Generating Music-Driven Choreography with Deep Learning

Songha Ban
STUDENT NUMBER: 2023907

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF SCIENCE IN COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE
DEPARTMENT OF COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE
SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
TILBURG UNIVERSITY

Thesis committee:

Dr. Sharon Ong
Dr. Gonzalo Napoles

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science & Artificial Intelligence
Tilburg, The Netherlands
May 2021

Preface

Throughout the study, I have received a great amount of support.

I would first like to thank my supervisor, Dr. Sharon Ong, for the valuable guidance. Her weekly feedback, advice, and inspiration greatly helped me formulating the research questions, proceeding with experiments, and organizing the results, from which I could learn a lot.

I would also like to acknowledge Nataraja Academy for providing their dance videos for the dataset and Ren et al. (Ren et al., 2020) for releasing the code of their excellent work of dance synthesis in public. In addition, I would like to thank Uni-T and TSDV DanceNation members for participating in the qualitative evaluation of the dance generation results.

Finally, I want to thank my family, friends, and boyfriend for their feedback, support and encouragement.

Generating Music-Driven Choreography with Deep Learning

Songha Ban

Generating choreography is challenging as it requires knowledge in both dance techniques and musical elements. In this work, to generate a high-quality dance dataset, different pose estimation methods were explored, and a pose cleaning method was developed. For the actual dance generation task, an existing framework was modified to investigate different music encoding approaches and to improve the performance. Specifically, LSTM and GRU were tested as a music encoder, and a novel music feature generator was proposed to reconstruct music features from the generated dance. The results demonstrate that the proposed approach with GRU as a music encoder performs better than the existing model, creating natural dance movements matching with the music. A demo video of the experiments is available at `https://youtu.be/UE9QnT59LlI`.

1. Introduction

Dance is a performing art consisting of sequential rhythmical motion units (Kim et al., 2003). Choreography is designing dance to express ideas and emotions. As choreography is usually made with and performed with music, it harmoniously engages different modalities such as motor, visual and auditory (Tang et al., 2018). However, generating natural choreography is technically challenging because dance and music are abstract art forms, and no established rules are defining the clear relationship between them (Lee et al., 2018). Furthermore, modeling dance movements is complicated because of the long-term spatio-temporal structures (Lee et al., 2019).

The generation of human body movements has been actively investigated recently. It is essential for a wide range of applications including, but not limited to, robots, gaming, animation, and virtual reality. While composing body movements into dances can be intuitive and spontaneous (Madison et al., 2011), creating aesthetic and rhythmic choreography is more complex as it requires creativity, diverse dance techniques, and comprehension of musical elements. A successful choreography generation could be beneficial in the field such as machine creativity, motion generation (Yalta et al., 2015), and cross-modality generative tasks (Lee et al., 2019).

Previous studies have made various attempts to generate dance movements. Lee et al. (2013) and Fan et al. (2012) generated dance in perspective of similarity-based retrieval, and Ofli et al. (2008) classified the genre of music first and then generated dance movements accordingly. However, these approaches work only within the predefined database or patterns and therefore limit the creativity and generalizability of the model.

This study aims to generate natural, novel, and beat-matching choreography that is consistent with the musical content using deep learning. Two main goals to achieve this are pose estimation and dance generation. For the first goal, different pose estimation methods will be compared to find the best one to create a dataset for dance generation. For the second goal, a deep learning framework will be developed to generate novel, dynamic, and beat-matching choreography that is consistent with the musical content.

The focus is on extracting powerful representation of music and creating quality dance movements that reflect the representation well enough. The dance in the second part of the study, both for the dataset and the generation target, refers to a sequence of poses extracted by pose estimation methods.

1.1 Research Question

This study was conducted based on the following research question:

RQ: "To what extent can deep learning generate natural and dynamic choreography from music?"

One of the recent studies by Ren et al. (2020) successfully generated natural and novel dance movements from the music with generative techniques. However, possible limitations are causing a lack of dynamics, variety, and style consistency. First, the dataset used for the training includes dances of extremely different genres such as ballet and popping. Dancers specialize in one or a few styles of dance through long-term training and continuous practice. Training different genres with not enough data in each genre could have resulted in dance with little dynamics and style-inconsistency. Second, dance styles selected for the dataset are not appropriate for generating quality choreography. Pose sequences of popping are impossible to reflect all the movements such as waves and pops, which are the main components of popping, because the pose data has a limited number of joints. Also, K-pop dance is usually choreographed as a group dance with complicated formations rather than an energetic solo dance. The data is from solo dancer's videos, which can influence the dynamics and variety of the generated dance. To improve this, this study created a new dataset composed of selected choreographies with more variety and dynamics.

Because dance in this study is represented as a sequence of poses, pose information needs to be extracted from the dance videos to create a dataset. Several pose estimation methods are available for converting videos to pose sequences, and popular methods are OpenPose (Cao et al., 2019) and AlphaPose (Fang et al., 2017). Because pose data output from these models can be messy, selection of an optimal algorithm and post-processing of the pose data are significant to achieve better results. This will was examined based on the following subquestion:

SQ1: What is the best approach to extract pose sequences from videos?

In addition, to improve the music encoder which extracts features of music, this study will explore LSTM (Sak et al., 2014) and GRU (Cho et al., 2014) that are proven to be effective in handling temporal information (Bai et al., 2018). Features from the music encoder are important because they are used to generate pose sequences, and therefore must contain useful information to create choreography. This leads to the following subquestion:

SQ2: "Which of LSTM and GRU as a music encoder does achieve the best performance?"

To make sure that encoded music features are a good representation to map between dance and music and that the generated dance is matching the music content, the generated dance will be used to reconstruct the music features. Dancers can not only come up with movements from music, but also can think of what musical elements such as beat, tempo, accent structure, and emotions would fit when watching dance without audio. The visual-based beat tracking method proposed by Pedersoli et al. (2020) also showed that inference of music features from dance movements is achievable. Music feature regeneration from the generated dance can be useful for obtaining better representations

of music and enhancing the expressiveness of the choreography. Moreover, in case of success, it has the potential to be used as a cross-modal evaluation metric.

SQ3: "Is it possible to reconstruct music features from the generated dance? If so, does it improve the overall performance?"

1.2 Summary of Contributions

The contributions of this study are summarized as follows. First, different pose estimation methods on solo dance videos were evaluated and compared. Second, a method to clean messy pose data was developed. Finally, the proposed music feature generator saved training time and improved the overall performance.

2. Related Work

2.1 Pose Estimation

Pose estimation is estimating the configuration of the human body from an image or a video. Recent studies have made remarkable advances in pose estimation. OpenPose is a multi-person 2D pose estimation system to detect the human body and hand, facial, and foot keypoints. It takes a bottom-up approach by using Part Affinity Field (PAFs) which is information about limbs from the image (Cao et al., 2019). Because OpenPose is only based on a single frame, it shows good results on clear images but disregards the context over consecutive frames, and therefore can be unreliable on frames with problems such as blurring due to fast motion or different lighting. AlphaPose (Fang et al., 2017), in contrast, takes a top-down approach which is also called as Detect-and-Track method (Girdhar et al., 2018) because it detects a bounding box first and then estimates the pose. It is also single-frame-based as OpenPose. Pose Flow (Xiu et al., 2018) is a pose tracker based on cross-frame poses, which is useful to extract poses from videos and to indicate the same person across frames. It uses AlphaPose as a pose estimator and fully considers spatio-temporal information to track poses. However, this does not mean that Pose Flow performs better or is more appropriate for this experiment than the other methods. It is possible that Pose Flow which takes cross-frame poses may not be appropriate for this experiment, and therefore, further investigation is needed to select the best pose estimation method.

Most of the previous studies about dance generation such as (Ren et al., 2020), (Lee et al., 2018), and (Lee et al., 2019) used OpenPose to estimate poses from the dance videos, but the results were reported to be messy, which could have had negative impacts on the generated dance. This work differs from other work as it investigates different pose estimators for dance generation specifically and applies a cleaning step to polish the noisy pose data. No existing work has compared these techniques for creating a dataset for dance generation.

2.2 Dance Generation

In recent years, researchers have proposed various approaches to generate dance movements conditioned on music. Tang et al. (2018) used LSTM-autoencoder, and Lee et al. (2018) used causal dilated highway convolutional blocks (CDHC) for an end-to-end mapping between music and dance. They used L_2 and L_1 distance respectively, yet L_2 and L_1 loss are demonstrated to ignore characteristics of some specific movements (Martinez et al., 2017).

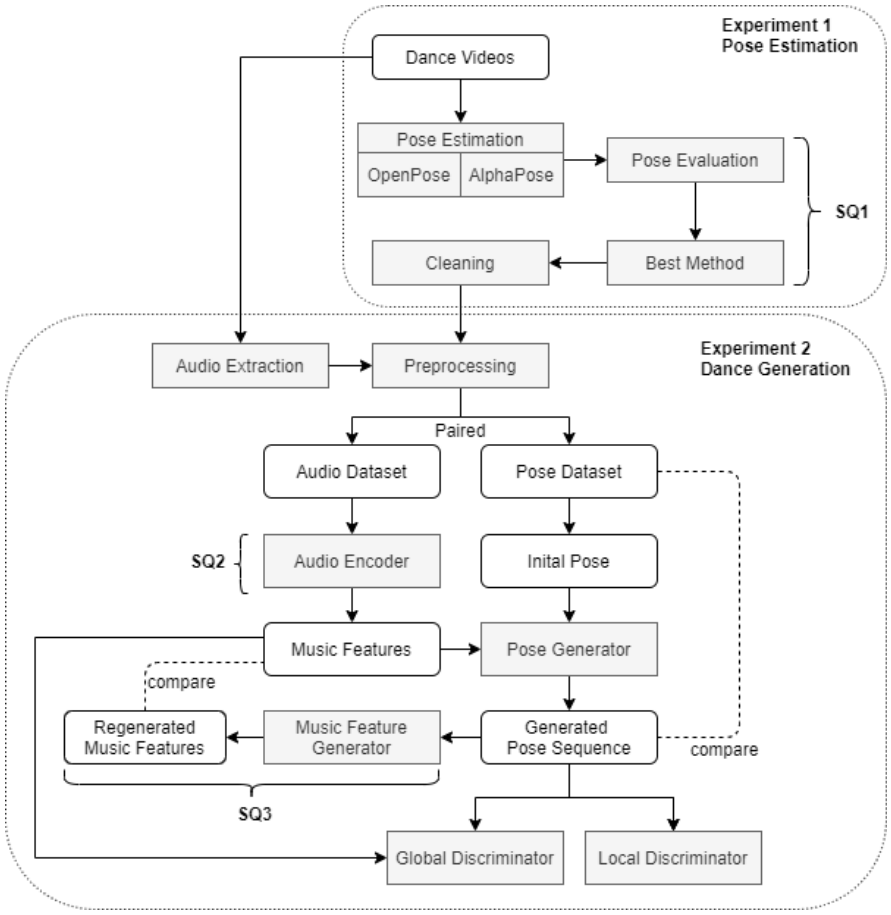

Figure 1
Overview of the complete process. This research consists of two parts: pose estimation and dance generation. They are two separate experiments but are also connected since the results of pose estimation will be used as dataset for dance generation.

To overcome this problem, Ren et al. (2020) applied ST-GCN (Yan et al., 2018) to pose sequences and computed perceptual loss to measure similarity between the original pose sequence and the generated one. In addition, Lee et al. (2019) proposed a more complicated framework consisting of a decomposition phase, where it learns basic movements, and a composition phase, where it learns to compose the basic movements to dance according to the given music. Common points of both studies are that they used GANs (Goodfellow et al., 2014) and that they successfully generated natural, style-consistent, and beat-matching dance movements. However, the movements were not diverse and dynamic especially in genres other than ballet.

This study compares different deep learning models for encoding music features which were not implemented in the previous work. In addition, this study will develop an auxiliary generator that regenerates music features from the generated dance.

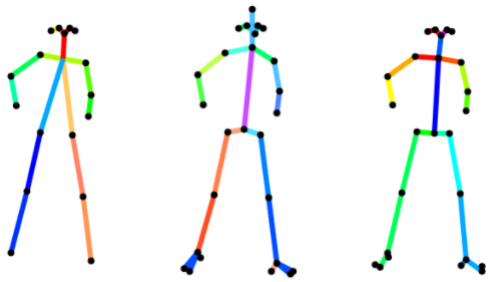

Figure 2
Sample poses extracted from the dataset by AlphaPose with 17 joints, AlphaPose with 26 joints, and OpenPose with 25 joints (from the left to the right).

3. Pose Estimation

Pose estimation is the first part of this study, and it aims to find the best method to extract pose sequences from dance videos to make a high-quality dataset for dance generation. Different pose estimation frameworks were investigated to select the best one to be used for this study.

3.1 Dataset

The dataset consists of carefully selected videos of open-style choreography performed by a solo dancer. Open-style, which is also known as urban dance, is not limited to one genre or specific techniques but draws inspiration from all different genres such as street, hip hop, contemporary, and jazz, and therefore is remarkably creative and expressive. The selected videos are mostly based on street dance or hip hop and meet the following criteria: 1) the dance movements illustrate musical styles and beats well; 2) the dance movements are dynamic with a variety of footwork, levels, and use of space; 3) the dance movements are big and clear enough so that the skeleton's movements can be recognized as real dance movements. The dance videos were provided by Nataraja Academy and myself, and 50 out of 140 videos of total 115,986 frames, which is about 40% of the entire dataset, were used for this experiment. The 50 videos had the same size of 640 x 320 and frame per second (fps) of 30.

3.2 Methods

The raw dance videos were fed into the pose estimation methods without any preprocessing. OpenPose (Cao et al., 2019), AlphaPose (Fang et al., 2017), and Pose Flow (Xiu et al., 2018), the most widely used frameworks, were used for this experiment to perform pose estimation on solo dance videos.

3.2.1 OpenPose. OpenPose is a 2D multi-person pose estimation library where each joint of the skeleton is represented as 2D coordinates. OpenPose has functions to detect keypoints of body, hand, and face, but hand and face are not necessary for this task, so only body detection was used. It takes a single image as an input and returns coordinates of the keypoints from the image. As mentioned in the previous section, OpenPose adopted a bottom-up approach using confidence maps to detect body parts

and PAFs to associate parts. A greedy parsing algorithm is used to parse the confidence maps and the PAFs, and they are assembled to be returned as a full body pose (Cao et al., 2019). Because a demo file for OpenPose is available in executable (EXE) format, a video can be directly given to the demo file which automatically splits the video into a sequence of images and outputs the pose estimation result per frame. Since OpenPose was developed for multi-person pose detection, it is important to make sure that the input image contains only one person to guarantee accuracy. *BODY_25* and *COCO* are the two available models of OpenPose, and the default model is *BODY_25*. It is documented that *COCO* is less accurate, hence *BODY_25* was used for this experiment.

3.2.2 AlphaPose. The use of AlphaPose is similar to OpenPose: a 2D multi-person pose estimator with each joint represented as coordinates. However, AlphaPose took a top-down approach which detects a bounding box, generates pose proposals, and then outputs the actual pose (Fang et al., 2017). Out of multiple models and the number of keypoints available, *Fast Pose* model with 26 keypoints was used because it returns poses in the most similar format as the OpenPose *BODY_25*. In addition, *Fast Pose* trained with ResNet152, which had the highest average precision (AP) among models with 17 keypoints, was used to compare with the others because most of the previous dance generation studies used only 17 or 18 joints. Sample poses from OpenPose, AlphaPose with 26 joints, and AlphPose with 17 joints can be found in Figure 2.

3.2.3 Pose Flow. The goal of Pose Flow is not pose estimation but pose tracking. It uses poses estimated by AlphaPose as input to conduct a cross-frame analysis and label each skeleton by tracking if a skeleton from one frame is the same person as a skeleton from another frame (Xiu et al., 2018). Pose Flow was used in this study because the cross-frame analysis is useful for video processing, and I expected it to improve the results of AlphaPose. However, it did not improve or update the single-person pose estimation result but only labels the skeletons. Therefore, Pose Flow was excluded from the evaluation of this experiment, even though it can be useful for future work which will be discussed in Section 6.1.

3.3 Post-processing

The output pose sequences from the above pose estimation methods can still be messy because of missing frames and incorrect detection. Therefore, a pose cleaning method was developed to improve the quality of the pose data. First, it filters out frames with misdetection. When $X \in \mathbb{R}^{V\times 2}$ is a set of coordinates of the keypoints, and i is an index of the current frame, a function f that determines whether the frame contains misdetection was defined as

$$f(X_i) = [\![\max \mid X_i - X_{prev} \mid > 50]\!] \tag{1}$$

where

$$prev = \begin{cases} i-2 & \text{if } f(X_{i-1}) \wedge \neg f(X_{i-2}) \\ i-1 & \text{otherwise} \end{cases}. \tag{2}$$

To be specific, whether the pose is correct or not was estimated by checking if any keypoint's difference with its previous frame is too large, because it is impossible for any

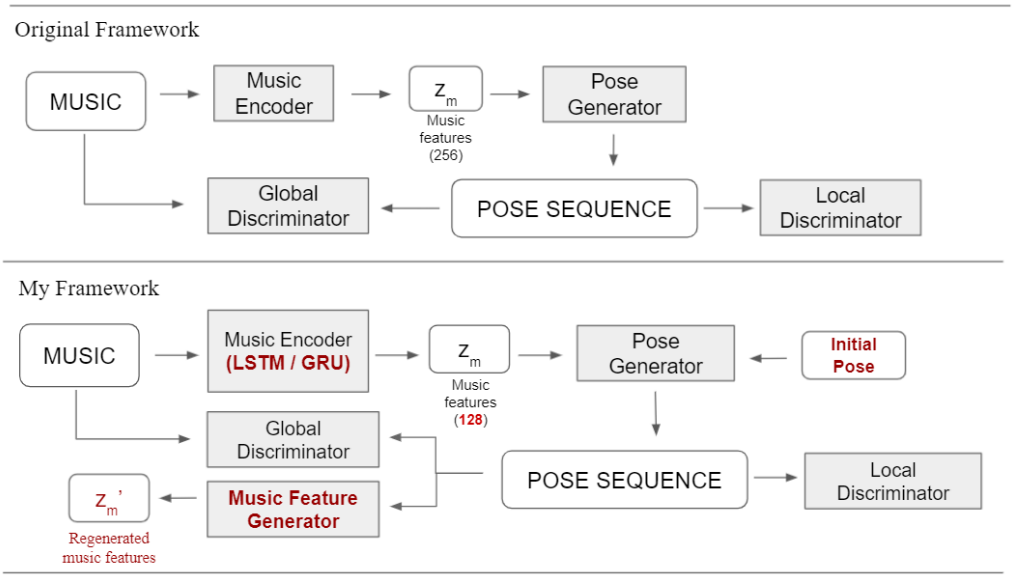

Figure 3
The original framework (top) proposed by Ren et al. (2020) and my adapted framework (bottom) for dance generation.

joint to travel from one position to another one far away within one frame that is 1/30 seconds. If the previous frame X_{i-1} was incorrect detection, it computes the difference with the one frame before, which was the correct frame. However, this applies only when there is no more than one consecutive incorrect frame right before the current frame. Otherwise, only the first incorrect frame is excluded, and the rest are still used to compute differences with the next frame. Then, both missing frames and incorrect frames are recovered using spline interpolation.

3.4 Evaluation Metrics

The number of joints, missing frames, incorrect detection, and confidence scores were used as evaluation metrics. The number of joints was used to check if more joints can reflect the dance better and if there is a difference in accuracy. The number of missing frames was counted from the results of each pose estimation output, and confidence scores were part of the output from the pose estimation. The number of incorrect detection was estimated by the same method as in equation 1. For each metric except for the number of joints, both mean and median were taken into account.

4. Dance Generation

The goal of dance generation is to learn a model $G : X \rightarrow Y$ where X is the music input and Y is dance movements such that the distribution of generated dance $G(X)$ is not distinguishable from the distribution of the real dance Y. The dance movement data are pose sequences extracted by the method selected in the pose estimation experiment.

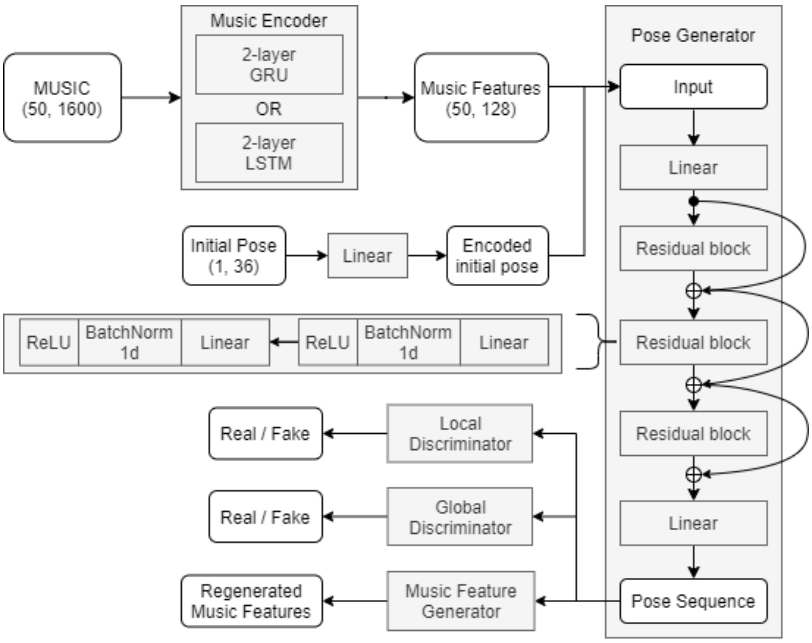

Figure 4
A detailed diagram of the proposed dance generation framework.

4.1 Dataset

To build the dataset, AlphaPose with 17 joints and pose cleaning were applied to all 140 videos with a total length of 9250 seconds. Because the sizes of the videos varied, pose sequences were normalized per clip by relocating and scaling. For relocation, the pose in the initial frame was located in the middle of the frame, and the rest of the frames were adjusted. Then for scaling, the maximum dancer size was calculated by subtracting nose position from the ankle position, and the poses were scaled in a way such that the dancer size fit in the window of 640 x 320. In addition, because 17 joints do not include a joint of the neck, we added it by computing the midpoint of the left and right shoulder, so the final poses have 18 joints. The frame rate was adjusted from 30 to 10. Then, audio was extracted from the videos with a sampling rate of 16000. It was converted to mono to keep only one channel and was normalized with min-max scaling. Then both pose and audio data were split into chunks of 5 seconds.

4.2 Methods

For the main framework, an end-to-end model proposed by Ren et al (2020) was adapted, hence the original framework refers to the framework proposed by Ren et al., and my model or modified model refers to the framework modified by me in the rest of the paper. In my framework, music feature generator was added additionally, which regenerates music features from the generated pose sequences. Figure 4 is a diagram of the framework modified for this study. For comparison of the original framework and my framework, see Figure 3. The following sub-sections are about the details of each

component of the framework, where discriminators are the same as the original and the rest are modified or newly proposed.

4.2.1 Music Encoder. Dance generator (G_d) is composed of music encoder (E_m) and pose generator (G_p). Music encoder as the first part of the dance generator transforms the audio input $X \in \mathbb{R}^{T \times S}$ into a hidden sequence of music features $Z \in \mathbb{R}^{T \times U}$ where T is the number of frames, S is the size of the audio input per frame, and U is the size of the hidden features per frame. The encoder consists of 1D convolution, RNN, and one fully connected layer. RNN part was tested with 2-layer bi-directional GRU (Cho et al., 2014) and 2-layer bi-directional LSTM (Sak et al., 2014). The size of the hidden features was 256 in the original model, but it was adjusted to 128 in the final model.

4.2.2 Pose Generator. Pose generator, as the second part of the dance generator, generates pose sequences $Y \in \mathbb{R}^{T \times 2V}$ given the music features encoded by E_m where V is the number of joints. The pose generator consists of 1 linear layer, 3 residual blocks (He et al., 2016), and the final linear layer as in the original model (Ren et al., 2020). In addition to the music features, an initial pose is also given as an input to the pose generator in my model, which was inspired by Lee et al. (2019). The initial pose is a beginning pose, which is the first frame of the pose sequence for training and the last frame of the previous sequence for inference. The initial pose was added with the expectation to work as a seed for a generation that can give diversity in movements given similar audio sequences and to make the transition between different clips smooth for long-term dance generation. Before the music features are fed into the pose generator, the initial pose goes through one linear layer and is concatenated with the music features.

4.2.3 Music Feature Generator. Music feature generator (G_{mf}) regenerates the music features from the generated pose sequence. It came from the idea that humans can infer some audio features such as beat and tempo from watching dance, and therefore, good dance movements should be able to regenerate the music features. A study about dance beat tracking (Pedersoli, 2020) shows that this is possible. By training the music feature generator, the generated dance can maximize reflecting the music features as dance movements, and the music encoder will also be able to encode the most essential features for making dance movements. The architecture of the music feature generator is a reversed version of the pose generator: 1 linear, 3 residual blocks, and another linear layer. A loss function for music feature generator which evaluates the similarity between encoded music features and regenerated music features from generated poses is defined as

$$\mathcal{L}_{MF} = \sum_{i=1}^{T} \|E_m(x_i) - G_{mf}(G_d(x_i))\|_1 \tag{3}$$

where x is the raw audio input.

4.2.4 Discriminators. Local Temporal Discriminator (D_{local}) and Global Content Discriminator (D_{global}) proposed by Ren et al. (2020) were used to evaluate the authenticity of the generated dance and improve the quality. Local Temporal Discriminator assures that consecutive frames are coherent. Similar to PatchGAN (Isola et al., 2017), it divides a sequence of pose into sub-sequences and determines the realism of the sub-sequences. Global Content Discriminator takes not only the whole pose sequence P but also the

music features Z as input to achieve the harmony between music and dance. Then it uses self-attention mechanism (Lin et al., 2017) to get a comprehensive embedding and classifies whether the pose sequence matches the music features. The adversarial loss is defined as

$$\begin{aligned}\mathcal{L}_{GAN} = &\mathbb{E}_y[\log D_{local}(y)] + \mathbb{E}_x[\log[1 - D_{local}(G_d(x)]]+ \\ &\mathbb{E}_{x,y}[\log D_{global}(x, y)] + \mathbb{E}_x[\log[1 - D_{global}(x, G_d(x))]]+ \\ &\lambda_{GP}\mathbb{E}_{x,y}[(\|\nabla_x D_{global}(x, G_d(x))\|_2 - 1)^2]\end{aligned} \tag{4}$$

where x and y are real music and pose sequence respectively, and λ_{GP} is a weight for the gradient penalty term.

4.2.5 Loss functions. In addition to the adversarial loss and music feature loss, feature matching loss and L_1 loss are used for training the dance generator as implemented by Ren et al. (2020). The feature matching loss (Wang et al., 2018) is similarity between global features of real and generated dance. It was adopted to train the Global Content Discriminator more steadily and is defined as

$$\mathcal{L}_{FM} = \mathbb{E}_{x,y} \sum_{i=1}^{M} \left\| D^i_{global}(x, y) - D^i_{global}(x, G_d(x)) \right\| \tag{5}$$

where M is the number of layers in D_{global}. To calculate pixel-wise distance between the real and generated pose sequences, L1 loss is defined as

$$\mathcal{L}_{L_1} = \sum_{j=1}^{2\times V} \|y_i - G_d(x_i)\|_1 . \tag{6}$$

The full objective is

$$\arg\min_G \max_D \mathcal{L}_{GAN} + \mathcal{L}_{MF} + \lambda_{FM}\mathcal{L}_{FM} + \lambda_{L_1}\mathcal{L}_{L_1} \tag{7}$$

where λ_{FM} and λ_{L_1} are weights for the loss terms. Even though the original framework included pose perceptual loss in the full objective, which is computed using features extracted by pretrained ST-GCN, my model did not use it because it performed better without the pose perceptual loss.

4.3 Implementation Details

The models in this study were implemented in PyTorch and were trained on an NVIDIA GeForce RTX 3060 Ti GPU. The hyper-parameters were set as follows: $T = 50$, $S = 1600$, $U = 128$, $V = 18$, $\lambda_{GP} = 1$, $\lambda_{FM} = 1$, and $\lambda_{L_1} = 200$. Adam optimizer (Kingma & Ba, 2015) was used for all the networks with a learning rate of 3e-4 for the dance generator, the music feature generator, and the local temporal discriminator, and 5e-4 for the global content discriminator.

4.3.1 Baselines. To compare my approach with the original framework, I evaluate the following baselines and models: (1) **Pretrained Ren et al.**. The original model from Ren

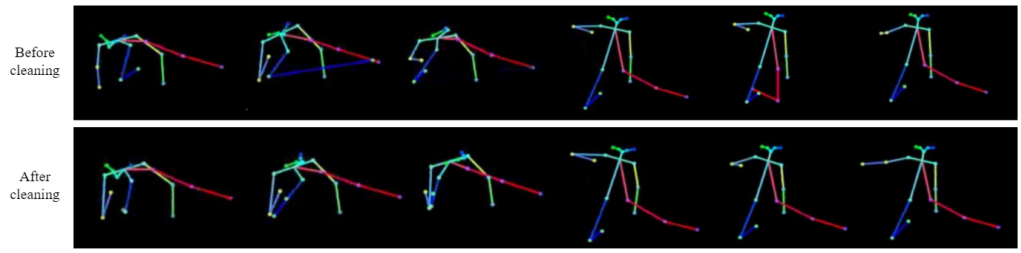

Figure 5
Before (top row) and after (bottom row) applying the pose cleaning method

et al. (2020) trained with their data is used as the primary baseline; (2) **Original.** The original framework from Ren et al. is trained with the new dataset created for this study, and this is the second baseline; (3) **LSTM**. The modified framework for this study with LSTM in the music encoder is evaluated; (4) **GRU**. The modified framework for this study with GRU in the music encoder is evaluated.

4.4 Evaluation Metrics

To compare the baselines and my models, I used automatic metrics for quantitative evaluation and real dancers' feedback for qualitative evaluation. For quantitative evaluation, Frechet Inception Distance (FID) (Heusel et al., 2017) was used to assess the similarity between the generated dances and the real dances. As traditional FID evaluates the quality of images generated by GAN by comparing distributions, in this task, the distance between the distribution of the generated dance and the distribution of the real dance was measured. In addition, beat coverage and beat hit rate were used to analyze if the music beats and motion beats from the generated dances are matching. As introduced in Lee et al. (2019), beat coverage is the number of motion beats divided by music beats (B_{motion}/B_{music}), and beat hit rate is the number aligned beats divided by total motion beats ($B_{aligned}/B_{motion}$), where the aligned beats $B_aligned$ is the number of motion beats aligned with the music beats. Music beats were obtained by computing onset strength (Ellis, 2007) from the audio, and motion beats were detected by using standard deviation (Yalta et al., 2019).

For qualitative evaluation, 22 amateur and professional dancers were invited to watch 6 pairs of dances generated with different styles of music by the **Original** model and the **GRU** model, which were the best baseline and the best model from the quantitative evaluation and manual observation. Each pair was labeled as (A, B) with a random order instead of the actual model names. Then, the dancers were asked to answer 4 questions for each pair: (1) *Which dance matches the music beats better?* (2) *Which dance is more natural/realistic?* (3) *Which dance is more dynamic?* (4) *Which dance do you like more?*.

5. Results

5.1 Pose Estimation

Table 1 is the result of simple evaluation of AlphaPose and OpenPose. OpenPose had the least number of missing frames but not with a big difference with the AlphaPose. AlphaPose with 17 joints had the least number of incorrect detections while OpenPose

Table 1
Comparison of pose estimation methods

Method	# joints	# missing frames		# incorrect detection		confidence score
		mean	median	mean	median	mean
AlphaPose1	17	20.0	6	**76.2**	**46**	0.82
AlphaPose2	26	19.9	6	127.4	83	**0.84**
OpenPose	25	**15.3**	**3**	339.1	340	0.62

Table 2
Result of quantitative evaluation. For FID, lower is better. For beat coverage and beat hit rate, higher is better. The details of the baselines are described in Section 4.3.1

Method	FID	Beat Coverage (%)	Beat Hit Rate (%)
Real dances	-	60.3	91.5
Ren et al.	18.3	47.9	89.9
Original	16.6	50.0	91.2
LSTM	16.0	**51.3**	90.6
GRU (my model)	**14.8**	50.4	**91.5**

had 4.5 times as much. AlphaPose with 26 joints had the highest average confidence score, while OpenPose had the lowest. Each metric shows different results, but the number of incorrect detection was considered the most significant to determine the best method because it directly influences the quality of pose data the most.

Although the number of missing frames is also important, the gap between OpenPose and AlphaPose was not noticeable. In addition, even though OpenPose had the least missing frames, it had a lot of frames missing parts of the keypoints, while AlphaPose had all the keypoints filled in. The confidence score was considered the least important because it is output from the pose estimation system and therefore can be model-dependent. In terms of the number of joints, 25 and 26 joints are more expressive than 17 joints because they have keypoints of feet. However, what feet add is trivial because most of the dance movements in the dataset are focused on arms, body, and legs rather than on feet, and the number of incorrect detection was significantly higher than 17 joints. 17 is enough to represent dance movements from the dataset, and it leaves less space to generate unrealistic poses for the training later as well. Therefore, AlphaPose with 17 joints was selected to create a complete dataset for the actual dance generation in Experiment 2.

After the best framework was selected, the pose cleaning method was applied to the output pose sequences. It successfully recovered missing frames and incorrect detection improving the overall quality of the pose sequences, making the dance movements look more stable than before. Sample results can be found in Figure 5.

5.2 Dance Generation

5.2.1 Quantitative Evaluation. Table 2 shows the automatic metrics for the baselines and the models. The original framework trained with my dataset performed better

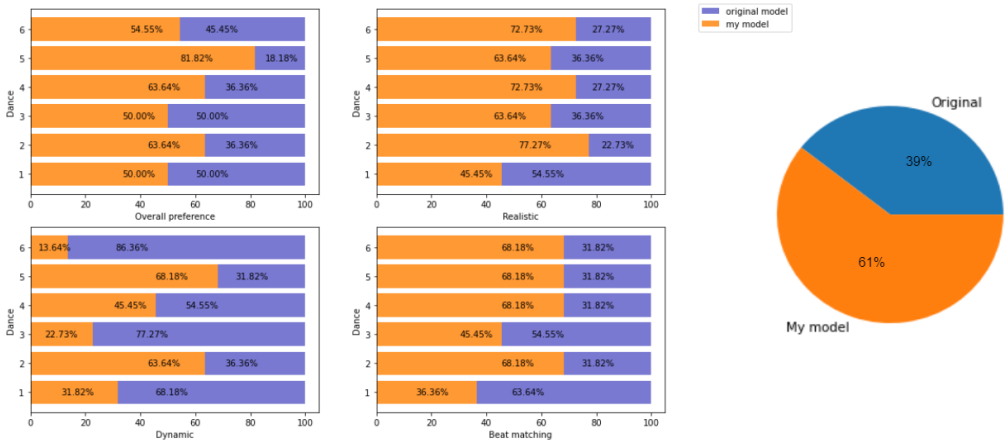

Figure 6
Horizontal bar graphs show the ratio of the dancers' preference on overall preference, realism, dynamic, and beat matching. The pie graph is the ratio of the overall preferences for all 6 dances. Orange is my model with GRU, and purple/blue is the original model trained with my dataset.

than the pretrained original model in all FID, beat coverage, and beat hit rate. The new framework with LSTM in the music encoder was the best in beat coverage and slightly worse in FID and beat hit rate than the original framework, yet still better than the pretrained original model. The new framework with GRU in the music encoder was the best in FID and beat hit rate. Overall, the differences between models are negligible in these metrics.

5.2.2 Qualitative Evaluation. The qualities of dance generated by different models were first manually compared. The pretrained model and my model with LSTM successfully generated beat matching choreography, but the dance movements were relatively small and unnatural. The original model trained with my dataset generated natural, beat matching, and dynamic dances, and the movements were remarkably bigger and more clear than the others. However, sometimes the movements were repetitive and not matching with the music style. My model with GRU generated natural, beat matching, and style-consistent dances. It generated more complicated, human-like movements including shoulder movements and jumps, but the movements were not as big or dynamic as the original model.

Figure 1 is the result of the evaluation by real dancers. As can be seen, dancers preferred my model to the original model in 4 out of 6 dances, and even for the other two dances, 50% preferred my model. They answered that my model's dance is more natural and realistic in 5 out of 6 dances, and more beat matching in 4 out of 6 dances. On the other hand, the original model was more dynamic in 4 out of 6 dances. On average, 61% preferred the dance generated from my model to the one from the original model.

6. Discussion

The first aim of this study was to find the best pose estimation method for the dance dataset, and the second aim was to generate natural, dynamic, and beat-matching

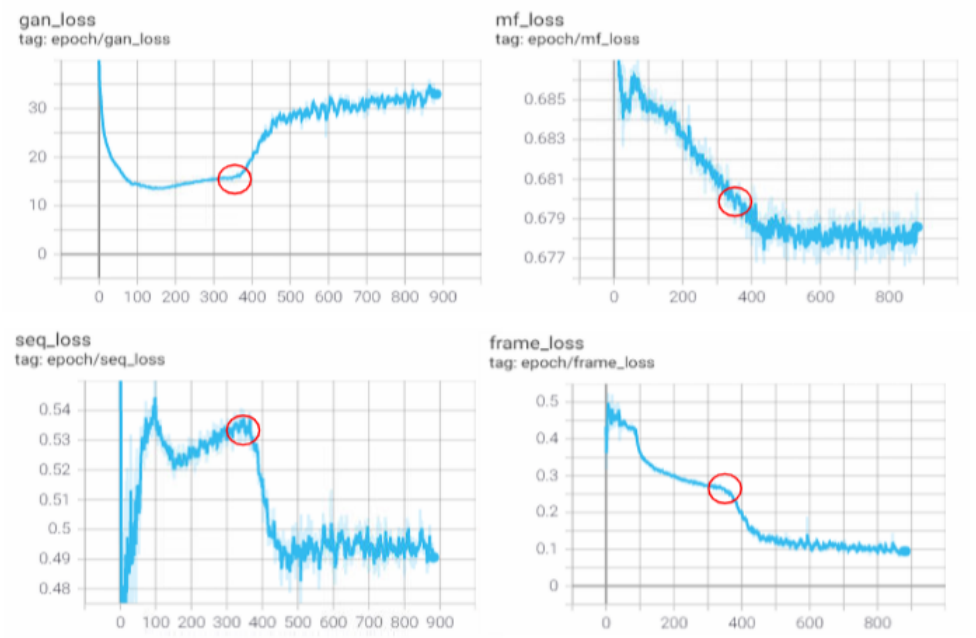

Figure 7
Training loss of the GRU model over epochs.

dance. In this section, the results of the experiments, limitations, and future work will be discussed addressing the research question and subquestions.

6.1 Pose Estimation

The first sub research question was *"What is the best approach to extract pose sequences from videos?"*. The result of the pose estimation experiment answers this question showing that AlphaPose with 17 joints produced the least number of incorrect detection and an acceptable number of missing frames. Also, interpolating the missing frames and incorrect detection improved the quality of the pose sequences. While previous studies (Ren et al., 2020), (Lee et al., 2018), (Lee et al., 2019) used OpenPose, and OpenPose had the least number of missing frames, the experiment showed that it caused more incorrect detection than AlphaPose, and there were a number of frames with parts of the keypoints missing which also counted as incorrect detection. Because the pose data is used for dance generation, the quality of the pose sequences can directly influence the quality of generated dance in the latter task. By selecting the best pose estimation method and cleaning the pose data, this study built the dance dataset of better quality for the second experiment.

There are still limitations in the pose cleaning method. Even though it improved the quality of the pose data, it still cannot find all the misdetections, especially when the pose is consistently incorrect over several consecutive frames because it detects incorrect frames by computing displacements. Further exploration to improve the cleaning method such as considering more frames and the context of the movements may be beneficial for handling messy pose data for similar tasks.

Pose Flow was not used in this study because the dance video dataset only had solo dance videos, but it can be used in the future to expand the dataset. There are not a lot of dance datasets available because the videos need to have only one person at a clean and stable angle. However, if Pose Flow is used for tracking one person in the multi-person dance videos, there will be many more options available to build a dance dataset. Moreover, it also opens up the potential for future study to generate group dance with formations, not just solo dance in one position.

6.2 Dance Generation

The main difference between the original framework and my framework for dance generation is the music feature generator. This answers the third subquestion: *"Is it possible to reconstruct music features from the generated dance? If so, does it improve the overall performance?"*. The number of music features was adjusted to 128 to facilitate the music feature regeneration and also to extract only essential features. The dance generated by the original model with 256 music features tends to react to minor beats which real dancers would have ignored. By reducing the size to 128, the music encoder was trained to extract only essential features, and the smaller size made it easier for the music feature generator to regenerate the music features from the generated dance. As a result, the music feature generator successfully reconstructed music features from the generated dance up to a certain level, and it improved the performance which can be seen in both quantitative and qualitative results when the music encoder was GRU. Figure 7 shows that the music feature loss (mf_loss) went down until training for 400 epochs when the optimal training state was after 350 epochs.

The second subquestion was *"Which of LSTM and GRU as a music encoder does achieve the best performance?"*. The quantitative evaluation results show that the distribution of the GRU-generated dance is more similar to the distribution of real dance than LSTM. LSTM had a higher beat coverage, and the GRU had a higher beat hit rate, but with only slight differences. After manual inspection, I found out that LSTM dances were more reserved, unnatural, and repetitive than GRU dances. A similar result was found in a previous study (Lee et al., 2019) as well that LSTM-generated dances collapsed into certain movements regardless of the input music, and it was because of the deterministic nature of LSTM, which can disturb mapping to the highly unconstrained dance movements. Some of the outputs from my LSTM model also showed similar repetitive movements even though they were generated from different styles of music. Moreover, for the LSTM model, the music feature generator did not converge, which means it failed to regenerate music features from the generated dances. Meanwhile, the GRU model did not have the same issues. It was able to generate more elaborate movements proactively using shoulders and legs, and the music feature generator was also successfully trained.

In addition, feeding the initial pose as an input to the pose generator not only saved training time but also made the transition between each sequence of poses more smooth. The original model trained for 800 epochs which took around 18 hours, but my model trained only for 350 epochs which took around 4 hours. Furthermore, both the original and my framework accept 50 frames for each sequence, which is about 5 seconds long, and this makes it difficult to generate dance longer than 5 seconds. By setting the last pose of the previous sequence as an initial pose of the current sequence, my model, even though it was not perfect, was able to generate long-term dance with more natural transitions.

The main research question was *"To what extent can deep learning generate natural and dynamic choreography from music?"*. This study would answer that deep learning can generate natural, dynamic, and beat-matching choreography. However, dynamic and natural is a trade-off, as the original model was more dynamic but less natural, and my model was the opposite. The overall results show that my dataset contributed to better performance within the same framework, and my model performed better than the baseline model. However, I had to rely on human evaluation more than the automatic metrics, because the existing evaluation methods are not good and sophisticated enough in practice. Therefore, developing better evaluation metrics for dance generation is one important future research topic.

7. Conclusion

In this study, experiments to compare different pose estimation methods for dance pose sequences and to compare different networks for dance generation framework were conducted. The best pose estimation method, AlphaPose with 17 joints, was applied to the dance videos to extract pose sequences, and then pose cleaning was performed to complete a dataset for dance generation. The dance generation framework used GRU to encode music features, and generated dance from the music features and a given initial pose. A music feature generator was added to regenerate music features from the generated dance to build a strong mapping between music and dance. This approach can generate long-term natural, dynamic, and beat-matching choreography. However, the dynamics and long-term generation still need to be improved to generate more human-like dance movements, and the development of better metrics to quantitatively evaluate the dance pose sequences remains as future work.

Acknowledgements

A part of the dance video dataset used in this study was provided by Nataraja Academy.

Self-reflection

Working on this bachelor thesis was an invaluable experience. As a student passionate about deep learning, and as an amateur dancer having danced for the whole life, combining different interests of mine was a big pleasure and motivation. Even though I underestimated the topic and was being overly ambitious at the beginning, I learned how to specify the goals, design the study, plan for the experiments, and analyze the results throughout the study. Moreover, I only focused on generating dance itself at first, but later I realized it is also very important to have solid metrics for evaluation. It was challenging for me to implement the evaluation metrics, but this process let me explore and learn more about different metrics used for GAN and the ideas behind them. In addition, while struggling to put my ideas and research into clear text, I learned how to use mathematical equations and notations for clarification. Although I was not able to test out all the ideas I had for this study because of the limited amount of time, I appreciate that I was able to gain more insights about cross-modal generative tasks and handling of spatio-temporal data, which I can expand for my future works.

References

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv*.

Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S., & Sheikh, Y. A. (2019). Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Cho, K., van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, (pp. 103–111)., Doha, Qatar. Association for Computational Linguistics.

Ellis, D. P. (2007). Beat tracking by dynamic programming. *Journal of New Music Research*, *36*(1), 51–60.

Fan, R., Xu, S., & Geng, W. (2012). Example-based automatic music-driven conventional dance motion synthesis. *IEEE Transactions on Visualization and Computer Graphics*, *18*(3), 501–515.

Fang, H. S., Xie, S., Tai, Y. W., & Lu, C. (2017). RMPE: Regional Multi-person Pose Estimation. *Proceedings of the IEEE International Conference on Computer Vision*, *2017-Octob*, 2353–2362.

Girdhar, R., Gkioxari, G., Torresani, L., Paluri, M., & Tran, D. (2018). Detect-and-Track: Efficient Pose Estimation in Videos. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 350–359.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks. *NIPS'14*, *2*, 2672–2680.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-December*, 770–778.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., & Hochreiter, S. (2017). GANs trained by a two time-scale update rule converge to a local Nash equilibrium. *Advances in Neural Information Processing Systems*, *2017-Decem*(June 2019), 6627–6638.

Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-January*, 5967–5976.

Kim, T.-h., Park, S. I., & Shin, S. Y. (2003). Rhythmic-motion synthesis based on motion-beat analysis. *ACM Transactions on Graphics*, *22*(3), 392–401.

Kingma, D. P. & Ba, J. L. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–15.

Lee, H. Y., Yang, X., Liu, M. Y., Wang, T. C., Lu, Y. D., Yang, M. H., & Kautz, J. (2019). Dancing to music. *arXiv*, (NeurIPS), 1–11.

Lee, J., Kim, S., & Lee, K. (2018). Listen to dance: Music-driven choreography generation using autoregressive encoder-decoder network. *CoRR*, *abs/1811.0*.

Lee, M., Lee, K., & Park, J. (2013). Music similarity-based approach to generating dance motion sequence. *Multimedia Tools and Applications*, *62*(3), 895–912.

Lin, Z., Feng, M., Dos Santos, C. N., Yu, M., Xiang, B., Zhou, B., & Bengio, Y. (2017). A structured self-attentive sentence embedding. *ICLR*.

Madison, G., Gouyon, F., Ullén, F., & Hörnström, K. (2011). Modeling the Tendency for Music to Induce Movement in Humans: First Correlations With Low-Level Audio Descriptors Across Music Genres. *Journal of Experimental Psychology: Human Perception and Performance*, *37*(5), 1578–1594.

Martinez, J., Black, M. J., & Romero, J. (2017). On human motion prediction using recurrent neural networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 4674–4683.

Ofli, F., Demir, Y., Yemez, Y., Erzin, E., Tekalp, A. M., Balcı, K., Kızoğlu, , Akarun, L., Canton-Ferrer, C., Tilmanne, J., Bozkurt, E., & Erdem, A. T. (2008). An audio-driven dancing avatar. *Journal on Multimodal User Interfaces*, *2*(2), 93–103.

Pedersoli, F. (2020). Dance beat tracking from visual information alone. *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR 2020*.

Ren, X., Li, H., Huang, Z., & Chen, Q. (2020). Self-supervised Dance Video Synthesis Conditioned on Music. *Proceedings of the 28th ACM International Conference on Multimedia*, 46–54.

Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *INTERSPEECH*.

Tang, T., Jia, J., & Mao, H. (2018). Dance with melody: An LSTM-autoencoder Approach to Music-oriented Dance Synthesis. *MM 2018 - Proceedings of the 2018 ACM Multimedia Conference*, 1598–1606.

Wang, T. C., Liu, M. Y., Zhu, J. Y., Tao, A., Kautz, J., & Catanzaro, B. (2018). High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 8798–8807.

Xiu, Y., Li, J., Wang, H., Fang, Y., & Lu, C. (2018). Pose flow: Efficient online pose tracking. *BMVC*.

Yalta, N., Ogata, T., & Nakadai, K. (2015). Sequential Deep Learning for Dancing Motion Generation. *Technical Report of Japanese Society for Artificiial Intelligence*, 43–49.

Yalta, N., Watanabe, S., Nakadai, K., & Ogata, T. (2019). Weakly-Supervised Deep Recurrent Neural Networks for Basic Dance Step Generation. *Proceedings of the International Joint Conference on Neural Networks*, *2019-July*(June 2020), 1–8.

Yan, S., Xiong, Y., & Lin, D. (2018). Spatial temporal graph convolutional networks for skeleton-based action recognition. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 7444–7452.

Appendices

Appendix A: Training loss

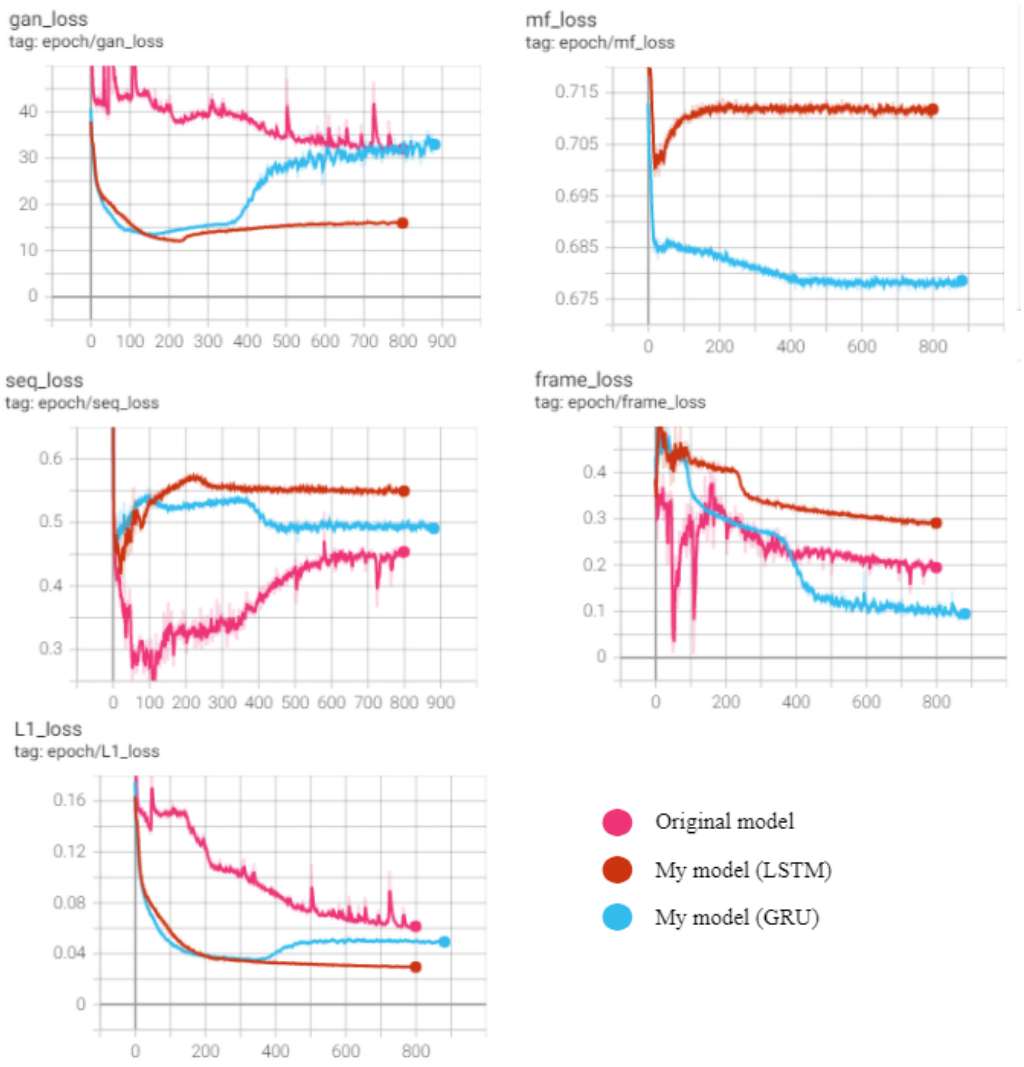

Figure 1
Training loss of the baseline and the models over the training epochs.

Appendix B: Github Repository

https://github.com/SonghaBan/DancingAI