Sorbonne Université
Master ANDROIDE

JustLMD: Dance Motion Synthesis from Music and Lyrics

Stage de Master 2

Réalisé par :

Qingyuan Yao

Encadré par :
Yi Yu, National Institut of Informatics

Référent :
Aurélie Beynier, LIP6, Sorbonne Université

September 1, 2023

Contents

Chapter 1

Introduction

1.1 Internship

This internship is a research internship of 115 working days, in the domain of generative deep learning, at Digital Content and Media Sciences Research Division of the National Institute of Informatics in Tokyo, under the supervision of Prof. Yi Yu.

The National Institute of Informatics is an inter-university research institute established in 2000. It has 4 main divisions carrying out research ranging from basic informatic theories to data science and artificial intelligence. For university students, it offers graduate programs and international exchange for Master's and Ph.D. students.

The internship was based on an idea of Prof. Yu, and was conducted with the help of Prof. Yu and Wenjie Yin, a Ph.D. intern student specializing in dance style transfer with diffusion model. The result of this internship (including the dataset and the baseline model) would potentially be used by Wenjie, as well as the members of Prof. Yu's laboratory.

1.2 Abstract

In recent years, alongside with other types of generation tasks, dance motion synthesis has been extensively explored with the rise of deep generative networks. However, while the status quo consists of generating dance motion from music, the contribution of lyrics in choreography is often ignored in spite of clear observations of semantic influences on the dance moves.

To address this problem, we have extracted from dance videos of the popular video game Just Dance and created a dataset consisting of 1867 triplets of motion sequence, audio and lyrics, in which the motions are aligned frame-by-frame with the audio, while the lyrics, for the moment, only have a sentence-level alignment with the other two.

In order to prove the validity of adding lyrics as condition, we have also created a baseline model which consists of a Conditional Variational Autoencoder with Transformer as encoder and decoder.

1.3 Validity of the Task

In the field of dance choreography, dancing in accordance to the lyrics of a song has already been explored, officially and unofficially, as the lyrics often contain the information of the song that the music doesn't. There exists a dance style called lyrical dance[1], despite its style being limited between ballet and jazz, it emphasizes the semantic meaning of the lyrics. Moreover, there has been analysis[2] proving a strong link between choreography and lyrics in modern dance.

While we are aware that the semantic influence of lyrics is not omnipresent, and that there exists a considerable amount of dance songs without lyrics. We are still convinced that lyrics play an important part during the creative process of choreography for some styles of dance. We consider the task of dance synthesis from music and lyrics should be considered valid by these arguments (examples provided alongside):

- **Direct semantic links to motion** (ex. "NO" links to shaking heads or making "X" with arms). This is the most obvious reason, but not always valid, as choreography also depends on the music and the style.
- **Emotional/contextual influence** (ex. "Break my heart" links to a more quiet and sad motion). This could work with music on reinforcing the emotional intensity.
- **Pattern Influence** (ex. "Oh oh oh" could indicate 3 repetitive motions)
- **Music-Lyrics distinguishing/linking** (same music different lyrics / same lyrics difference music. Ex. when two verses have the same music, but not the same lyrics and vice versa) This helps to vary the dance by avoiding having the same choreography in similar parts.
- **Parsing dance sequences**: as lyrics are separated by sentences and paragraphs, it is easier and more natural to separate the sequences of dance by lyrics.

We also provide an analysis of lyrical influence on the choreography in appendix A.

1.4 Limitations

As the dataset consists only English songs from Just Dance, and most of those songs are of the pop genre (despite the effort of expanding variety from the Just Dance team), we are aware of the limitation in terms of the generated content. Therefore, it is recommended to use English pop songs for the inference process. There is, however, future prevision for adding non-English songs into the dataset.

Another limitation is that, while the audio and motions are synced frame-by-frame, the lyrics only have a coarse-level alignment with the former two modalities. This implies that we extracted relationships among them based on sentence-level semantics, which makes it hard to demonstrate the first argument of the validity (cf. 1.3), and could potentially lead to mismatched semantics.

Chapter 2

State of the Art

2.1 Multimodal Datasets of Motion, Audio, and Text

The modalities of text, audio, and motion have been extensively explored throughout the current research landscape.

With motion synthesis being a popular domain in recent years, there exist several datasets that pair descriptions of actions with motion such as HumanML3D, as well as datasets that pair music with motion such as AIST++[3], whose 3D motion data is generated from multi-view video-music dataset AIST[4]. Furthermore, though irrelevant for this internship, it is also important to acknowledge the datasets with paired music and lyrics[5] used for audio-to-text[6] and text-to-audio[7] generation.

As explained above, currently, no lack of paired data has been observed in the domain of text-audio, motion-audio, or motion-text. However, at the writing of this report, a dataset consisting of motion, audio, and text has not been discovered. Upon examining the current tendency of dance in pop culture, it is easy to conclude that choreography of songs with lyrics has become more and more relevant. Therefore, it would be in the interest of pioneering in a new research field, that a dataset of Lyrics-Music-Dance (LMD) triplets be created.

2.2 Deep Learning Models

For the purpose of implementing the baseline and anticipating future improvements, several deep learning structures have been studied in varying degrees depending on the relevance to the internship.

2.2.1 Variational Autoencoder

An autoencoder is a structure including both an encoder that maps input data into a latent space, and a decoder that subsequently reconstructs the data. The latent space represents extracted features from the original data, it is based on the idea that data can be represented by less amount of information depending on the task. In the case of MNIST[8], the images of written numbers could be represented by whether there are curves or straight lines in some positions of the image (cf. Figure 2.1)

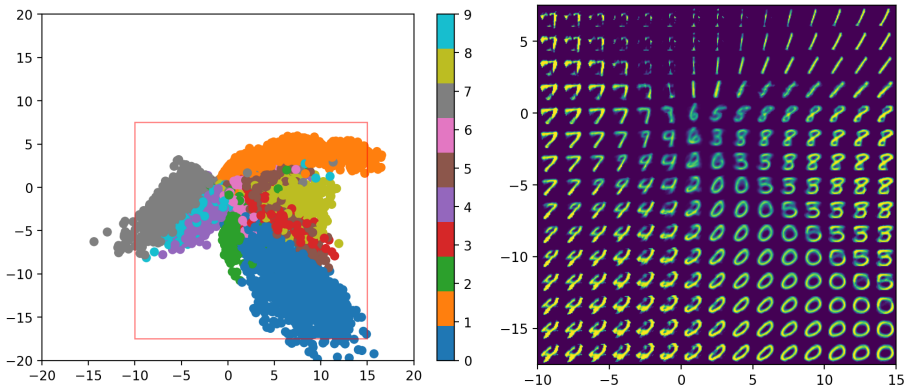


Figure 2.1: Autoecoder's 2D latent space of MNIST (left) and results sampled from the red square area (right)[9]. It can be observed that the images at the top have straighter lines than those at the bottom, which could be a feature represented by the latent space.

A Variational Autoencoder[10] (VAE) incorporates a Gaussian Distribution layer (cf. Figure2.2) to avoid generating blank data when decoding from an area of latent space with no mapped data (cf. Figure 2.3).

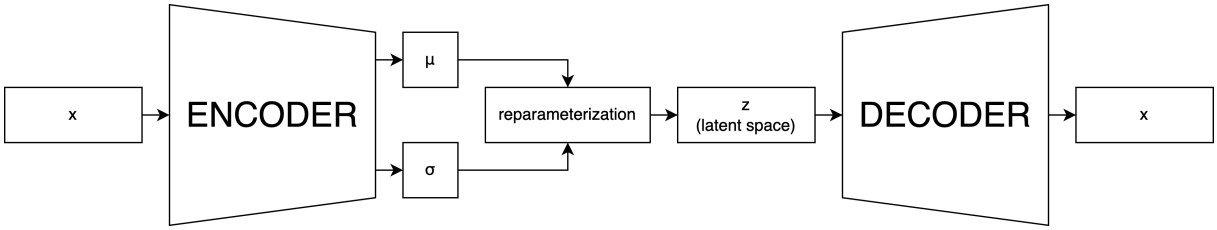


Figure 2.2: **Variational Autoencoder** (x: input and output data with the same shape, trapezoid: linear transformation, μ and σ: mean and standard deviation of the Gaussian distribution)

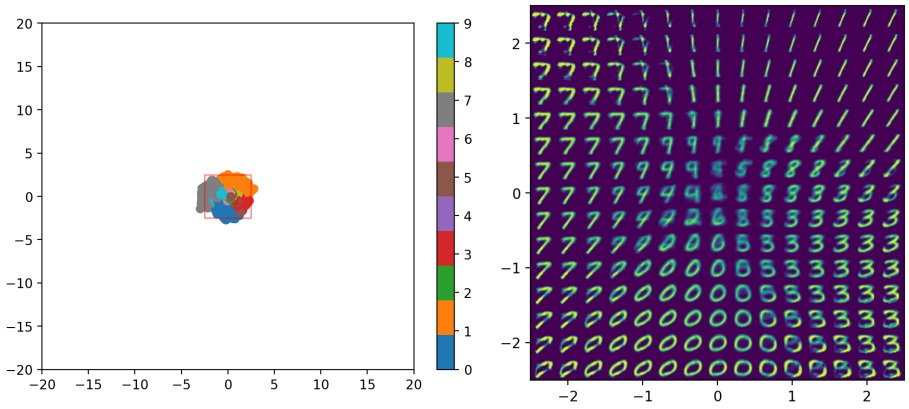


Figure 2.3: VAE's 2D latent space of MNIST (left) and results sampled from the red square area (right)[9]. Compared to a regular autoencoder, variational autoencoder has a smaller and more centralized distribution, producing more realistic digits.

2.2.2 Transformers

Ever since the first paper[11] from 2017, Transformers have been gaining popularity in the research field. They consist of stacks of encoders that encode the tokens and stacks of

decoders that predict the next token given the previous tokens (cd. Figure 2.4) . Unlike recurrent neural networks such as LSTM[12], Transformers use self-attention mechanisms to learn dependencies between each token, and can execute in parallel. They are often used to process sequential data such as text or videos.

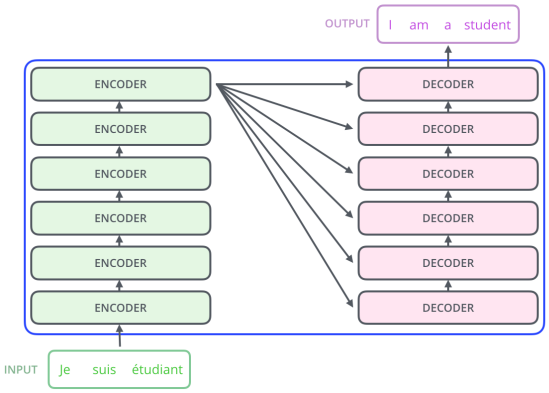


Figure 2.4: Simplified representation of stacks of encoders and decoders in Transformers[13]

As PyTorch already includes a transformer model[14], the internship does not consist of building a transformer model by hand, thus a more in-depth examination of the encoder and decoder layers has not been done.

2.2.3 Diffusion Model

The diffusion model is based on the process of adding noise to data until it becomes pure noise, and then denoising the data until it resembles the original data. Since the noising and denoising processes have the same but reversed endpoints (cf. Figure 2.5), the noising process can help the denoising process by providing a reference at each step of the denoising process. The type of data used can vary from images [15], to audio mel-spectrogram[16], text embeddings[17], and motion[18].

It is worth mentioning that, the structure used in each step of the denoising process is a U-Net[19], which is also the structure of a VAE (cf. Figure 2.2).

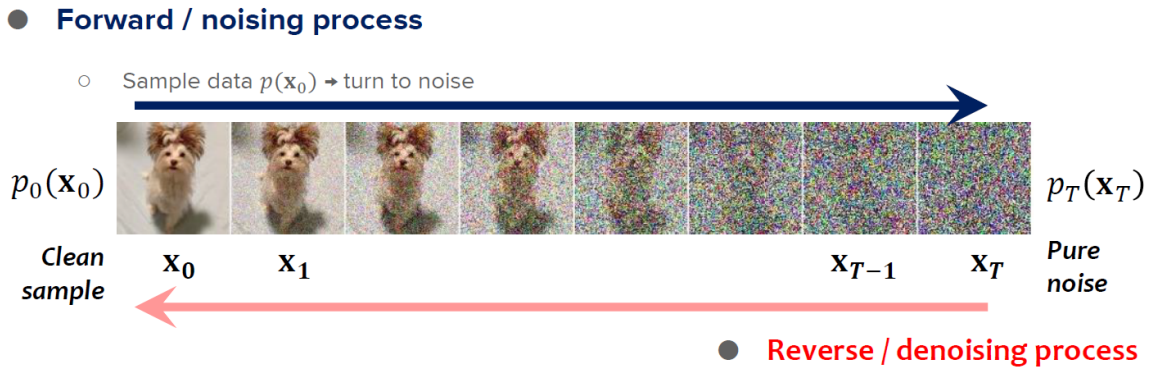


Figure 2.5: Noising and denoising process[20]

2.3 Dance Motion Synthesis

The domain of human motion generation mainly focuses on generating everyday life behaviors, which have evolved from motion-matching to deep generative models. In most cases, the actions are generated from labels[21] or text[4]. Dance motion generation is a subdomain that is usually conditioned with music, it has also been extensively studied with the rise of generative models such as VAE[22], LSTM[23], Transformer[24] and diffusion model[18].

2.4 Pose Estimation

In order to extract motion from videos, the task of pose estimation[25] has also been studied. It describes the process of extracting skeleton motion (2D or 3D) or human mesh shape from videos. While it suffices with a skeleton representation of the human body, some models also recover the mesh shape of the person in the videos (usually to the SMPL[26] format), which has a lesser known name: Human Mesh Recovery. This task would sometimes rely on a 2D skeleton extraction to reconstruct the mesh animation.

The most popular benchmark for 3D Human Pose Estimation is the dataset of Human3.6M[27]. Which consists of 3.6 million 3D human poses and corresponding images from 4 cameras. The benchmark ranks both models that work for monocular view (images from one camera), as well as multi-view (images from different cameras at different angles).

2.5 Just Dance

Just Dance[28] is a motion-based rhythm dancing game first published in 2009, and it has been getting yearly releases since 2014. As of 2023, there are 14 main entries, along with more than 10 spin-offs. It is estimated to have more than 1000 songs, most of which are English songs with lyrics. Furthermore, despite the dances varying from solo dance to quartet, most choreographies are individual ones that don't require interactions between the dancers.

Chapter 3

Contributions

3.1 Dataset

3.1.1 Video Selection

Both motion and audio are extracted from dance videos. The motion is extracted with a mesh recovery model (cf. 3.1.2), and the music is extracted directly from the video. Most videos are not the original gameplay videos of Just Dance, as the characters' skin and costumes are over-processed, and the background changes constantly. Moreover, in recent years, the instances of camera movement of the videos has been increasing, which makes it impossible to track all parts of the body for some songs.

Instead, videos of YouTubers dancing to the game have been selected for the extraction of the motion. The samples must satisfy the following criteria:

- The dancer in the samples should be performing identical motions to the original choreography.
- The whole body of the dancer can be visible throughout the dance.
- There should be no or minimal camera movement.
- The frame rate of the video should be 30 frames per second (fps) or the multiple of that.

After manually selecting videos that correspond to these criteria, a .json file is then created containing links to all the songs, which would be passed into the pipeline.

3.1.2 Motion Extraction

As the videos lack the motion data themselves, there needs to be the process of extracting skeleton/mesh animation from the video. Several models of pose estimation and mesh recovery have been tested:

- **MMPose**[29]: The model was used by a paper[30] that has previously worked with Just Dance. Its 2D pose estimation has shown accurate results, but the 3D pose estimation shows some stuttering.

- **EasyMocap**[31]: Despite the model being originally implemented for multi-view mesh recovery, it includes an option for monocular mesh recovery, and it works well with dance videos, showing precise and fluid motions.
- **MotionBERT**[32]: At the time of writing this report, this model is still ranked the first on the benchmark of Monocular 3D Human Pose Estimation on Human3.6M[33]. It does a good job on 3D pose estimation as expected, however, the mesh recovery results aren't as good as EasyMocap.
- **FrankMocap**[34]: This is a relatively light weighted model for mesh recovery partially published by Facebook, compared to EasyMocap, the results aren't as smooth.

It is worth noting that, the models have also been used to extract motions from both original gameplay videos and YouTuber dance videos, the former have resulted in different degrees of impreciseness and stuttering whilst the latter have shown much more promising results.

After extensive testing, EasyMocap shows the best performance (cf. Figure 3.1) along with a relatively easy setup.

Figure 3.1: Result of EasyMocap

3.1.3 Embeddings for Audio and Lyrics

As it is hard to store raw audio data into the Dataset class of PyTorch, and the lyrics don't have the same length, the audio and the lyrics are first extracted into embeddings before storing into the dataset. The audio features are extracted by librosa[35], each frame has 128 bits of features. The text of lyrics has been tokenized and then extracted by a Bert model[36], in order to concatenate with audio features, there are extra linear transformations reshaping the text features as the same size as audio features (the validity of this extra process is debatable, cf. 3.2.3).

3.1.4 Pipeline

After choosing the tools for extracting motion as well as the embeddings of lyrics and music, a pipeline has been established to generate the three modalities of data from the provided YouTube links and lyrics files.

Most of the pipeline execution, including the inference process of the EasyMocap model used for motion extraction, has been run on a T4 Nvidia graphics card on the computer from the lab of Prof. Yu.

Version 1

The first version of the pipeline aims to carefully keep the choreography of each dance sequence as complete as possible, with length varying between 5 and 15 seconds. All lyrics have been manually examined for synchronization and concatenation. The pipeline operates as follows:

1. **Pick dance videos**: The videos of each song need to be manually selected and marked for cropping if necessary.
2. **1_DOWNLOAD.py**: Download videos and audios from the given YouTube links using pytube[37], apply the cropping if specified.
3. **Add lyrics**: Unfortunately, simply downloading the .lrc files doesn't suffice, as most lines of lyrics only last 1-2s in the songs, and some videos have extra introductions before the song starts. It is necessary to first synchronize the lyrics' timestamps with the audio, and then concatenate lines together in order to obtain dance sequences that last between 5 and 15 seconds.
4. **2_SLICE.py**: Slice the audios and the videos according to the timestamps of the lyrics.
5. **3_POSE2D.py**: Estimate 2D "skeleton" animations.
6. **4_EXTRACT.py**: Generate mesh animations from the skeleton animations and the images from the videos.
7. **Filter results**: Some videos can't be extracted by the inference model, and thus need to be re-extracted or deleted. For viewing the motion, a video with the mesh covering the person (cf. 3.1) is generated for each dance sequence. It is also possible to convert the SMPL file to .bvh and view it in Blender (cf. 3.1.5).
8. **5_LOAD.py**: Create embeddings from the audios and lyrics and load them into DataLoader alongside the motion data.

The first version guarantees that each dance sequence is well-defined and well-sliced from the videos. However, it requires quite a lot of amount of manual work which takes one week (while working on other tasks) to finish the extraction from Just Dance 2022.

Version 2

After revising the first pipeline with the supervisor, it has been identified that varying lengths of each dance sequence poses a challenge to the process of model learning. Moreover, the manual process of slicing lyrics takes too much time. Therefore, it has been decided to optimize the pipeline by fixing the length of each sequence to 6 seconds and automating some processes on the lyrics.

The second version of the pipeline follows the steps below:

1. **Pick dance videos** (same as version 1)

2. **1_DOWNLOAD.py**: download videos with yt-dlp (the previous method stopped working due to changes on the YouTube website).
3. **Add lyrics**: Manually search for lyrics and sync ONLY the first line of lyrics to the audio.
4. **2_SLICE.py**: Trim the video to contain only the dance sequence. Autoslice by timestamps into sequences longer than 6 seconds and save the sliced data in a .json file. (cf. Figure 3.2 and 3.3)
5. **3_POSE2D.py** (same as version 1)
6. **4_EXTRACT.py**: Extract 3D mesh recovery via the previous 2D pose estimation and the images. All the frames are concatenated and stored in a single smplfull.json.
7. **5_LOAD.py**: Load the motion, the audio embeddings, and the lyrics embeddings of each sequence into a Dataset class. (The slicing of the dance sequences happens only HERE and not before.)

The new pipeline speeds up the process by reducing manual labor such as slicing the lyrics. Thanks to this, the extraction from Just Dance 2020-2022 which consists of 103 songs, or 1867 sequences, has taken less than a week.

```
[00:14.830] Yeah, breakfast at Tiffany's and bottles of bubbles
[00:18.260] Girls with tattoos who like getting in trouble
[00:21.690] Lashes and diamonds, ATM machines
[00:25.280] Buy myself all of my favorite things (Yeah)
[00:28.870] Been through some bad ****, I should be a sad *****
[00:32.010] Who woulda thought it'd turn me to a savage?
[00:35.440] Rather be tied up with calls and not strings
[00:38.900] Write my own checks like I write what I sing, yeah (Yeah)
[00:42.180]
[00:42.400] My wrist, stop watchin', my neck is flossin'
[00:45.330] Make big deposits, my gloss is poppin'
[00:48.950] You like my hair? Gee, thanks, just bought it
[00:52.630] I see it, I like it, I want it, I got it (Yeah)
[00:55.670]
[00:56.160] I want it, I got it, I want it, I got it
[00:59.490] I want it, I got it, I want it, I got it
[01:02.620] You like my hair? Gee, thanks, just bought it
[01:06.360] I see it, I like it, I want it, I got it (Yeah)
[01:09.540]
```

Figure 3.2: Example of a .lrc file


```
{
    "00:14.830": "Yeah, breakfast at Tiffany's and bottles of bubbles  Girls with tattoos who like getting in trouble",
    "00:21.690": "Lashes and diamonds, ATM machines  Buy myself all of my favorite things (Yeah)",
    "00:28.870": "Been through some bad ****, I should be a sad *****  Who woulda thought it'd turn me to a savage?",
    "00:35.440": "Rather be tied up with calls and not strings  Write my own checks like I write what I sing, yeah (Yeah)",
    "00:42.180": "My wrist, stop watchin', my neck is flossin'  Make big deposits, my gloss is poppin'",
    "00:48.950": "You like my hair? Gee, thanks, just bought it  I see it, I like it, I want it, I got it (Yeah)",
    "00:55.670": "I want it, I got it, I want it, I got it  I want it, I got it, I want it, I got it",
    "01:02.620": "You like my hair? Gee, thanks, just bought it  I see it, I like it, I want it, I got it (Yeah)",
    "01:09.540": "Wearing a ring, but ain't gon' be no \"Mrs.\"  Buy matching diamonds for six of my *******",
```


Figure 3.3: Example of the .json file resulted from auto-slicing of the original .lrc file

3.1.5 Preview and Export

In order to better visualize the dance motion, verify the synchronization of music and dance, and eventually export the data, we have added methods to view and export the sequences in the dataset, as well as the inference.

There are three types of functions which serve different needs:

- visualize(): For visualizing the SMPL mesh frame by frame, we use the package of NoSMPL [38], upon executing the function, a window pops up showing a model moving in accordance with the given SMPL data. This process does not, however, save the video to a directory, nor is accompanied by audio or lyrics.
- export(): This function is an extension of visualize(), which saves the video to a designated directory, in which it adds the corresponding music sequence to the video and prints the lyrics on the bottom of the video. (cf. Figure 3.4)
- toBvh(): As SMPL is more oriented towards research use than commercial use, for the purpose of facilitating compatibility with the commercial standard format, a script modified from smpl2bvh [39] which translates SMPL to .bvh has been prepared for future need of commissioning professional animators to beautify the results. (cf. Figure 3.5)

Figure 3.4: Exported video with SMPL animation, audio, and lyrics

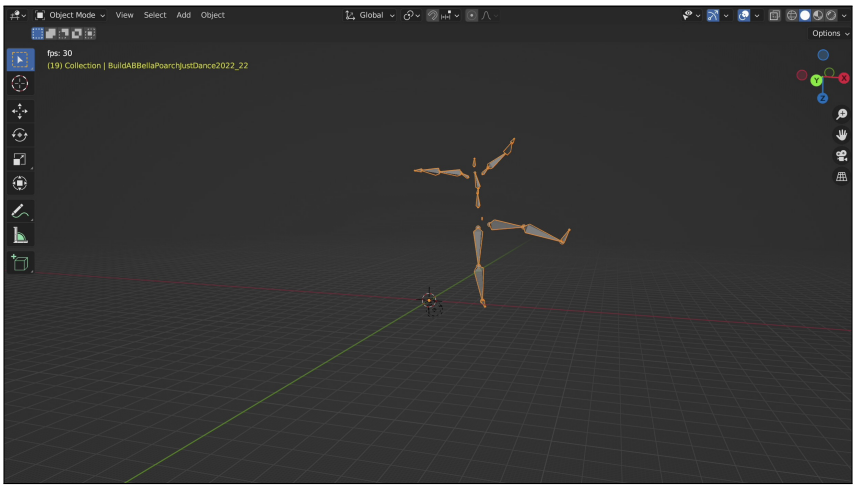


Figure 3.5: .bvh file format in Blender

3.2 Baseline

Given the complexity of our multimodal dataset with audio, lyrics, and motion, generating motion when provided with audio and lyrics is indeed a challenging task. Therefore, we

have chosen to initially address this intricate challenge using Variational Autoencoder (VAE)[10] instead of the state-of-the-art technology such as diffusion model (though future implementation has been envisioned, cf. 3.2.3).

The training of the first version of the baseline has been run on my personal computer and Google Colaboratory, while the second version has been run on a T4 Nvidia graphics card on the computer from the lab of Prof. Yu.

3.2.1 Version 1: Conditional VAE

Setup

As the task requires motion data as input, and audio/lyrics as condition, in order for the decoder to generate according to the lyrics and the audio, instead of regular VAE, a Conditional VAE (CVAE)[40] is used. The conditioning inputs are the embeddings of the lyrics and features of the audio clip, they are passed into the encoder as well as the decoder.

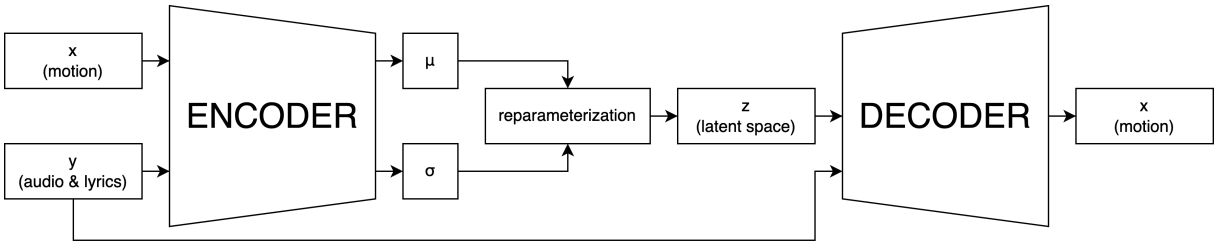


Figure 3.6: **Conditional Variational Autoencoder** (trapezoid: linear transformation, μ and σ: mean and standard deviation of the Gaussian distribution)

As the length of the sequences varies and the feature embeddings of lyrics and audio don't have the same dimension, we have first tried to flatten all the data before passing to the CVAE (the validity of this process is debatable).

The dimensions of the data are as follows:

- **Motion**: 600 frames $\times$ 72 rotations
- **Lyrics + audio**: 50 tokens $\times$ 768 features + 600 frames $\times$ 128 features
- **Mean (μ) + standard deviation (σ)**: 400 + 400 = 800 bits
- **Latent space**: 512 $\times$ 512 = 262,144 bits

The hyperparameters and training configurations are as follows (the initial configurations are based on the original setup for MNIST, which is debatable):

- **Learning rate**: 1e-3
- **Batch size**: 1 (device constraint)
- **Number of epochs**: 100
- **Optimizer**: Adam
- **Activation functions**: elu for encoder, sigmoid for decoder
- **Loss functions**: binary cross entropy + KL divergence

Result

The result of the inference was not ideal, after passing noise as the latent space to the decoder, with flattened embeddings of music and the lyrics as conditions, the model generated noise as output (cf. Figure 3.7).

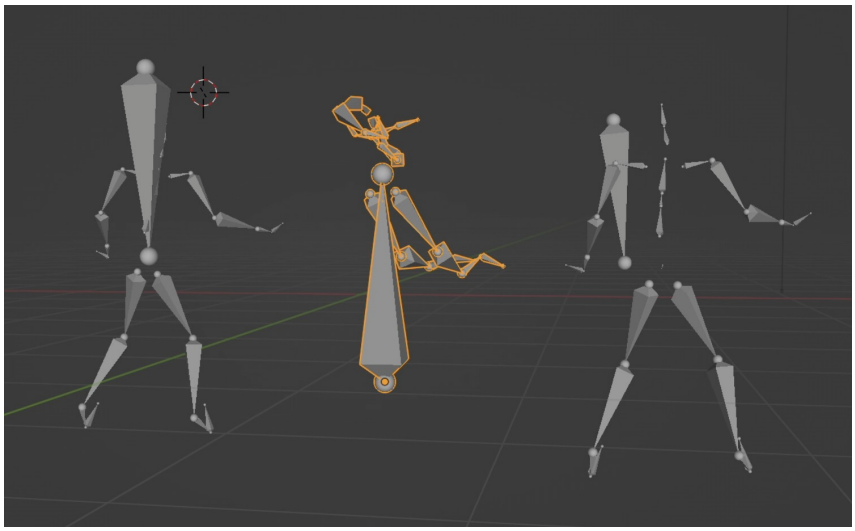


Figure 3.7: Inference in the middle, ground truth on both sides (front and back). Lyrics: "Oh she's sweet but a psycho, a little bit psycho. At night she's screaming: 'I'm-ma-ma-ma out of my mind'"

As the first result showed little promise, and flaws of the setup have been pointed out in the meeting, no future refinement has been done.

3.2.2 Version 2: Transformer CVAE

Setup

As the previous results are mostly noise, it has been deduced that it is due to the difficulty of learning flattened data, as well as the lack of fine-grained alignment information and temporal features of the three modalities, which hinders the effectiveness of representing essential features from both the motion and the lyrics/audio data in the latent space.

In order to fix the problem, instead of flattening all the data, the motion data has been reshaped into 3D tensors which contain the information for each joint at each frame, and the audio data has also been reshaped into 2D tensors which contain audio features at each frame.

Some pre-existing papers whose models consist of temporal-conscience structures such as LSTM, and Transformers have been studied[41][42][24][23]. Eventually, the code for the paper which implements a Transformer Condition VAE[43] has been selected as the base of the second version of the code.

The structure can be explained by Figure 3.8:

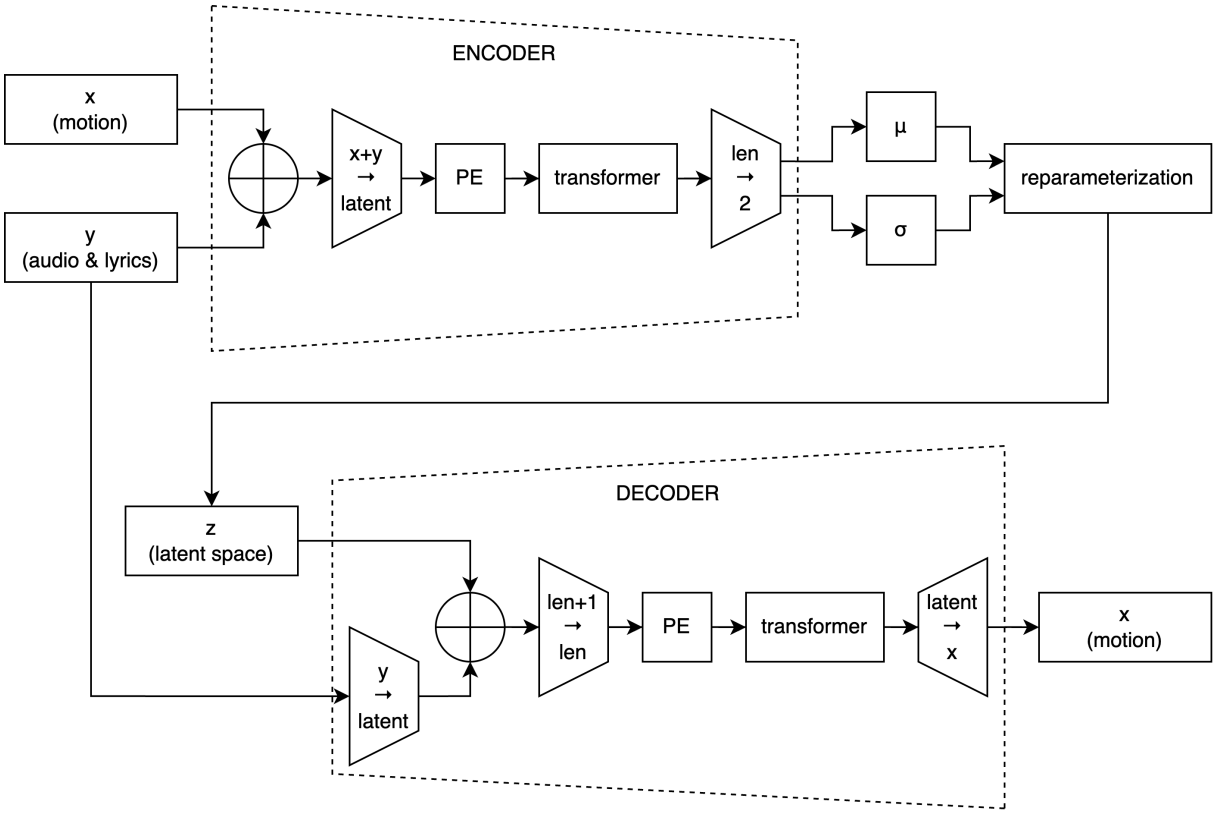


Figure 3.8: **Transformer CVAE** (small trapezoid: linear transformation, μ and σ: mean and standard deviation of the Gaussian distribution, $\oplus$: concatenation, PE: positional encoding)

The dimensions of the data are as follows:

- **Motion**: 180 frames × (24 joints + 1 global rotation + 1 transition) × 3 axes
- **Lyrics + Audio**: 180 frames × (128 audio features + 128 lyrical features) (the embeddings for the text have been linearly transformed into 180×128, therefore it is a false alingement)
- **Mean (μ) + standard deviation (σ)**: 512 + 512 = 1024 bits
- **Latent space**: 512 bits

The hyperparameters and training configurations are as follows:

- **Learning rate**: 0.0001
- **Batch size**: 200
- **Number of epochs**: 5000
- **Optimizer**: AdamW
- **Activation function**: gelu
- **Loss functions**: mean squared error + KL divergence (the rcxyz loss in the original model has been left out)

Result

The first training has been done with only rotations of the joints, excluding the global rotation and the transition. Despite the small dataset and the lack of fine-tuning, the inference result has shown continuous movement mostly correlated with the beat of the music, the semantic influence, however, has not been clearly observed. (cf. Figure 3.9)

Later it has been pointed out that, without the global rotation and transition data, the center of the SMPL model would be immobile, which is not natural human behavior. After adding global rotation and transition data to the training, the inference result has shown stuttering in the movement, as well as irregular rotations (cf. Figure 3.9). This might be due to the reason that the current model is unable to learn with the global rotation and transition simply concatenated to the rotations of the joints.

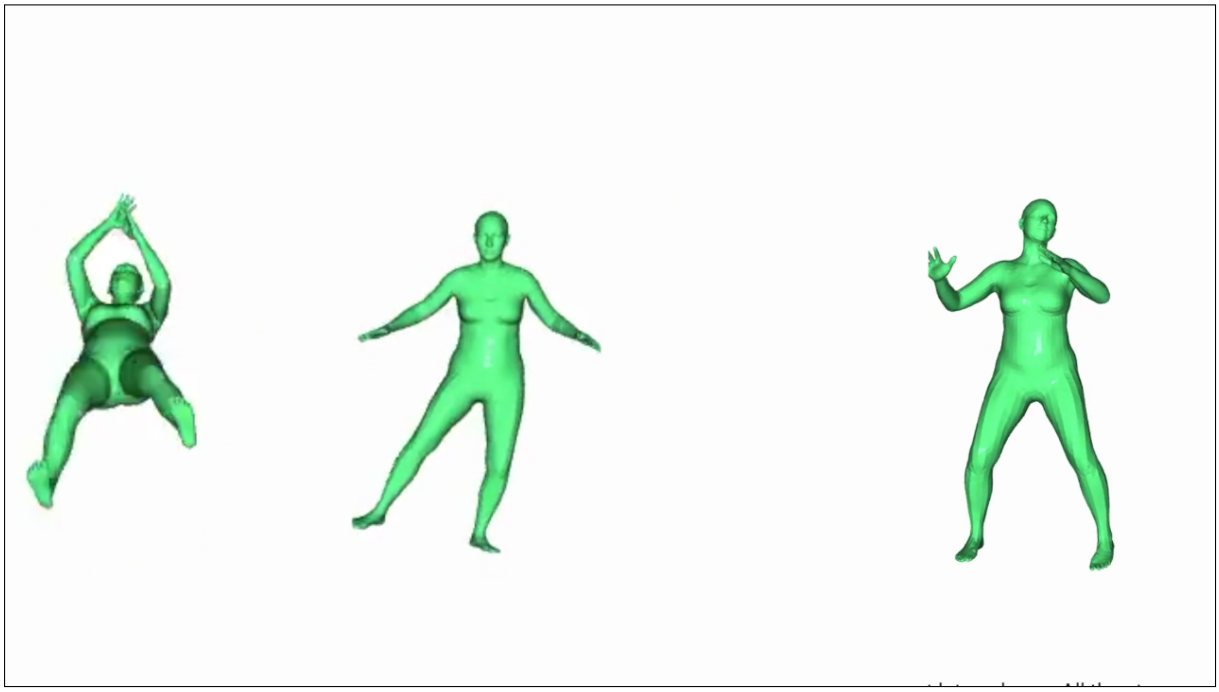


Figure 3.9: Side-by-side comparison between the ground truth (middle), and the inference result of the Transformer CVAE trained with (left) and without (right) global rotation and transition. Lyrics: "This may be the night that my dreams might let me know. All the stars are closer."

3.2.3 Future Improvements

Although the initial results may not be satisfied with our expectations, there are several potential enhancements proposed in meetings with lab members that could lead to improvements of the model in future work:

- Expand the dataset to provide more data to learn.
- Use Jukebox[7] and CLIP[44] as feature extractors. Since Jukebox is trained on audio of songs with lyrics, and CLIP is trained on text association with videos, using these two models could lead to embeddings of better qualities.
- Seeing how smooth the result of the model without global rotation and transition is. It would be interesting to make the model learn joint rotations, transition, and

global rotation separately.

- It is possible that passing non-synchronized lyrics concatenated with synchronized audio would cause confusion in the model. It would be better to use another way to input the lyrics embeddings as conditions, or invent a way for the pipeline to add word-by-word synchronization to the dance and audio sequences in the dataset.
- Investigate alternative designs for the latent space by experimenting with different dimensions, distributions, or re-parameterization techniques to better represent relationships between modalities.
- Lastly, as previously mentioned, VAE is a U-Net structure which is used by diffusion models. There could be a possibility to integrate the entire Transformer CVAE model into the U-Net of each denoising step in a diffusion model.

Chapter 4

Conclusion

In this internship, I have discovered the field of deep generative models with limited prior knowledge of neural networks. I managed to find a solution to the lack of the dataset for the new task proposed by the supervisor. I have tested on multiple pre-existing models and tools in order to create an efficient and accurate pipeline. Furthermore, after self-learning popular models like VAE, Transformers, and diffusion models, I have managed to implement 2 versions of baseline models based on pre-existing codes.

Though I have made some debatable decisions due to inexperience, and the circumstances didn't allow me the time to fine-tune and evaluate my model, I am certain future interns will benefit greatly from the dataset, the pipeline, and the baseline that I have created. I look forward to the publication of the relevant paper in the future.

4.0.1 Acknowledgments

I would like to express my gratitude for the positive discussions within the group during this internship. Thanks to Prof. Yu and Wenjie who have lent me their expertise, it has saved me a lot of time from being uncertain of all the possibilities.

I would also like to thank my fellow interns in the lab, Karol and Amine, for helping me understand advanced neural structures, having the patience to listen to my difficulties, and giving me advice on improvements to my model.

Bibliography

[1] *Lyrical Dance - Wikipedia*. en. Page Version ID: 1163306493. July 2023. URL: https://en.wikipedia.org/w/index.php?title=Lyrical_dance&oldid=1163306493 (page 2).

[2] Hayley Elizabeth Powell. "Modern Dance Choreography: Beyond the Movement an Analysis between Lyrics and Movement". en. In: 16 (2019) (page 2).

[3] Ruilong Li, Shan Yang, David A. Ross, and Angjoo Kanazawa. *Learn to Dance with AIST++: Music Conditioned 3D Dance Generation*. 2021. arXiv: 2101.08779 [cs.CV] (page 3).

[4] Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. "Generating Diverse and Natural 3D Human Motions From Text". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 5152–5161 (pages 3, 6).

[5] Gabriel Meseguer-Brocal, Alice Cohen-Hadria, and Geoffroy Peeters. "DALI: a large Dataset of synchronized Audio, LyrIcs and notes, automatically created using teacher-student machine learning paradigm." In: *19th International Society for Music Information Retrieval Conference*. Ed. by ISMIR. Sept. 2018 (page 3).

[6] Olga Vechtomova, Gaurav Sahu, and Dhruv Kumar. *Generation of lyrics lines conditioned on music audio clips*. 2020. arXiv: 2009.14375 [cs.CL] (page 3).

[7] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. *Jukebox: A Generative Model for Music*. 2020. arXiv: 2005.00341 [eess.AS] (pages 3, 15).

[8] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791 (page 3).

[9] Alexander Van de Kleut. *Variational AutoEncoders (VAE) with PyTorch*. URL: https://avandekleut.github.io/vae/ (page 4).

[10] Diederik P. Kingma and Max Welling. "An Introduction to Variational Autoencoders". en. In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392. ISSN: 1935-8237, 1935-8245. DOI: 10.1561/2200000056 (pages 4, 12).

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention Is All You Need". In: arXiv:1706.03762 (Dec. 2017). arXiv:1706.03762 [cs]. URL: http://arxiv.org/abs/1706.03762 (page 4).

[12] Haşim Sak, Andrew Senior, and Françoise Beaufays. *Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition*. 2014. arXiv: 1402.1128 [cs.NE] (page 5).

[13] Jay Alammar. *The Illustrated Transformer*. URL: http://jalammar.github.io/illustrated-transformer/ (page 5).

[14] *Transformer - PyTorch*. URL: https://pytorch.org/docs/stable/generated/torch.nn.Transformer.html#torch.nn.Transformer (page 5).

[15] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising Diffusion Probabilistic Models*. 2020. arXiv: 2006.11239 [cs.LG] (page 5).

[16] Jinglin Liu, Chengxi Li, Yi Ren, Feiyang Chen, and Zhou Zhao. *DiffSinger: Singing Voice Synthesis via Shallow Diffusion Mechanism*. 2022. arXiv: 2105.02446 [eess.AS] (page 5).

[17] Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. *DiffuSeq: Sequence to Sequence Text Generation with Diffusion Models*. 2023. arXiv: 2210.08933 [cs.CL] (page 5).

[18] Jonathan Tseng, Rodrigo Castellon, and C. Karen Liu. *EDGE: Editable Dance Generation From Music*. 2022. arXiv: 2211.10658 [cs.SD] (pages 5, 6).

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV] (page 5).

[20] *Mathematical Foundation of Diffusion Generative Models | Binxu Wang*. URL: https://scholar.harvard.edu/binxuw/classes/machine-learning-scratch/materials/foundation-diffusion-generative-models (page 5).

[21] Taeryung Lee, Gyeongsik Moon, and Kyoung Mu Lee. *MultiAct: Long-Term 3D Human Motion Generation from Multiple Action Labels*. 2023. arXiv: 2212.05897 [cs.CV] (page 6).

[22] Hsin-Ying Lee, Xiaodong Yang, Ming-Yu Liu, Ting-Chun Wang, Yu-Ding Lu, Ming-Hsuan Yang, and Jan Kautz. "Dancing to Music". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/7ca57a9f85a19a6e4b9a248c1daca185-Paper.pdf (page 6).

[23] Taoran Tang, Jia Jia, and Hanyang Mao. "Dance with Melody: An LSTM-autoencoder Approach to Music-oriented Dance Synthesis". In: *2018 ACM Multimedia Conference on Multimedia Conference*. ACM. 2018, pp. 1598–1606 (pages 6, 13).

[24] Jiaman Li, Yihang Yin, Hang Chu, Yi Zhou, Tingwu Wang, Sanja Fidler, and Hao Li. *Learning to Generate Diverse Dance Motions with Transformer*. 2020. arXiv: 2008.08171 [cs.CV] (pages 6, 13).

[25] Haoming Chen, Runyang Feng, Sifan Wu, Hao Xu, Fengcheng Zhou, and Zhenguang Liu. *2D Human Pose Estimation: A Survey*. 2022. arXiv: 2204.07370 [cs.CV] (page 6).

[26] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. "SMPL: A Skinned Multi-Person Linear Model". In: *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34.6 (Oct. 2015), 248:1–248:16 (page 6).

[27] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. "Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.7 (July 2014), pp. 1325–1339 (page 6).

[28] *Just Dance - Wikipedia*. en. Page Version ID: 1168209825. Aug. 2023. URL: https://en.wikipedia.org/w/index.php?title=Just_Dance_(video_game_series)&oldid=1168209825 (page 6).

[29] MMPose Contributors. *OpenMMLab Pose Estimation Toolbox and Benchmark*. https://github.com/open-mmlab/mmpose. 2020 (page 7).

[30] Adi Ojha, Kevin Wang, Mingxuan Zhao, and Yunong Liu. "Just Dance Everywhere: Using Deep Pose Estimation to Get Your Groove On!" en. In: () (page 7).
[31] *EasyMoCap - Make human motion capture easier.* Github. 2021. URL: https://github.com/zju3dv/EasyMocap (page 8).
[32] Wentao Zhu, Xiaoxuan Ma, Zhaoyang Liu, Libin Liu, Wayne Wu, and Yizhou Wang. "Learning Human Motion Representations: A Unified Perspective". In: *arXiv preprint arXiv:2210.06551* (2022) (page 8).
[33] *Human3.6M Benchmark (Monocular 3D Human Pose Estimation).* en. URL: https://paperswithcode.com/sota/monocular-3d-human-pose-estimation-on-human3 (page 8).
[34] Yu Rong, Takaaki Shiratori, and Hanbyul Joo. "FrankMocap: A Monocular 3D Whole-Body Pose Estimation System via Regression and Integration". In: *IEEE International Conference on Computer Vision Workshops*. 2021 (page 8).
[35] Brian McFee, Colin Raffel, Dawen Liang, Daniel P. W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. "librosa: Audio and Music Signal Analysis in Python". In: *Proceedings of the 14th Python in Science Conference*. Ed. by Kathryn Huff and James Bergstra. 2015, pp. 18–24. DOI: 10.25080/Majora-7b98e3ed-003 (page 8).
[36] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL] (page 8).
[37] *pytube*. URL: https://pytube.io/en/latest/ (page 9).
[38] MagicSource. *NoSMPL*. Python. Aug. 2023. URL: https://github.com/lucasjinreal/nosmpl (page 11).
[39] Fukahire. *smpl2bvh*. Python. Aug. 2023. URL: https://github.com/KosukeFukazawa/smpl2bvh (page 11).
[40] William Harvey, Saeid Naderiparizi, and Frank Wood. *Conditional Image Generation by Conditioning Variational Auto-Encoders*. 2022. arXiv: 2102.12037 [cs.CV] (page 12).
[41] Hui He, Qi Zhang, Kun Yi, Kaize Shi, Zhendong Niu, and Longbin Cao. *Distributional Drift Adaptation with Temporal Conditional Variational Autoencoder for Multivariate Time Series Forecasting*. 2022. arXiv: 2209.00654 [cs.LG] (page 13).
[42] Xiaogang Xu, Yi Wang, Liwei Wang, Bei Yu, and Jiaya Jia. *Conditional Temporal Variational AutoEncoder for Action Video Prediction*. 2021. arXiv: 2108.05658 [cs.CV] (page 13).
[43] Mathis Petrovich, Michael J. Black, and Gül Varol. *Action-Conditioned 3D Human Motion Synthesis with Transformer VAE*. 2021. arXiv: 2104.05670 [cs.CV] (page 13).
[44] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV] (page 15).

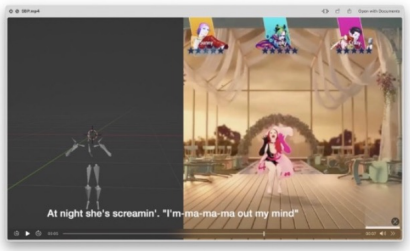
Appendix A

Analysis of Lyrical Influence on the Choreography

Sweet: posing a cute gesture

Psycho: hitting her head with her fists, portraying mental problems

Screaming: scream



Ma-ma-ma: 3 repetitive motions (also a "vulgar" gesture relating to "I'm out of my mind")

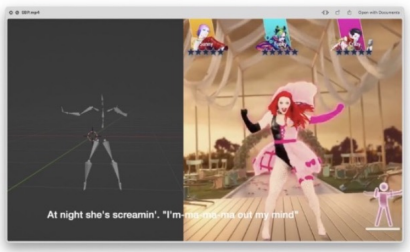


Figure A.1: Analysis of Sweet But Psycho from Just Dance 2023 Edition