-6_15.

[6] Bos P, Reidsma D, Ruttkay Z, Nijholt A. Interacting with a virtual conductor. In *Proc. the 5th International Conference on Entertainment Computing*, September 2006, pp.25-30. DOI: 10.1007/11872320_3.

[7] Nijholt A, Reidsma D, Ebbers R, Maat M. The virtual conductor: Learning and teaching about music, performing, and conducting. In *Proc. the 8th IEEE International Conference on Advanced Learning Technologies*, July 2008, pp.897-899. DOI: 10.1109/ICALT.2008.43.

[8] Maat M, Ebbers R, Reidsma D, Nijholt A. Beyond the beat: Modelling intentions in a virtual conductor. In *Proc. the 2nd International Conference on Intelligent Technologies for Interactive Entertainment*, January 2008, Article No. 12. DOI: 10.4108/ICST.INTETAIN2008.2489.

[9] Reidsma D, Nijholt A, Bos P. Temporal interaction between an artificial orchestra conductor and human musicians. *Comput. Entertain.*, 2008, 6(4): Article No. 53. DOI: 10.1145/1461999.1462005.

[10] Takatsu R, Maki Y, Inoue T, Okada K, Shigeno H. Multiple virtual conductors allow amateur orchestra players to perform better and more easily. In *Proc. the 20th IEEE International Conference on Computer Supported Cooperative Work in Design*, May 2016, pp.486-491. DOI: 10.1109/CSCWD.2016.7566038.

[11] Katayama N, Takatsu R, Inoue T, Shigeno H, Okada K. Efficient generation of conductor avatars for the concert by multiple virtual conductors. In *Proc. the 8th International Conference on Collaboration Technologies and Social Computing*, Sept. 2016, pp.45-57. DOI: 10.1007/978-981-10-2618-8_4.

[12] Wang T, Zheng N, Li Y, Xu Y, Shum H. Learning kernel-based HMMs for dynamic sequence synthesis. *Graph. Model.*, 2003, 65(4): 206-221. DOI: 10.1016/S1524-0703(03)00040-7.

[13] Shu X, Qi G, Tang J, Wang J. Weakly-shared deep transfer networks for heterogeneous-domain knowledge propagation. In *Proc. the 23rd Annual ACM Conference on Multimedia*, October 2015, pp.35-44. DOI: 10.1145/2733373.2806216.

[14] Tang J, Shu X, Qi G, Li Z, Wang M, Yan S, Jain R C. Tri-clustered tensor completion for social-aware image tag refinement. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017, 39(8): 1662-1674. DOI: 10.1109/TPAMI.2016.2608882.

[15] Tang J, Shu X, Li Z, Jiang Y, Tian Q. Social anchor-unit graph regularized tensor completion for large-scale image retagging. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019, 41(8): 2027-2034. DOI: 10.1109/TPAMI.2019.2906603.

[16] Du X, Yang Y, Yang L, Shen F, Qin Z, Tang J. Captioning videos using large-scale image corpus. *J. Comput. Sci. Technol.*, 2017, 32(3): 480-493. DOI: 10.1007/s11390-017-1738-7.

[17] Korbar B, Tran D, Torresani L. Cooperative learning of audio and video models from self-supervised synchronization. In *Proc. the Annual Conference on Neural Information Processing Systems*, December 2018, pp.7774-7785.

[18] Arjovsky M, Chintala S, Bottou L. Wasserstein GAN. arXiv:1701.07875, 2017. https://arxiv.org/pdf/1701.07875.pdf, Dec. 2021.

[19] Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville A C. Improved training of Wasserstein GANs. In *Proc. the Annual Conference on Neural Information Processing Systems*, December 2017, pp.5767-5777.

[20] Redmon J, Farhadi A. YOLOv3: An incremental improvement. arXiv:1804.02767, 2018. https://arxiv.org/abs/1804.02767, Dec. 2021.

[21] Fang H, Xie S, Tai Y, Lu C. RMPE: Regional multi-person pose estimation. In *Proc. the 2017 IEEE International Conference on Computer Vision*, October 2017, pp.2353-2362. DOI: 10.1109/ICCV.2017.256.

[22] Geuther B, Breese A, Wang Y. A study on musical conducting robots and their users. In *Proc. the 10th IEEE-RAS International Conference on Humanoid Robots*, December 2010, pp.124-129. DOI: 10.1109/ICHR.2010.5686302.

[23] Salgian A, Ault C, Nakra T M, Wang Y, Stone M. Multidisciplinary computer science through conducting robots. In *Proc. the 42nd ACM Technical Symposium on Computer Science Education*, March 2011, pp.219-224. DOI: 10.1145/1953163.1953229.

[24] Salgian A, Ault C, Nakra T M, Wang Y, Stone M. A theory of 'multiple creativities': Outcomes from an undergraduate seminar in conducting robots. In *Proc. the Music, Mind, and Invention Workshop*, March 2012.

[25] Dansereau D G, Brock N, Cooperstock J R. Predicting an orchestral conductor's baton movements using machine learning. *Comput. Music. J.*, 2013, 37(2): 28-45. DOI: 10.1162/COMJ_a_00173.

[26] Yalta N. Sequential deep learning for dancing motion generation. In *Proc. the 46th AI Challenge Study Group*, November 2016, pp.43-49.

[27] Li Z, Liu F, Yang W, Peng S, Zhou J. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*. DOI: 10.1109/TNNLS.2021.3084827.

[28] Yalta N, Watanabe S, Nakadai K, Ogata T. Weakly-supervised deep recurrent neural networks for basic dance step generation. In *Proc. the 2019 International Joint Conference on Neural Networks*, July 2019. DOI: 10.1109/IJCNN.2019.8851872.

[29] Tang T, Jia J, Mao H. Dance with melody: An LSTM-autoencoder approach to music-oriented dance synthesis. In *Proc. the 2018 ACM Multimedia Conference on Multimedia*, October 2018, pp.1598-1606. DOI: 10.1145/3240508.3240526.

[30] Bogaers A, Yumak Z, Volk A. Music-driven animation generation of expressive musical gestures. In *Proc. the 2020 International Conference on Multimodal Interaction*, October 2020, pp.22-26. DOI: 10.1145/3395035.3425244.

[31] Qi Y, Liu Y, Sun Q. Music-driven dance generation. *IEEE Access*, 2019, 7: 166540-166550. DOI: 10.1109/ACCESS.2019.2953698.

[32] Shlizerman E, Dery L M, Schoen H, Kemelmacher-Shlizerman I. Audio to body dynamics. In *Proc. the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp.7574-7583. DOI: 10.1109/CVPR.2018.00790.

[33] Haag K, Shimodaira H. Bidirectional LSTM networks employing stacked bottleneck features for expressive speech-driven head motion synthesis. In *Proc. the 16th International Conference on Intelligent Virtual Agents*, September 2016, pp.198-207. DOI: 10.1007/978-3-319-47665-0_18.

[34] Ferstl Y, McDonnell R. Investigating the use of recurrent motion modelling for speech gesture generation. In *Proc. the 18th International Conference on Intelligent Virtual Agents*, November 2018, pp.93-98. DOI: 10.1145/3267851.3267898.

[35] Sadoughi N, Busso C. Joint learning of speech-driven facial motion with bidirectional long-short term memory. In *Proc. the 17th International Conference on Intelligent Virtual Agents*, August 2017, pp.389-402. DOI: 10.1007/978-3-319-67401-8_49.

[36] Huang R, Hu H, Wu W, Sawada K, Zhang M, Jiang D. Dance revolution: Long-term dance generation with music via curriculum learning. In *Proc. the 9th International Conference on Learning Representations*, May 2021.

[37] Sun G, Wong Y, Cheng Z, Kankanhalli M S, Geng W, Li X. DeepDance: Music-to-dance motion choreography with adversarial learning. *IEEE Trans. Multim.*, 2020, 23: 497-509. DOI: 10.1109/TMM.2020.2981989.

[38] Ahn H, Kim J, Kim K, Oh S. Generative autoregressive networks for 3D dancing move synthesis from music. *IEEE Robotics and Automation Letters*, 2020, 5(2): 3501-3508. DOI: 10.1109/LRA.2020.2977333.

[39] Lee J, Kim S, Lee K. Automatic choreography generation with convolutional encoder-decoder network. In *Proc. the 20th International Society for Music Information Retrieval Conference*, November 2019, pp.894-899. DOI: 10.5281/zenodo.3527958.

[40] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser L, Polosukhin I. Attention is all you need. In *Proc. the Annual Conference on Neural Information Processing Systems*, December 2017, pp.5998-6008.

[41] Li R, Yang S, Ross D A, Kanazawa A. Learn to dance with AIST++: Music conditioned 3D dance generation. arXiv:2101.08779, 2021. https://arxiv.org/abs/2101.08779, Dec. 2021.

[42] Goodfellow I J, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A C, Bengio Y. Generative adversarial nets. In *Proc. the Annual Conference on Neural Information Processing Systems*, December 2014, pp.2672-2680.

[43] Ginosar S, Bar A, Kohavi G, Chan C, Owens A, Malik J. Learning individual styles of conversational gesture. In *Proc. the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2019, pp.3497-3506. DOI: 10.1109/CVPR.2019.00361.

[44] Eskimez S E, Maddox R K, Xu C, Duan Z. End-to-end generation of talking faces from noisy speech. In *Proc. the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2020, pp.1948-1952. DOI: 10.1109/ICASSP40776.2020.9054103.

[45] Song Y, Zhu J, Li D, Wang A, Qi H. Talking face generation by conditional recurrent adversarial network. In *Proc. the 28th International Joint Conference on Artificial Intelligence*, August 2019, pp.919-925.

[46] Sadoughi N, Busso C. Novel realizations of speech-driven head movements with generative adversarial networks. In *Proc. the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, April 2018, pp.6169-6173. DOI: 10.1109/ICASSP.2018.8461967.

[47] Ferstl Y, Neff M, McDonnell R. Multi-objective adversarial gesture generation. In *Proc. the Motion, Interaction and Games*, October 2019, Article No. 3. DOI: 10.1145/3359566.3360053.

[48] Sarasúa Á. Context-aware gesture recognition in classical music conducting. In *Proc. the 21st ACM International Conference on Multimedia*, October 2013, pp.1059-1062. DOI: 10.1145/2502081.2502216.

[49] Sarasúa Á, Guaus E. Beat tracking from conducting gestural data: A multi-subject study. In *Proc. the International Workshop on Movement and Computing*, June 2014, pp.118-123. DOI: 10.1145/2617995.2618016.

[50] Karipidou K, Ahnlund J, Friberg A, Alexanderson S, Kjellström H. Computer analysis of sentiment interpretation in musical conducting. In *Proc. the 12th IEEE International Conference on Automatic Face & Gesture Recognition*, May 30-June 3, 2017, pp.400-405. DOI: 10.1109/FG.2017.57.

[51] Huang Y, Chen T, Moran N, Coleman S, Su L. Identifying expressive semantics in orchestral conducting kinematics. In *Proc. the 20th International Society for Music Information Retrieval Conference*, November 2019, pp.115-122. DOI: 10.5281/zenodo.3527753.

[52] Lemouton S, Borghesi R, Haapamäki S, Bevilacqua F, Fléty E. Following orchestra conductors: The IDEA open movement dataset. In *Proc. the 6th International Conference on Movement and Computing*, October 2019, Article No. 25. DOI: 10.1145/3347122.3359599.

[53] Yan S, Xiong Y, Lin D. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proc. the 32nd AAAI Conference on Artificial Intelligence*, February 2018, pp.7444-7452.

[54] Bai S, Kolter J Z, Koltun V. Convolutional sequence modeling revisited. In *Proc. the 6th International Conference on Learning Representations*, April 30-May 3, 2018.

[55] Arandjelovic R, Zisserman A. Look, listen and learn. In *Proc. the 2017 IEEE International Conference on Computer Vision*, October 2017, pp.609-617. DOI: 10.1109/ICCV.2017.73.

[56] Chung J S, Zisserman A. Out of time: Automated lip sync in the wild. In *Proc. the 2016 ACCV International Workshops on Computer Vision*, November 2016, pp.251-263. DOI: 10.1007/978-3-319-54427-4_19.

[57] Chen L, Srivastava S, Duan Z, Xu C. Deep cross-modal audio-visual generation. In *Proc. the Thematic Workshops of the 2017 ACM Multimedia*, October 2017, pp.349-357. DOI: 10.1145/3126686.3126723.

[58] Hao W, Zhang Z, Guan H. CMCGAN: A uniform framework for cross-modal visual-audio mutual generation. In *Proc. the 32nd AAAI Conference on Artificial Intelligence*, February 2018, pp.6886-6893.

[59] Zhou H, Liu Z, Xu X, Luo P, Wang X. Vision-infused deep audio inpainting. In *Proc. the 2019 IEEE/CVF International Conference on Computer Vision*, October 27-November 2, 2019, pp.283-292. DOI: 10.1109/ICCV.2019.00037.

[60] Choi H, Park C, Lee K. From inference to generation: End-to-end fully self-supervised generation of human face from speech. In *Proc. the 8th International Conference on Learning Representations*, April 2020.

[61] Johnson J, Alahi A, Li F F. Perceptual losses for real-time style transfer and super-resolution. In *Proc. the 14th European Conference on Computer Vision*, October 2016, pp.694-711. DOI: 10.1007/978-3-319-46475-6_43.

[62] Li M, Hsu W, Xie X, Cong J, Gao W. SACNN: Self-attention convolutional neural network for low-dose CT denoising with self-supervised perceptual loss network. *IEEE Trans. Medical Imaging*, 2020, 39(7): 2289-2301. DOI: 10.1109/TMI.2020.2968472.

[63] Akella R T, Halder S S, Shandeelya A P, Pankajakshan V. Enhancing perceptual loss with adversarial feature matching for super-resolution. In *Proc. the 2020 International Joint Conference on Neural Networks*, July 2020. DOI: 10.1109/IJCNN48605.2020.9207102.

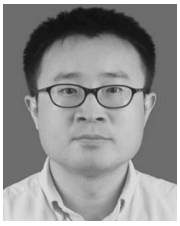
[64] Tieleman T, Hinton G. Lecture 6.5-rmsprop, COURSERA: Neural networks for machine learning. Technical Report, University of Toronto, 2012.

[65] Diederik P K, Jimmy B. Adam: A method for stochastic optimization. In *Proc. the 3rd International Conference on Learning Representations*, May 2015.

[66] Sarasúa Á, Caramiaux B, Tanaka A. Machine learning of personal gesture variation in music conducting. In *Proc. the 2016 CHI Conference on Human Factors in Computing Systems*, May 2016, pp.3428-3432. DOI: 10.1145/2858036.2858328.

[67] Cosentino S, Petersen K, Lin Z, Bartolomeo L, Sessa S, Zecca M, Takanishi A. Natural human-robot musical interaction: Understanding the music conductor gestures by using the WB-4 inertial measurement system. *Adv. Robotics*, 2014, 28(11): 781-792. DOI: 10.1080/01691864.2014.889577.

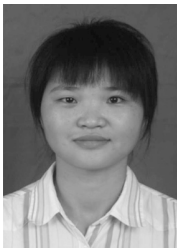
[68] Lee K, Junokas M J, Amanzadeh M, Garnett G E. An analysis of basic expressive qualities in instrumental conducting. In *Proc. the 2nd International Workshop on Movement and Computing*, August 2015, pp.148-155. DOI: 10.1145/2790994.2791005.



Fan Liu is currently a professor of Hohai University, Nanjing. He received his B.S. degree in networking and Ph.D. degree in technology for computer applications from Nanjing University of Science and Technology (NUST), Nanjing, in 2009 and 2015, respectively. From September 2008 to December 2008, he studied at Ajou University, Suwon City. From February 2014 to May 2014, he worked at Microsoft Research Asia, Beijing. His research interests include computer vision, pattern recognition, and machine learning. Dr. Liu also serves as a reviewer of IEEE TNNLS, IEEE TKDE, ACM TIST, Information Sciences, Neurocomputing, Pattern Analysis and Application.

De-Long Chen received his B.S. degree in computer science in Hohai University, Nanjing, in 2021. He is currently a research assistant in Hohai University, Nanjing, and a research intern at MEGVII Technology, Beijing. His research includes computer vision, music information retrieval, multimodal learning, unsupervised learning and self-supervised learning.

Rui-Zhi Zhou is currently pursuing his B.S. degree in computer science in Hohai University, Nanjing. His current research interests include deep learning, pattern recognition, music information retrieval, human motion analysis and generation.



Sai Yang is currently a lecturer of Nantong University, Nantong. She received her M.S. degree in control theory and control engineering from School of Mechanical and Electrical Engineering, Jiangxi University of Science and Technology, Nanchang, in 2010, and her Ph.D. degree in pattern recognition and intelligent system from School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, in 2015. Her research interests include computer vision, image processing, pattern recognition and machine learning.

Feng Xu is currently a professor at Hohai University, Nanjing. He received his Ph.D. degree in software theory from Nanjing University, Nanjing, in 2008. He received his B.S. and M.S. degrees in technology for computer applications from Hohai University, Nanjing, in 1998 and 2001, respectively. His research interests include cloud computing, network information security, domain software engineering, etc. He has authored over 100 journal and conference papers in these areas.