# Taming Diffusion Models for Music-Conditioned Conductor Motion Generation

**Jinbin Bai  Zhuoran Zhao  Wang Debang  Yubo Pan**
School of Computing
National University of Singapore
{jinbin.bai, zhuoran.zhao, debang, e0954779}@u.nus.edu

## Abstract

Conditional motion generation is a series of tasks that aims to generate realistic and plausible human body animations in response to a given prompt. And it has witnessed an explosion of methods and architectures to solve this task. The existing methods mainly rely on generative adversarial networks (GANs), which typically suffer from notorious mode collapse and unstable training, thus making it difficult to learn accurate motion representation. Moreover, only a few prior studies have focused on the generation of conductor motion. In this paper, we take a step towards constructing a music conditional conductor motion generation framework, by introducing a contrastive pre-training before the controllable diffusion model. Extensive experiments are conducted to improve the generation performance. Codes are available at `https://github.com/JB-Bai/DM_for_Music_Conditioned_Motion_Generation`

## 1 Introduction

In CS5340, we learnt a lot about probabilistic graphical models. Probabilistic graphical models (PGMs) are a family of powerful machine learning techniques that combine probability theory and graphical models to represent complex relationships among the variables in the model. PGMs have been used for many applications ranging from natural language processing, to computer vision, and robotics. At their core, PGMs are a type of directed or undirected graph, where nodes represent variables and edges represent dependencies. PGMs also provide a way to capture probabilistic relationships between variables and to quantify uncertainty in decision-making processes. By incorporating prior knowledge into the model, PGMs enable us to make reliable predictions about future outcomes based on past occurrences.

Meanwhile, diffusion models [1, 2] have emerged as the new state-of-the-art family of deep generative models. And they have shown potential in a variety of domains, ranging of computer vision, natural language processing, and acoustics signal processing. Popular examples include GLIDE [3] and DALL-E 2 [4] by OpenAI, Latent Diffusion [5] by the University of Heidelberg, ImageGen [6] by Google Brain and Stable Diffusion [7] by Stability AI.

Conditional motion generation [8] is a series of tasks that generates realistic and plausible human body animation in response to a given prompt. And it has witnessed an explosion of methods and architectures to solve this task in computer vision and graphics, be it 2D [9] or 3D [10], music guided or text guided, dance motions or instrument-playing motions. The paradigm of uncertainty models, like Variational Auto-Encoder (VAE) [11] and Generative Adversarial Network (GAN) [12], has shown promising success with human annotated datasets, after aligning the prompt representation and motion representation. However, only a few prior studies have focused on the generation of conductor's motions. And the existing methods mainly rely on generative adversarial networks

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

(GANs), which typically suffer from notorious mode collapse and unstable training, thus making it difficult to learn accurate motion representation.

In this paper, we take a step towards constructing a music conditional conductor motion generation framework, by introducing a contrastive pre-training before the controllable diffusion model. And extensive experiments are conducted to improve the generation performance.

In summary, our main contributions are as follows:

- Our project is the first work to use diffusion model for music-conditional conductor motion generation.
- We propose a new L2 constraint of $x_0$ instead of $\epsilon$ to achieve a better performance on generating conductor motion.

## 2 Related Work

### 2.1 Diffusion Model

Diffusion models have emerged as the new state-of-the-art family of deep generative models. DDPM [1], DDIM [13], Classifier-Guidance [14] and Classifier-Free [2]. Algorithm 1 and Alogrithm 2 reveal how to train and inference with denoising diffusion probabilistic models. And they have shown potential in a variety of domains, ranging of computer vision, natural language processing, and acoustics signal processing. Popular examples include GLIDE [3] and DALL-E 2 [4] by OpenAI, Latent Diffusion [5] by the University of Heidelberg, ImageGen [6] by Google Brain and Stable Diffusion [7] by Stability AI.

Diffusion models have been widely explored in computer vision applications, but there is still limited work on diffusion models in motion applications. To the best of our knowledge, we are the first to use diffusion model for music-conditional conductor motion generation. We believe that this approach could be a promising direction for future research in the application of diffusion models to motions.

**Algorithm 1 Training**

**repeat**
  $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$
  $t \sim \mathrm{Uniform}(\{1, \dots, T\})$
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
  Take gradient descent step on $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
**until** converged

**Algorithm 2 Sampling**

Trained diffusion model $\theta$, $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
**for** $t = T, \dots, 1$ **do**
  $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ if $t > 1$, else $\boldsymbol{z} = \mathbf{0}$
  $\boldsymbol{x_{t-1}} = \frac{1}{\sqrt{\alpha_t}} \left( \boldsymbol{x_t} - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \hat{\boldsymbol{\epsilon}}_\theta \right) + \sigma_t \boldsymbol{z}$
**end for**
**return** $\boldsymbol{x}_0$

### 2.2 Self-supervised Learning

As the scale of training datasets increases, the acquisition of labeled datasets becomes increasingly difficult, prompting the emergence of the self-supervised paradigm. Self-supervised learning mainly utilizes auxiliary tasks (pretexts) to extract self-supervision information from large-scale unlabeled data, and by constructing this supervision information, the network can be trained to learn representations that are valuable for downstream tasks.

In the field of natural language processing, works such as Bert [15] and GPT [16, 17, 18] have demonstrated the success of this self-supervised paradigm. In the field of image processing, self-supervised learning has been further extended through data augmentation works such as SimCLR [19],

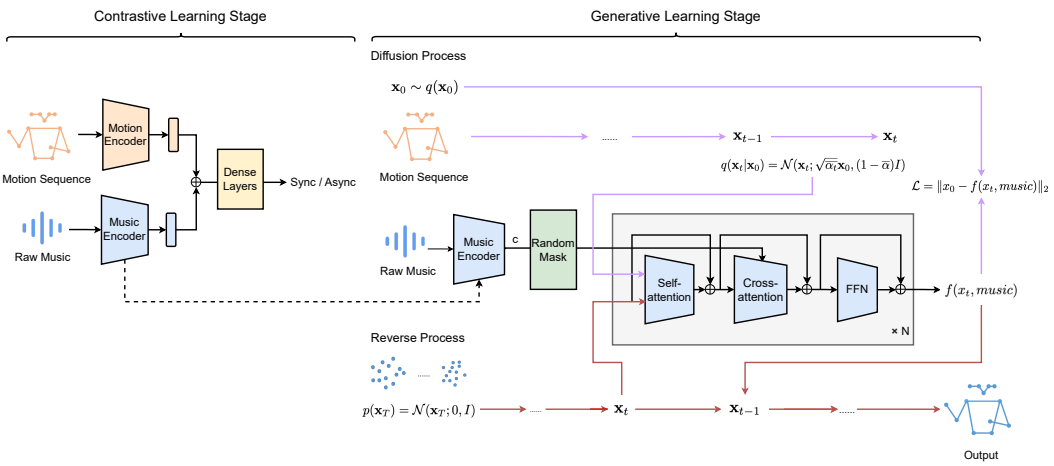


Figure 1: Overview of the proposed framework. The colors of the arrows in Generative Learning Stage represent different stages: purple for training, red for inference, and black for both training and inference.

contrastive learning works such as CLIP [17], and MAE [20] which borrows the idea from the NLP domain to mask parts of images for restoration, etc. In our project, due to the lack of pre-trained general music representations models, we adopted a self-supervised approach to learn a music encoder.

# 3 Method

In this section, we will first present the definition of our problem, then provide an overall illustration of our method, as well as the loss functions used in different frameworks.

## 3.1 Problem Definition

Conditional motion generation is a series of tasks that generates realistic and plausible human body motions $\mathbf{Y}$ with specified actions given specific prompt. The motion sequence $\mathbf{Y}$ is an array of poses $[\mathbf{y}_i], i \in \{1, 2, 3, \ldots, F\}$, where $\mathbf{y}_i \in \mathbb{R}^D$ represents the pose state in the $i$-th frame, and $F$ is the number of frames. In music-conditional motion generation, the specified prompt is the music feature $E_{music}(\mathbf{M})$ extracted from raw music $\mathbf{M}$. Our goal is to learn a diffusion network $G$, which can generate the motion sequence $\mathbf{Y}$ corresponding to given $E_{music}(\mathbf{M})$.

## 3.2 Overview of Our Framework

Our proposed architecture is illustrated in Fig. 1. In the contrastive learning stage, a contrastive pre-training network with a motion encoder $E_{motion}$ and a music encoder $E_{music}$, is used to learn the music representations that are aligned to their corresponding motion representations. In the generative learning stage, a generation network $G$ is used to generate a motion sequence based on the music embeddings outputted by the pre-trained $E_{music}$. Employing the denoising diffusion probabilistic model (DDPM) [1], a Cross-Modality Linear Transformer is introduced to facilitate motion generation while undergoing the denoising process. Our methods are further elaborated upon in the following sections. During inference, a Gaussian distribution noise is sampled according to the given random seed and passed into the denoising process with cross-attention between the music features. Finally, music-conditioned conductor motions will be generated.

### 3.3 Contrastive pre-training

Contrastive pre-training network consists of three parts: Motion Encoder $E_{motion}$, Music Encoder $E_{music}$, and Dense Layers $D$. The motion embeddings and music embeddings output by $E_{motion}$ and $E_{music}$ are concatenated together and passed to $D$. Then, a binary cross-entropy loss is used to calculate whether music and motion are paired with each other.

**Music Encoder**. $E_{music}$ is used to generate music features from the raw music. $E_{music}$ is composed of three groups of layers, where each layer has 3 residual layers and a max-pooling layer.

**Motion Encoder**. $E_{motion}$ is used to generate motion features for the conducting motion sequence. In order to analyze the conducting motion both spatially and temporally, we employ the Spatial-Temporal Graph Convolutional Network (ST-GCN) [21], which has been widely used in human pose estimation tasks.

### 3.4 Diffusion model for Motion Generation

Diffusion models generally have a diffusion process and a reverse process. In the diffusion process, a Gaussian noise is gradually added to the motion sequence data following the Markov chain rule to approximate the posterior $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$. At the end of the diffusion process, the data distribution $\mathbf{x}_T$ should be equivalent to an isotropic Gaussian distribution:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t-1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t I)$$

Using reparameterization trick, we can sample $\mathbf{x}_t$ at any arbitrary time step $t$ in a closed form:

$$q(\mathbf{x}_t|\mathbf{x}_0) = \sqrt{\overline{\alpha_t}}\mathbf{x}_0 + \epsilon\sqrt{1-\overline{\alpha_t}}, \epsilon \sim \mathcal{N}(0, I)$$

In order to run the reverse process, we need to learn a model $p_\theta$ to approximate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ since $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is intractable:

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t-1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$$

Like [3], we train a model to fit $\epsilon_\theta(\mathbf{x}_t, t, E_{music}(M))$. Then, we calculate the mean square error between the noise term $\epsilon$ and $\epsilon_\theta(\mathbf{x}_t, t, E_{music}(M))$ to optimize the model.

Therefore, we can run the reverse process to denoise the motion sequence step by step and generate a clean motion sequence conditioned on the given music embeddings.

### 3.5 Random Mask

Inspired by masked language modeling and masked image modeling, we adopt a ramdom mask [22] block after music encoder for achieving better generalization performance.

### 3.6 Cross-Modality Linear Transformer

We propose to use Transformer [23] as the denoising model. We first use a music encoder to extract the music embeddings. Since the music encoder has been pre-trained in the contrastive learning stage, which can facilitate the generation process. Then, we employ a self-attention module to enable motion features from different times to incorporate with each other. Also, a cross-attention module is used to fuse the music embeddings and motion sequence together and a feed-forward network is used to output the predicted $\epsilon_\theta(\mathbf{x}_t, t, E_{music}(M))$.

### 3.7 Training Objective

At the first stage, we adopt a binary cross-entropy loss to learn the representation of music under the supervision of motion, which can be formulated as:

$$\mathcal{L}_{bce} = \sum_{i,j=1}^{N} \Big( c_{ij} log_2(f[E_{music}(\mathbf{M}_i) \oplus E_{motion}(\mathbf{Y}_j)]) + (1 - c_{ij}) log_2(1 - f[E_{music}(\mathbf{M}_i) \oplus E_{motion}(\mathbf{Y}_j)]) \Big)$$

where $c_{ij}$ is defined by

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

$\mathbf{M}_i, \mathbf{Y}_j$ denotes the $i$-th music data and $j$-th motion data. $\oplus$ denotes the feature concatenation operation. $E_{music}, E_{motion}$ denote the music encoder and motion encoder. $f$ denotes the dense layer.

At the second stage, we optimize the generation model by minimizing the variational lower bound on negative log likelihood: $\mathbb{E}[-\log \mathrm{p}_\theta(\mathbf{x}_0)] \leq \mathbb{E}_\mathrm{q}[-\log \frac{\mathrm{p}_\theta(\mathbf{x}_0)}{\mathrm{q}(\mathbf{x}_{1:\mathrm{T}}|\mathbf{x}_0)}]$. Then we eliminate the constant terms and formulate the loss function into: $L(\theta) = \mathbb{E}_q[\Sigma_{t=2}^{T} D_{KL}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t))]$. We follow [1] to simplify the loss function to MSE loss, which can be formulated as:

$$\mathcal{L}_{ddpm} = ||\epsilon - \epsilon_\theta(\mathbf{x}_t, t, E_{music}(M))||_2^2$$

where $\epsilon$ denotes the noise (here we use $\epsilon$ to refer $q(\mathbf{x}_{t-1}|\mathbf{x}_t, x_0)$), and $\epsilon_\theta(\mathbf{x}_t, t, E_{music}(M))$ denotes the noise predicted by the model, and $\mathbf{x}_i (i = 0, 1, \ldots, t)$ denotes $i$-th step of motion sequence.

For better generation performance, we follow [4] to predict the motion itself instead of predicting $\epsilon$ as formulated by [1]. The final loss can be represented by:

$$\mathcal{L}_{ddpm}^{'} = ||x_0 - f(\mathbf{x}_t, t, E_{music}(M))||_2^2$$

where $x_0$ denotes the original motion sequences, and $f(\mathbf{x}_t, t, E_{music}(M)$ denotes the final step of motion sequence predicted by the model.

The generative model is further regularized using geometric loss. This loss enforces physical properties and prevents artifacts, encouraging natural and coherent motion. Here, we add geometric restriction $L_{body}$ on the acceleration of body keypoints to prevent them from fluctuation.

$$\mathcal{L}_{body} = \frac{1}{N-1} \sum_{i=1}^{N-1} ||\hat{x_0}_{body}^{i+1} - \hat{x_0}_{body}^{i}||_2^2$$

Finally, the overall loss of our training objective is weighted sum of $\mathcal{L}_{ddpm}$ and $\mathcal{L}_{body}$. To minimize the impact of geometric loss on overall loss, we limit it to a range between -0.1 and 0.1.

## 4 Experiment

In this section, we will first provide an overview of the training datasets and evaluation metrics, followed by quantitative and qualitative experiments that are compared with our baseline method. Finally, we will present some examples to demonstrate the effectiveness of our frameworks.

### 4.1 Datasets

We use ConductorMotion100 [9] as our training dataset. The ConductorMotion100 dataset consists of training set, validation set and test set, with respective durations of 90, 5 and 5 hours. 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, ConductorMotion only preserve 13 2D keypoints of the upper body. The perserved keypoints are identical to the first 13 key points in the MS COCO format. All motion data is thus re-sampled to 30 fps, with corresponding music motion encoding at 90 Hz.

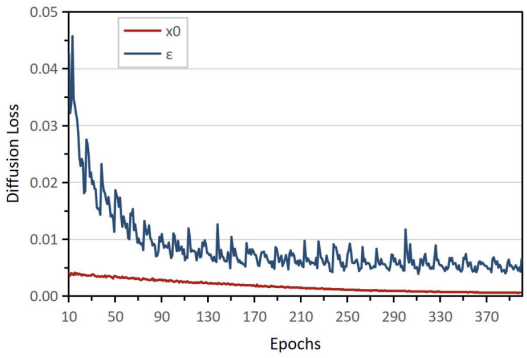


Figure 2: Comparison of diffusion loss for predicting $\epsilon$ and $x_0$.

### 4.2 Evaluation Metrics

As the task of learning music-driven conducting generated motion is relatively new, there are few metrics available to measure its outcomes. Usually, the objective evaluation of deep generative models relies on the Inception Score (IS) or Frechet Inception Distance (FID), both of which require a feature encoder, typically obtained through classification pre-training. Unfortunately, there is no available class label for conducting motions, so we follow [9] to use Mean Square Error (MSE) to evaluate the quality of conducting motions.
MSE has been widely used as an evaluation metric by existing audio-to-motion tasks and is the most direct way to measure how close the generated motion is to the ground-truth motion. The representation of MSE is defined by:

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

where $Y$ is the ground-truth motion and $\hat{Y}$ is the generated motion.

### 4.3 Implementation Details

Our model is implemented by PyTorch. As for diffusion model, we set the diffusion steps to 1000. The optimization is carried out using Adam [24]. The diffusion model is trained with learning rate of 2e-4 and batch size of 32. The number of total training epochs is 400. We set the unconditional rate of random mask to 0.1. The experiments are conducted on two NVIDIA TESLA V100 GPUs.

### 4.4 Main Results on ConductorMotion100 test set

We report the MSE of generated motion and ground-truth motion on ConductorMotion100 test set with 293 test samples.(see table 1)

Table 1: Main results on ConductorMotion100 test set

| Model | MSE | Total Loss |
|---|---|---|
| Ours | 0.02 | 6.25 |

### 4.5 Ablation Study

#### 4.5.1 Comparison of predicting $\epsilon$ and $x_0$

We conduct further study to evaluate the effect of predicting noise term $\epsilon$ and the motion $x_0$ itself. As illustrated in Fig. 2, predicting $x_0$ converges to a much lower error than predicting $\epsilon$ in the training stage. In the inference stage, predicting $\epsilon$ fails to generate plausible motion sequence in longer frames while predicting $x_0$ is able to successfully generate stable and plausible motion sequence (see Fig. 3).

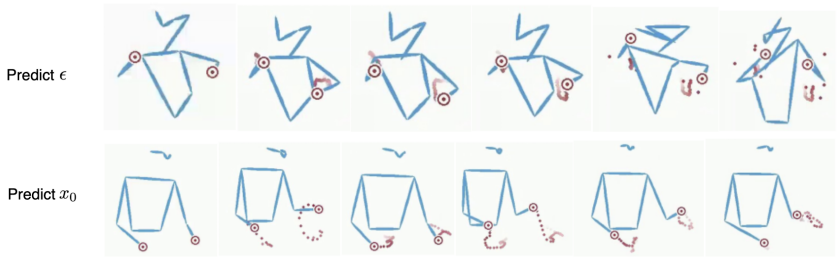


Figure 3: Comparison of generated motion of predicting $\epsilon$ and $x_0$.

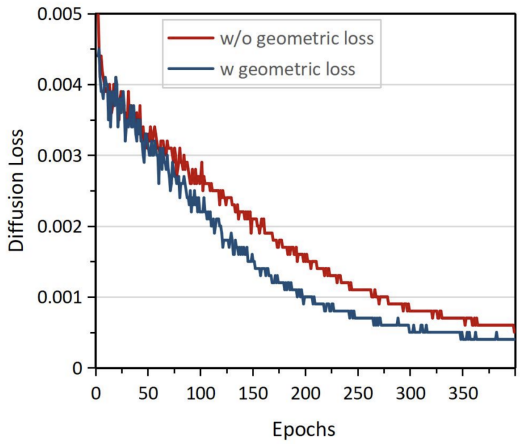


Figure 4: Comparison of diffusion loss with and without geometric loss.

#### 4.5.2 Effect of geometric loss

We analyze the effect of incorporating geometric loss in the training of our model, and compare its performance against model trained without the use of a geometric loss. As illustrated in Fig. 4, model trained with geometric loss converges to a lower error than model trained without geometric loss in the training stage. Also, model trained with geometric loss can achieve lower MSE than model trained without geometric loss on test set (see table 2). It proves that geometric loss helps yielding high-quality motion.

Table 2: Comparison of MSE on ConductorMotion100 test set with and without geometric loss

| Model | Method | MSE | Total Loss |
|---|---|---|---|
| Ours | w geometric loss | 0.02 | 6.25 |
| Ours | w/o geometric loss | 0.74 | 216.63 |

### 4.6 Qualitative Results

We provide some visualization of motion generation conditioned on music which are not included in training or test set. We choose the following symphonies: Rachmaninoff Piano Concerto No.2, Chopin Piano Concerto No.1, Tchaikovsky Piano Concerto No.1, Vivaldi Four Seasons (Spring) (see Fig. 5).

## 5 Discussion for Future Work

There still exists lots of work to be made. And we list some of them.

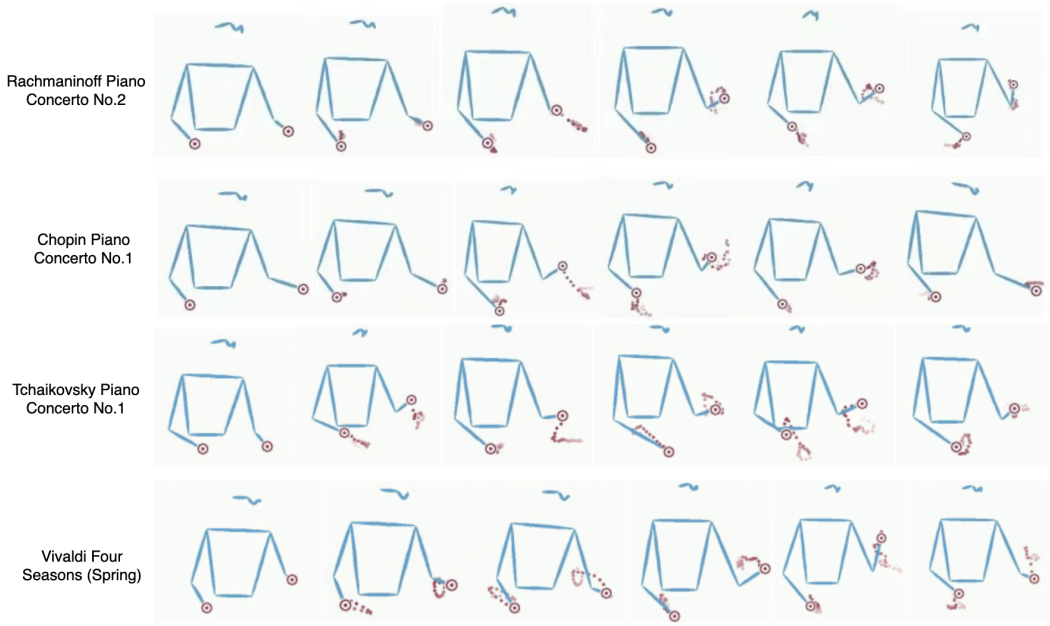


Figure 5: Qualitative results of four symphonies.

### 5.1 Without Dense Layers

CLIP [17] is simple and powerful for multi-modal pre-train, which is also used in [4]. Our current method, in comparison, still uses dense layers for merging embeddings from different modalities. Intuitively, training a music encoder supervised by motion encoder with contrastive loss, without dense layers, will be helpful for our task.

### 5.2 Diffusion on latent space

It is worth noticing that in the contrastive learning stage, we align output of the music encoder with output of the motion encoder, while in generative learning stage, we use the music encoder only. At the same time, multiple works including [4] and [25] apply diffusion on the latent space, which in essence reduces the amount of computation, regularizes generated motion and is potentially easier to learn by utilizing the constrastive learning stage better.

### 5.3 Conditional-unconditional training

As is used in [4] and [26], training the diffusion model with a mixture of music-conditional and unconditional could potentially allows us to trade off between the diversity and quality. A conditional diffusion goal is to train on $\mathcal{L} = ||x_0 - f(\mathbf{x}_t, t, E_{music}(M))||_2^2$, while an unconditional goal is to train on $\mathcal{L} = ||x_0 - f(\mathbf{x}_t, t, \emptyset)||_2^2$, where the music embedding will be replaced with zeros in practice.

### 5.4 Additional geometric loss

Incorporate additional surrogative geometric loss is helpful in regulating the motion as is shown in Sec 3.7. We found success in limiting body shake in the session, while failed in generating motions with more intensive arm swing by simply adding $\mathcal{L}_{elbow} = -\frac{1}{N-1}\sum_{i=1}^{N-1} ||\hat{x_0}_{elbow}^{i+1} - \hat{x_0}_{elbow}^{i}||_2^2$ term. However, we believe that, with better engineering and design efforts on hand keypoints, more stylistic and vivid motions can be generated.

## 6 Conclusion

In this paper, we take a step towards constructing a music conditional conductor motion generation framework, by introducing a contrastive pre-training before the controllable diffusion model. And extensive experiments are conducted to improve the generation performance. In summary, our project is the first work to use diffusion model for music-conditional conductor motion generation. And we propose a new L2 constraint of $x_0$ instead of $\epsilon$ to achieve a better performance on generating conductor motion. Finally, we discuss some potential possible improvements.

## References


[1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[2] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[3] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

[4] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[5] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[6] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

[7] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[8] Lucas Mourot, Ludovic Hoyet, François Le Clerc, François Schnitzler, and Pierre Hellier. A survey on deep learning for skeleton-based human animation. In *Computer Graphics Forum*, volume 41, pages 122–157. Wiley Online Library, 2022.

[9] Delong Chen, Fan Liu, Zewen Li, and Feng Xu. Virtualconductor: Music-driven conducting video generation system. *CoRR*, abs/2108.04350, 2021.

[10] Siyuan Yang, Jun Liu, Shijian Lu, Meng Hwa Er, and Alex C Kot. Skeleton cloud colorization for unsupervised 3d action representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13423–13433, 2021.

[11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[13] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[14] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[16] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[17] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[18] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[19] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[20] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[21] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[22] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Xin Chen, Biao Jiang, Wen Liu, Zilong Huang, Bin Fu, Tao Chen, Jingyi Yu, and Gang Yu. Executing your commands via motion diffusion in latent space. *arXiv preprint arXiv:2212.04048*, 2022.

[26] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H Bermano. Human motion diffusion model. *arXiv preprint arXiv:2209.14916*, 2022.