AUTOMATIC DJ TRANSITIONS WITH DIFFERENTIABLE AUDIO EFFECTS AND GENERATIVE ADVERSARIAL NETWORKS

Bo-Yu Chen[1*], *Wei-Han Hsu*[1], *Wei-Hsiang Liao*[2], *Marco A. Martínez Ramírez*[2], Yuki Mitsufuji[2], and Yi-Hsuan Yang[1]

[1] Research Center for IT Innovation, Academia Sinica, Taiwan
[2] Sony Group Corporation, Japan

ABSTRACT

A central task of a Disc Jockey (DJ) is to create a mixset of music with seamless transitions between adjacent tracks. In this paper, we explore a data-driven approach that uses a generative adversarial network to create the song transition by learning from real-world DJ mixes. The generator uses two differentiable digital signal processing components, an equalizer (EQ) and a fader, to mix two tracks selected by a data generation pipeline. The generator has to set the parameters of the EQs and fader in such a way that the resulting mix resembles real mixes created by human DJ, as judged by the discriminator counterpart. Result of a listening test shows that the model can achieve competitive results compared with a number of baselines.

Index Terms— DJ mix, audio effects, deep learning, differentiable signal processing, generative adversarial network

1. INTRODUCTION

Recent years witnessed growing interest in building automatic DJ systems. Research has been done to not only computationally analyze existing mixes made by human DJs [1–3], but also develop automatic models that mimic how DJs create a mix [4–8]. We are in particular interested in the automation of DJ transition making, for it involves expert knowledge of DJ equipment and music mixing and represents one of the most important factors for shaping DJ styles.

As depicted in Figure 1, the DJ transition making process is composed of at least two parts. Given a pair of audio files x_1 and x_2, *cue point selection* decides the "cue out" time (C_{out}) where the first track becomes inaudible, and the "cue in" time (C_{in}) where the second track becomes audible. Then, *mixer controlling* applies audio effects such as EQ to the portion of x_1 and x_2 where they overlap (i.e., the *transition region*), so that the resulting mix $\hat{x}_3$ contains a smooth transition of the tracks. While cue point selection has been more often studied in the literature [6, 9], automatic mixer controlling for DJ transition generation remains much unexplored. Existing methods are mostly based on hand-crafted rules [5,7], which may not work well for all musical materials. We aim to automate this process by exploring for the first time a data-driven approach to transition generation, learning to DJing directly from real-world data.

To achieve this, a supervised approach would require the availability of the *original* song pairs (x_1, x_2) as model input and the corresponding DJ-made mix $\hat{x}_3$ as the target output. Preparing such a training set is time-consuming for we need to collect the original songs involved in mixes, detect the cue points, and properly segment the original songs. To get around this, we resort to a generative adversarial network (GAN) approach [10] that requires only a collection of real-world mixes (as real data) and a separate collection of song pairs (called "paired tracks in Figure 1) that do not need to correspond to the real-world mixes (as input to our generator). We use the principle of GAN to learn how to mix the input paired tracks to create realistic transitions.

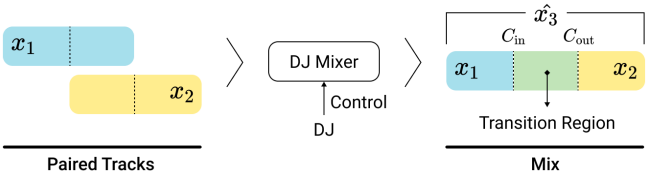

Fig. 1. Illustration of the DJ transition making process.

Moreover, instead of using arbitrary deep learning layers to build the generator, based on *differentiable digital signal processing* (DDSP) [11] we propose a novel and light-weight differentiable EQ and fader layers as the core of our generator. This provides a strong inductive bias as the job of the generator now boils down to determining the parameters of the EQ and fader to be applied respectively to the paired tracks. While DDSP-like components has been applied to audio synthesis [11,12], audio effect modeling [13–15], and automatic mixing [16], they have not been used for transition generation, to our best knowledge. We refer to our model as DJtransGAN.

For model training, we develop a data generation pipeline to prepare input paired tracks from the MTG-Jamendo dataset [17], and collect real-world mixes from *livetracklist* [18]. Examples of the generated mixes and source code can be found at `https://paulyuchen.com/djtransgan-icassp2022/`.

2. DIFFERENTIABLE DJ MIXER

EQs and faders are two essential components in the DJ-made mixing effects. To make an automatic DJ mixer that is both differentiable and is capable of producing mixing effects, we incorporate DDSP components that resemble EQs and faders in our network.

We consider audio segments of paired tracks (x_1, x_2), whose lengths have been made identical by zero-padding x_1 at the back after C_{out} and zero-padding x_2 at the front before C_{in}. We denote their STFT spectrograms as $S_1, S_2 \in \mathbb{R}^{T\times F}$, in which they have T frames and F frequency bins. Two time-frequency masks $M_1(\theta_1), M_2(\theta_2) \in [0.0, 1.0]^{T\times F}$, parameterized by θ_1 and θ_2, are used to represent the mixing effects to be applied to S_1 and S_2. The effects are applied via element-wise product, i.e. $\hat{S}_1 = S_1 \odot M_1(\theta_1)$

∗ Work done during an internship at Sony Group Corporation.

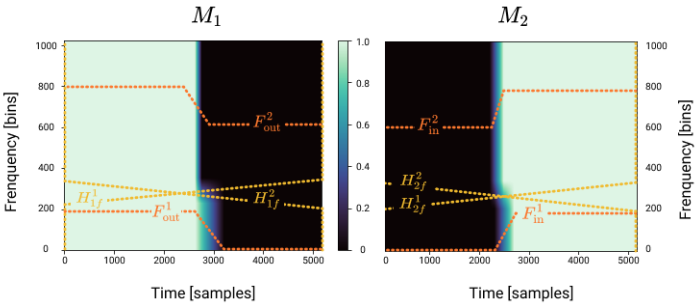

Fig. 2. Differentiable DJ Mixer, illustrating a 2-band EQ.

and $\hat{S}_2 = S_2 \odot M_2(\theta_2)$. The two time-frequency masks $M_1(\theta_1)$ and $M_2(\theta_2)$ represent the DJ mixing effect of fade-out and fade-in, respectively. The goal is to generate $M_1(\theta_1)$ and $M_2(\theta_2)$ such that we can get a proper DJ mix by reverting $\hat{S}_1, \hat{S}_2$ back to the time domain and take their summations as the output $\hat{x}_3$.

2.1. Differentiable Fader

Being inspired by [19,20], we combine two clipped ReLU functions as the basic template of a fading curve, called $PReLU1$. We parameterize $PReLU1$ by its starting time and the slope of its transition region, as formulated below:

$$\begin{aligned} &PReLU1(\vec{v_t}, s_t, \delta_t) = \\ &\min(\max(0, \min(\max(0, \vec{v_t} - s_t), 1) * \delta_t), 1)\,, \end{aligned} \tag{1}$$

where s_t, δ_t and $\vec{v_t}$ respectively correspond to the starting time, the slope and a linear increasing sequence ranged from 0 to 1 with T points in the time axis. Accordingly, the fading curves for fade-in and fade-out can be formulated as:

$$\begin{aligned} F_{\text{in}}(\vec{v_t}, \theta_t) &= PReLU1(\vec{v_t}, s_t, \delta_t)\,, \\ F_{\text{out}}(\vec{v_t}, \theta_t) &= 1 - PReLU1(\vec{v_t}, s_t, \delta_t)\,. \end{aligned} \tag{2}$$

To ensure that the fading occurs only inside of the transition region, we add two extra parameters C_{in} and C_{out}, where $0 \le C_{\text{in}} \le C_{\text{out}} \le 1$, and constrain that $C_{\text{in}} \le s_t \le C_{\text{out}}$. This is similar to the cue button function in an ordinary DJ mixer. Likewise, we impose $\delta_t \ge \frac{1}{C_{\text{out}} - s_t}$ to prevent the sound volume from reaching the maximum after C_{out}. Note that F_{in} and F_{out} can be solely determined by s_t and δ_t; we refer to F_{in} and F_{out} collectively as θ_t below.

2.2. Differentiable Equalizer

EQs are used to amplify or attenuate the volume of specific frequency bands in a track. Our network achieves the effect of EQ by decomposing audio into sub-bands with several low-pass filters [21], and then using faders to adjust the volume of each sub-band.

There are several ways to implement a differentiable low-pass filter, such as via FIR filters [11], IIR filters [13], or a frequency sampling approach [15, 22]. However, all of them have limitations. FIR filter requires learning a long impulse response in this case. IIR filter is recurrent, which leads to computational inefficiency. The frequency sampling approach is relatively efficient but requires an inverse FFT every mini-batch during model training. To avoid these issues, we propose to calculate loss in the time-frequency domain to reduce the number of inverse FFT, and we apply fade-out curves along the frequency axis to achieve low-pass filtering. This can be formulated as:

$$H_{lpf}(\vec{v_f}, \theta_f) = 1 - PReLU1(\vec{v_f}, s_f, \delta_f)\,, \tag{3}$$

where H_{lpf} is the proposed low-pass filter, which can be thought of as "a fade-out curve in the frequency domain." The two parameters s_f and δ_f serve similar purposes as the cutoff frequency and the Q factor, which are typical parameters of time-domain filters. $\vec{v_f}$ is a linear increasing sequence ranged from 0 to 1 with F points in the frequency axis. Moreover, to constrain the filtering within a proper frequency band, we introduce $f_{\min}$ and $f_{\max}$ and require $f_{\min} \le s_f \le f_{\max}$ and $\delta_f \ge \frac{1}{f_{\max} - s_f}$. As H_{lpf} can be solely parameterized by s_f and δ_f, we refer to them collectively as θ_f.

Following the same light, we decompose an input track to in total k sub-bands, and we combine multiple low-pass filters to form a series of fading curves in the frequency domain. We denote the ith fading curve as H^i_f and the ith low-pass filter along with its band limitations as H^i_{lpf}, $f^i_{\min}$ and $f^i_{\max}$, with $f^{i-1}_{\max} \le f^i_{\min} \le f^i_{\max}$. The ith filter implementing the EQ effect, i.e., $H^i_f(v_f, \theta^i_f, \theta^{i-1}_f)$, can be accordingly formulated as below. See Figure 2 for an illustration.

$$H^i_f = \begin{cases} H^i_{lpf}(\vec{v_f}, \theta^i_f)\,, & \text{if } i = 1\,. \\ H^i_{lpf}(\vec{v_f}, \theta^i_f) - H^{i-1}_{lpf}(\vec{v_f}, \theta^{i-1}_f)\,, & \text{if } 1 < i < k\,. \\ 1 - H^{i-1}_{lpf}(\vec{v_f}, \theta^{i-1}_f)\,, & \text{if } i = k\,. \end{cases} \tag{4}$$

2.3. Differentiable Mixer

Now we propose a differentiable mixer, it has k sub-bands, and each sub-band has an EQ with its own fader. This means we have two fading-curves for each sub-band, where one works along the time axis and the other along the frequency axis. We denote F^i_{in} and F^i_{out} as the ith fade-in and fade-out curves along the time axis, and H^i_{1f} and H^i_{2f} as the ith fading curves along the frequency axis. They are parameterized by $\theta_1 : \{\theta^i_{1t}, \theta^i_{1f}, \theta^{i-1}_{1f}\}$ and $\theta_2 : \{\theta^i_{2t}, \theta^i_{2f}, \theta^{i-1}_{2f}\}$.

We construct the final fade-in and fade-out effects, namely $M_1(\theta_1)$ and $M_2(\theta_2)$, by summing up the outer products between the corresponding F^i and H^i_f pairs, as shown below.

$$\begin{aligned} M_1(\theta_1) &= \sum\nolimits_{i=1}^{k} F^i_{\text{out}}(\vec{v_t}, \theta^i_{1t}) \otimes H^i_{1f}(\vec{v_f}, \theta^i_{1f}, \theta^{i-1}_{1f})\,, \\ M_2(\theta_2) &= \sum\nolimits_{i=1}^{k} F^i_{\text{in}}(\vec{v_t}, \theta^i_{2t}) \otimes H^i_{2f}(\vec{v_f}, \theta^i_{2f}, \theta^{i-1}_{2f})\,. \end{aligned} \tag{5}$$

In sum, given C_{in}, C_{out}, $f_{\min}$, and $f_{\max}$, the proposed differentiable mixer has $2 * (k-1) * 4$ trainable parameters, encompassing θ_1 and θ_2. The remaining challenge is to learn proper parameters for the differentiable mixer to generate feasible DJ mixing effects.

3. METHODOLOGY

3.1. Dataset

To train our model, we collect a dataset of real-world mixes, and another dataset of paired tracks with artificially generated cue points.

Real DJ transition data. We obtain a collection of DJ mixes by crawling *livetracklist*, a website hosting DJ mixes [18]. We select 284 mixes that use the *mix* tag and based on human-annotated boundaries, and segment the mixes into individual tracks and regard two consecutive tracks as a music section that contains a DJ transition. Sections shorter than 1 minute are discarded. We finally retrieve 7,064 music sections in total.

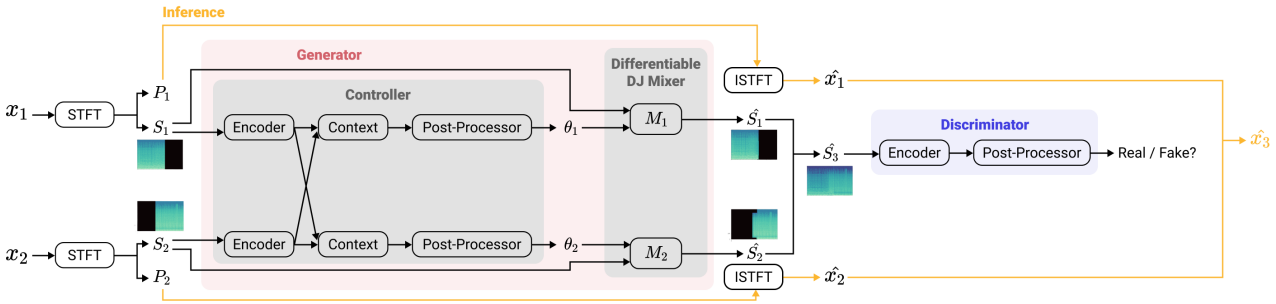

Fig. 3. Schematic plot of the model architecture of the proposed DJtransGAN model for generating DJ transition.

Data generation pipeline. To generate x_1, x_2 and the corresponding $C_{\text{in}}, C_{\text{out}}$, we follow a similar pipeline from [5], which uses most of the domain knowledge of DJ practices. We first compute structure boundaries using the constant-Q transform (CQT)-based "structure feature" algorithm from MSAF [23], and detect the beat, downbeat, tempo, and musical keys by Madmom [24]. Next, we compile a group of segments x_1 by extracting those that are between two structure boundaries and with a length greater than 30 bars of music. For each x_1, we find a suitable x_2 to pair with it by firstly selecting 100 candidates that satisfy 1) a bpm difference with x_1 no greater than five, 2) a key difference with x_1 no greater than two semitones, and 3) originally from a different track. Among the 100 candidate, we identify the best fit x_2 by the highest mixability score (with x_1) as measured by the Music Puzzle Game model [25]. Moreover, we pitch-shift and time-stretch x_2 to match the pitch and tempo of x_1. To fix the transition region to eight bars [3, 7], we set C_{in} and C_{out} as the first downbeat of the last eight bars of x_1 and the last downbeat of the first eight bars of x_2, respectively.

We build a training and a testing corpora respectively with 1,000 tracks and 100 tracks from the songs tagged with *electronic* in the MTG-Jamendo dataset [17]. In total, we generate 8,318 and 830 paired tracks from each corpus.

Preprocessing. We fix all the inputs to 60 seconds at 44.1 kHz sampling rate. For the real mix dataset, the input is centered to the human-annotated boundary sample. For the data from the generation pipeline, we compute the "cue-mid" points C_{mid} as $(C_{\text{in}} + C_{\text{out}})/2$. When needed, music sections are zero-padded symmetrically.

3.2. Model Architecture

The proposed DJtransGAN model consists of a generator G and discriminator D, as shown in Figure 3. We use S_1 and S_2 as the input features for both G and D during training. The architecture of G follows the controller network of the Differentiable Mixing Console proposed by Steinmetz *et al.* [16]. The goal is to learn θ_1 and θ_2, the parameters of the the differentiable DJ mixer.

Each controller network *encoder* learns a feature map from the respective input S_1 or S_2. We stack the resulting feature maps and feed them to each context block. The *context* blocks downscale the output channels from two to one and then directly feed them to the post-processors to predict θ_1 and θ_2. S_1, θ_1 and S_2, θ_2 are fed to the differentiable DJ mixer to get $\hat{S}_1$ and $\hat{S}_2$. The input to D corresponds to the summation of $\hat{S}_1$ and $\hat{S}_2$.

The encoder applies a log-scaled Mel-spectrogram layer followed by three residual convolutional blocks. The residual convolutional blocks contain two convolutional layers with a filter size of 3x3 and strides of 2 and 1, respectively. The three residual convolutional blocks have 4, 8, and 16 filters, respectively. Besides, ReLU and batch normalization are applied after all convolutional layers.

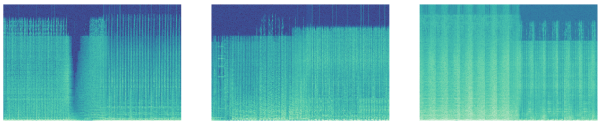

Fig. 4. The STFT spectrograms of some random mixes $(\hat{S}_3)$ generated by the DJtransGAN model.

The context block contains a convolutional layer with one filter of size 1x1. The purpose of this block is to provide sufficient cross-information to each post-processor when predicting θ_1 and θ_2. The post-processor contains three MLP layers. Leaky ReLU and batch normalization are applied after all layers except the last MLP layer, where the Sigmoid function is used. The output dimension of each MLP layer is 1,024, 512, and $2*(k-1)*4$, where k is the number of bands in the differentiable DJ mixer. The encoder in D is identical to that in the controller network. Similarly, the post-processor in D is the same, although the dimension of the last MLP layer is set to 2.

3.3. Training and Inference

In our implementation, we use 128-bin log-scaled Mel-spectrogram computed by using a 2,048-point Hamming window and 512-point hop-size for STFT. We set the differentiable DJ mixers' k as 4, where we set H_f's f_{min} as 20, 300, 5,000 Hz and f_{max} as 300, 5,000 and 20,000 Hz, to focus on a low, mid, high-mid and high frequency, respectively. Overall, G is trained to use the differentiable DJ mixer by controlling 24 parameters. We choose a min-max loss [10] as our training objective and train it with the Adam optimizer for 5,298 steps, batch size of 4, and learning rate 1e$-$5 for both G and D.

As the final step, we apply inverse STFT using the phases P_1 and P_2 from the inputs x_1 and x_2, as shown in Figure 3. We obtain $\hat{x_1}$ and $\hat{x_2}$ as a result and sum them together to get $\hat{x_3}$, i.e., the waveform of the generated transition. Figure 4 shows examples of $\hat{S}_3$.

4. SUBJECTIVE EVALUATION

We compare our model *GAN* with the following four baselines.

- *Sum* is a summation of x_1, x_2 without any effects.
- *Linear* applies a linear cross-fading in the transition region which is a common practice in automatic DJ systems [6, 7].
- *Rule* consists of a set of general-purpose decision rules that we devised after consulting with expert DJs. Depending on

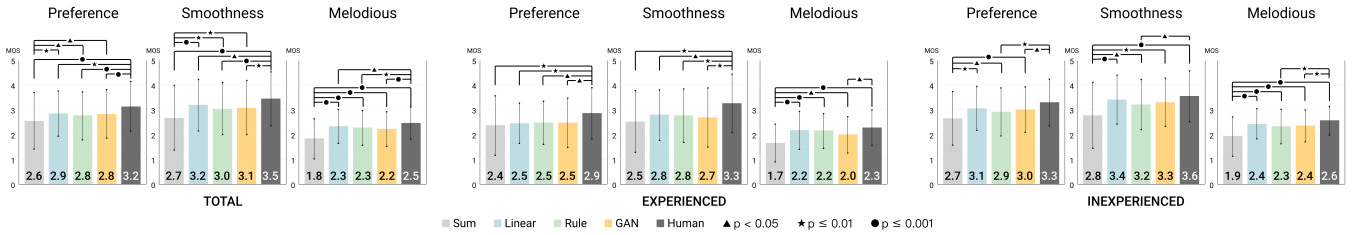

Fig. 5. Result of the subjective evaluation for (left) total people, people (middle) experienced and (right) inexperienced in DJing.

the presence or absence of vocals, *Rule* applies one of the following four *transitions types*; vocal to vocal (V-V), non-vocal to vocal (NV-V), vocal to non-vocal (V-NV) and non-vocal to non-vocal (NV-NV). For each transition type, a specific fading pattern and EQ preset is applied.

- *Human*: lastly, the second author of the paper, who is an ameature DJ, creates the transitions by hand to serve as a possible performance upper bound.

For fair comparison, *Rule* and *Human* use a 4-band EQ. For all methods and for each pair x_1, x_2, the same C_{in} and C_{out} are used.

We include eight groups of paired tracks from the testing set, two for each transition type (e.g., V-V) mentioned above. We conduct the listening test via an online questionnaire. Participants are directed to a listening page containing the mixes of the five methods for one random group of paired tracks. We first ask them to identify the actual timestamps when they notice the transition, and then to asses each test sample on a five-point Likert scale (1–5) in terms of *Preference* and *Smoothness*, and on a three-point Likert scale (1–3) in terms of *Melodious* (to further test whether the transition contains conflicting sounds). Participants can also leave comments on each test sample. Finally, we ask them to specify the mix they like the most. We ask this last question mainly to test consistency with the given *Preference* ratings of the individual mixes.

Figure 5 shows the mean opinion score (MOS). We discard participants whose answer to the first question is not within the transition region, and those whose answer to the last question shows inconsistency. 136 out of the 188 participants met the requirements. They self-reported their gender (105 male), age (17-58), and experience of DJing (46 experienced). A Wilcoxon signed rank test [26] as performed to statistically assess the comparative ability of the five approaches. Overall, the responses indicated an acceptable level of reliability (Cronbach's α =0.799 [27]).

Experienced subjects tend to score lower than inexperienced subjects for the transitions. *Human* receives the highest ratings as expected, but there is no statistically significant difference among the methods with respect to Melodious. *Linear*, *Rule*, and *GAN* are significantly better than *Sum* for Melodious. Inexperienced subjects also rate *Human* the highest, but *Human* is not significant better than *Linear* in Preference and Melodious. The difference between *Human* and either *Linear* or *GAN* is also not significant in Smoothness. *Sum* receives the lowest ratings as expected.

We find that some of the experienced subjects tended to give low scores to all transitions. Based on the comments, we infer that the music style of the selected paired tracks is unacceptable to them and therefore greatly influences their ratings. On the other hand, this was not observed in the inexperienced subjects. Overall, there is no significant difference among *GAN*, *Linear* and *Rule*, suggesting that our *GAN* approach can achieve competitive performance compared to the baselines except for *Human*.

DJ transitions can be considered an art [3, 6] and therefore a highly subjective task whose result cannot be objectively categorized as correct or incorrect. This is similar to the field of automatic music mixing [28], where an exploration of this issue has led to an analysis of the participants' listening test comments [29].

Following this idea, we collect in total 150 comments from 58 subjects (24 experienced). According to the comments, a large part of the experienced subjects indicate that the paired tracks are not suitable for each other and that the decision of the selected structure and cue points is erroneous, suggesting room for improving the data generation pipeline. In addition, several comments indicate that these subjects rate *Linear*, *Rule* and *Human* higher because they can recognize the mixing technique being used. In contrast, they are unfamiliar with possible techniques employed by our *GAN*, which may imply that *GAN* creates its own style by integrating multiple DJ styles from real-world data. Nevertheless, in-depth analysis of the learned mixing style is needed in the future. Finally, some experienced subjects commend that *GAN* makes good use of the high-pass filter when mixing vocals and background music, especially in the V-NV transition type. The mixes from *GAN* may not be perfect, but they feel more organic than the others (except for the one by *human*). People also criticize *GAN* for making the transition too fast within the mix. These kinds of fast transitions are in particular unpleasant for the V-V type of transition.

5. CONCLUSION AND FUTURE WORK

In this paper, we have presented a data-driven and adversarial approach to generate DJ transitions by machine. We have developed a data generation pipeline and proposed a novel differentiable DJ mixer for EQs and loudness faders. Differentiable EQ is achieved in the time-frequency domain by using trainable fade-out curves along the frequency axis, which are based on the frequency response of low-pass filters. Our method is an alternative to differentiable FIR and IIR filters, although we do not have space to present an empirical performance comparison against these alternatives. We have also conducted a subjective listening test and showed that our model is competitive with baseline models. While not reaching human-level quality, our method shows the feasibility of GANs and differentiable audio effects when performing audio processing tasks. This has potentials to be applied in other tasks such as automatic music mixing and mastering, audio effect modeling, and music synthesis.

As future work, the quality of the data generation pipeline can be improved, especially the structure segmentation and segment pairing parts. The cue points selection process could also be learned instead of using fixed cue points. Global tempo matching can be replaced by individual beat matching. Furthermore, an analysis of the transitions learned by the model and an exploration of comprehensive evaluation metrics can also be explored.

6. REFERENCES

[1] Diemo Schwarz and Dominique Fourer, "Methods and datasets for DJ-mix reverse engineering," in *Proc. International Symposium on Computer Music Multidisciplinary Research (CMMR)*, 2019.

[2] Taejun Kim et al., "A computational analysis of real-world DJ mixes using mix-to-track subsequence alignment," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[3] Taejun Kim, Yi-Hsuan Yang, and Juhan Nam, "Reverse-engineering the transition regions of real-world DJ mixes using sub-band analysis with convex optimization," in *Proc. International Conference on New Interfaces for Musical Expression (NIME)*, 2021.

[4] Yu-Siang Huang, Szu-Yu Chou, and Yi-Hsuan Yang, "DJnet: A dream for making an automatic DJ," in *Proc. International Society for Music Information Retrieval Conference (ISMIR), Late-Breaking Paper*, 2017.

[5] Len Vande Veire and Tijl De Bie, "From raw audio to a seamless mix: Creating an automated DJ system for drum and bass," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2018, no. 1, pp. 1–21, 2018.

[6] Rachel M. Bittner et al., "Automatic playlist sequencing and transitions," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[7] Adrian Kim et al., "Automatic DJ mix generation using highlight detection," in *Proc. International Society for Music Information Retrieval Conference (ISMIR), Late-Breaking Paper*, 2017.

[8] Viktor Kronvall, "Mixing music using deep reinforcement learning," M.S. thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2019.

[9] Diemo Schwarz, Daniel Schindler, and Severino Spadavecchia, "A heuristic algorithm for dj cue point estimation," in *Sound and Music Computing*, 2018.

[10] Ian Goodfellow et al., "Generative adversarial nets," *Proc. Advances in neural information processing systems*, vol. 63, no. 11, pp. 139–144, 2014.

[11] Jesse Engel et al., "DDSP: Differentiable digital signal processing," in *Proc. International Conference on Learning Representations (ICLR)*, 2020.

[12] Juan Alonso and Cumhur Erkut, "Latent space explorations of singing voice synthesis using DDSP," in *Proc. Sound and Music Computing Conference (SMC)*, 2021.

[13] Boris Kuznetsov, Julian D Parker, and Fabián Esqueda, "Differentiable iir filters for machine learning applications," in *Proc. Int. Conf. Digital Audio Effects*, 2020.

[14] Marco A. Martínez Ramírez et al., "Differentiable signal processing with black-box audio effects," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[15] Shahan Nercessian et al., "Lightweight and intepretable neural modeling olf an audio distortion effect using hyperconditioned differentiable biquads," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 890–894.

[16] Christian J Steinmetz et al., "Automatic multitrack mixing with a differentiable mixing console of neural audio effects," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, 2021.

[17] Dmitry Bogdanov et al., "The MTG-Jamendo dataset for automatic music tagging," in *Proc. Machine Learning for Music Discovery Workshop, International Conference on Machine Learning (ICML)*, 2019.

[18] "Livetracklist," `https://www.livetracklist.com/`.

[19] Mathieu Ramona and Gaël Richard, "A simple and efficient fader estimator for broadcast radio unmixing," in *Proc. Digital Audio Effects (DAFx)*, 2011, pp. 265–268.

[20] Andrew G Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[21] Haohe Liu et al., "Channel-wise subband input for better voice and accompaniment separation on high resolution music," in *Proc. Interspeech 2020*, 2020, pp. 1241–1245.

[22] Shahan Nercessian, "Neural parametric equalizer matching using differentiable biquads," in *Proc. Int. Conf. Digital Audio Effects (eDAFx-20)*, 2020, pp. 265–272.

[23] Oriol Nieto and Juan Pablo Bello, "Systematic exploration of computational music structure research," in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[24] Sebastian Böck et al., "madmom: a new Python Audio and Music Signal Processing Library," in *Proc. ACM International Conference on Multimedia*, 2016.

[25] Yu-Siang Huang, Szu-Yu Chou, and Yi-Hsuan Yang, "Generating music medleys via playing music puzzle games," in *Proc. AAAI*, 2018.

[26] Robert F. Woolson, "Wilcoxon signed-rank test," *Wiley Encyclopedia of Clinical Trials*, 2007.

[27] Lee J. Cronbach, "Coefficient alpha and the internal structure of tests," *Psychomerika*, 1951.

[28] Brecht De Man et al., "Perceptual evaluation of music mixing practices," in *Proc. Audio Engineering Society Convention*, 2015.

[29] Brecht De Man and Joshua D Reiss, "Analysis of peer reviews in music production," *Journal on the Art of Record Production*, 2015.