Ⓞ Discover

CycleDRUMS: automatic drum arrangement for bass lines using CycleGAN

Giorgio Barnabò[1] · Giovanni Trappolini[1] · Lorenzo Lastilla[1] · Cesare Campagnano[2] · Angela Fan[3] · Fabio Petroni[3] · Fabrizio Silvestri[1]

Abstract

The two main research threads in computer-based music generation are the construction of *autonomous music-making systems* and the design of *computer-based environments to assist musicians*. In the symbolic domain, the key problem of automatically arranging a piece of music was extensively studied, while relatively fewer systems tackled this challenge in the audio domain. In this contribution, we propose CycleDRUMS, a novel method for generating drums given a bass line. After converting the waveform of the bass into a mel-spectrogram, we can automatically generate original drums that follow the beat, sound credible, and be directly mixed with the input bass. We formulated this task as an *unpaired image-to-image translation* problem, and we addressed it with CycleGAN, a well-established unsupervised style transfer framework designed initially for treating images. The choice to deploy raw audio and mel-spectrograms enabled us to represent better how humans perceive music and to draw sounds for new arrangements from the vast collection of music recordings accumulated in the last century. In the absence of an objective way of evaluating the output of both generative adversarial networks and generative music systems, we further defined a possible metric for the proposed task, partially based on human (and expert) judgment. Finally, as a comparison, we replicated our results with Pix2Pix, a paired image-to-image translation network, and we showed that our approach outperforms it.

Keywords Automatic music arrangement · Cycle-GAN · Deep learning · Source separation · Audio and speech processing

1 Introduction

The development of home music production has brought significant innovations into the process of pop music composition. Software like Pro Tools, Cubase, and Logic—as well as MIDI-based technologies and digital instruments—provide a broad set of tools to manipulate recordings and simplify the composition process for artists and producers. After recording a melody, with the aid of a guitar or a piano, songwriters can start building up the arrangement one piece at a time, sometimes not needing professional musicians or proper music training. As a result, singers and songwriters—as well as producers—have started asking for tools that could facilitate, or to some extent even automate, the creation of full songs around their lyrics and melodies. To meet this new demand, the goal of designing computer-based environments

✉ Giorgio Barnabò, barnabo@diag.uniroma1.it; Giovanni Trappolini, trappolini@diag.uniroma1.it; Lorenzo Lastilla, lastilla@ diag.uniroma1.it; Cesare Campagnano, campagnano@di.uniroma1.it; Angela Fan, angelafan@fb.com; Fabio Petroni, fabiopetroni@ fb.com; Fabrizio Silvestri, fsilvestri@diag.uniroma1.it | [1]Department of Computer Engineering, Sapienza, University of Rome, Rome, Italy. [2]Department of Computer Science, Sapienza, University of Rome, Rome, Italy. [3]FAIR, META, London, UK.

to assist human musicians has become central in the field of automatic music generation [1]. IRCAM [2], some examples are Sony CSL-Paris FlowComposer [3], and Logic Pro X Easy Drummer. In addition, more solutions based on deep learning techniques, such as RL-Duet [4]—a deep reinforcement learning algorithm for online accompaniment generation—or PopMAG, a transformer-based architecture which relies on a multi-track MIDI representation of music [5], continue to be studied. A comprehensive review of the most relevant deep learning techniques applied to music is provided by Briot et al. [1].

Unlike most techniques that rely on a symbolic representation of music (i.e., MIDI, piano rolls, music sheets), the approach proposed in this paper is a first attempt at automatically generating drums in the audio domain, given a bass line encoded in the mel-spectrogram time-frequency domain. As extensively shown in Sect. 2, mel-spectrograms are already commonly and effectively used in many music information retrieval tasks [6]. Nonetheless, music generation models applied to this intermediate representation are still relatively scarce. Although arrangement generation has been extensively studied in symbolic audio, switching to mel-spectrograms allowed us to preserve the sound heritage of other musical pieces and represent a valid alternative for real-case scenarios. Indeed, even if it is possible to use synthesizers to produce sounds from symbolic music, MIDI, music sheets, and piano rolls are not always easy to find or produce, and they sometimes lack expressiveness. Moreover, state-of-the-art synthesizers cannot yet reproduce the infinite nuances of authentic voices and instruments, whereas raw audio representation guarantees more flexibility and requires little music competence. On the other hand, thanks to this two-dimensional time-frequency representation of music based on mel-spectrograms, we can treat the problem of automatically generating an arrangement or accompaniment for a specific musical sample equivalent as an image-to-image translation task. For instance, if we have the mel-spectrogram of a bass line, we may want to produce the mel-spectrogram of the same bass line together with suitable drums.

To solve this task, we tested an unpaired image-to-image translation strategy known as CycleGAN [7]. In particular, we trained a CycleGAN architecture on 5s bass and drum samples (equivalent to 256×256 mel-spectrograms) coming from both the Free Music Archive (FMA) dataset [8] and the musdb18 dataset [9]. The short sample duration does not affect the proposed methodology, at least concerning the arrangement task we focus on, and inference could also be performed on longer sequences. Since the FMA songs lack source-separated channels (i.e., differentiated vocals, bass, drums), it was pre-processed first. The required channels were extracted using Demucs [10]. The results were then compared to Pix2Pix [11], another popular paired image-to-image translation network. To sum up, our main contributions are the following:

- we trained a CycleGAN architecture on bass and drum mel-spectrograms in order to automatically generate drums that follow the beat and sound credible for any given bass line;
- our approach can generate drum arrangements with low computational resources and limited inference time, if compared to other popular solutions for automatic music generation [12];
- we developed a metric—partially based on or correlated to human (and expert) judgment—to automatically evaluate the obtained results and the creativity of the proposed system, given the challenges of a quantitative assessment of music;
- we compared our method to Pix2Pix, another popular image transfer network, showing that the music arrangement problem can be better tackled with an unpaired approach and adding a cycle-consistency loss.

To the best of our knowledge, we are the first to exploit cycle-consistent adversarial networks and a two-dimensional time-frequency representation of music for automatically generating suitable drums given a bass line.

2 Related works

The interest in automatic music generation, translation, and arrangement has dramatically increased in the last few years, as proven by the many proposed solutions—see [1] for a comprehensive and detailed survey. Here we present a brief overview of the key contributions in the symbolic and audio domains.

Music generation & arrangement in the symbolic domain: there is an extensive body of research using symbolic music representation to perform music generation and arrangement. The following contributions used MIDI, piano rolls, chord and note names to feed several deep learning architectures and tackle different aspects of the music generation problem. In [13], CNNs are used for generating melody as a series of MIDI notes either from scratch, by following a chord sequence, or by conditioning on the melody of previous bars, whereas in [14–17] LSTMs are used to generate musical notes, melodies, polyphonic music pieces, and long drum sequences under constraints imposed by metrical rhythm information and

a given bass sequence. The authors of [18–20] instead use a variational recurrent auto-encoder to generate melodies. In [21], symbolic sequences of polyphonic music are modeled in an entirely general piano-roll representation, while the authors of [22] propose a novel architecture to generate melodies satisfying positional constraints in the style of the soprano parts of the J.S. Bach chorale harmonizations encoded in MIDI. In [23], RNNs are used for the prediction and composition of polyphonic music; in [24], highly convincing chorales in the style of Bach were automatically generated using note names [25]; added higher-level structure on generated polyphonic music, whereas in [26] an end-to-end generative model capable of composing music conditioned on a specific mixture of composer styles was designed. The approach described in [27], instead, relies on notes as an intermediate representation to a suite of models—namely, a transcription model based on a CNN and an RNN network [28], a self-attention-based music language model [29] and a WaveNet model [30]—capable of transcribing, composing, and synthesizing audio waveforms. Finally [31], proposes an end-to-end melody and arrangement generation framework called XiaoIce Band, which generates a melody track with multiple accompaniments played by several types of instruments.

Music generation & arrangement in the audio domain: some of the most relevant approaches proposed in waveform music generation deal with raw audio representation in the time domain [32]. Many of these approaches draw methods and ideas from the extensive literature on audio and speech synthesis [33, 34]. For instance, in [35], a flow-based network capable of generating high-quality speech from mel-spectrograms is proposed. In contrast, in [36], the authors present a neural source-filter (NSF) waveform modeling framework that is straightforward to train and fast to generate waveforms. In [37], recent neural waveform synthesizers such as WaveNet, WaveG-low, and a neural source filter (NSF) are compared. Mehri et al. [38] tested a model for unconditional audio synthesis based on generating one audio sample at a time, and [39] applied Restricted Boltzmann Machine and LSTM architectures to raw audio files in the frequency domain in order to generate music. In contrast, the authors of [30] propose a fully probabilistic and auto-regressive model, with the predictive distribution for each audio sample conditioned on all previous ones, to produce novel and often highly realistic musical fragments. The authors of [40] present a raw audio music generation model based on the WaveNet architecture, which takes the composition notes as a secondary input. The authors of [41] instead propose a transformer vq-vae model to generate a drum truck that accompanies a user-provided drum-free free recording. Finally, in [12], the authors tackled the long context of raw audio using a multi-scale VQ-VAE to compress it to discrete codes and modeled such context through Sparse Transformers, in order to generate music with singing in the raw audio domain. Nonetheless, due to the computational resources required to model long-range dependencies in the time domain directly, either short samples of music can be generated or complex and large architectures and long inference time are required. On the other hand, in [42], authors discuss a novel approach that proves that long-range dependencies can be more tractably modeled in two-dimensional time-frequency representations such as mel-spectrograms. More precisely, the authors of this contribution designed a highly expressive probabilistic model and a multi-scale generation procedure over mel-spectrograms capable of generating high-fidelity audio samples which capture structure at timescales. It is worth recalling, as well, that treating spectrograms as images is the current standard for many Music Information Retrieval tasks, such as music transcription [43], music emotion recognition [44], and chord recognition.

Generative adversarial networks for music generation: such two-dimensional representation of music paves the way for applying several image processing techniques and image-to-image translation networks to carry out style transfer and arrangement generation [7, 11]. It is worth recalling that the application of GANs to music generation tasks is not new: in [45], GANs are applied to symbolic music to perform music genre transfer, while in [46, 47], authors construct and deploy an adversary of deep learning systems applied to music content analysis; however, to the best of our knowledge, GANs have never been applied to raw audio in the mel-frequency domain for music generation purposes. As to the arrangement generation task, the large majority of approaches proposed in the literature is based on a symbolic representation of music: in [5], a novel multi-track MIDI representation (MuMIDI) is presented, which enables simultaneous multi-track generation in a single sequence and explicitly models the dependency of the notes from different tracks utilizing a Transformer-based architecture; in [4], a deep reinforcement learning algorithm for online accompaniment generation is described.

Coming to the most relevant issues in the development of music generation systems, both the training and evaluation of such systems have proven challenging, mainly because of the following reasons: (i) the available datasets for music generation tasks are challenging due to their inherent high-entropy [48], and (ii) the definition of an objective metric and loss is a common problem to generative models such as GANs: as of now, generative models in the music domain are evaluated based on the subjective response of a pool of listeners, because an objective metric for the raw audio representation has never been proposed so far. Just for the MIDI representation, a set of simple musically informed objective metrics was proposed [49].

3 Method

We present CycleDRUMS, a novel approach for automatically adding credible drums to bass lines based on an adversarially trained deep learning model.

3.1 Source separation for music

A key challenge to our approach is the scarce availability of music data featuring source-separated channels (i.e., differentiated vocals, bass, drums). To this end, we leverage Demucs by [10], a freely available tool that separates the music into its generating sources. Demucs is an extension to Conv-Tasnet [50], purposely adapted to the field of music source separation. It features a U-NET encoder–decoder architecture with a bidirectional LSTM as hidden layer. In particular, we exploited the authors' pre-trained model consisting of 6 convolutional encoder and decoder blocks and a hidden size of length 3200. Thanks to the randomized equivariant stabilization, Demucs is time-equivariant, meaning that any shifts in the input mixture will cause congruent shifts in the output.

However, a potential weakness of this method is that it sometimes produces noisy separations, with watered-down harmonics and traces of other instruments in the vocal segment. The usage of Demucs could hinder our pipeline from properly recognizing and reconstructing the accompaniment, where the harmonics play a critical part. Nonetheless, even if better source-separation methods are available, achieving slightly higher values of signal-to-distortion ratio (SOTA SDR = 5.85, Demucs SDR = 5.67), we chose to use Demucs because it is faster and easier to embed in our pipeline. Moreover, Demucs outperforms the current state of the art for bass source separation [SOTA SDR = 5.28, Demucs SDR = 6.21], as seen in Table 1 from [10].

Thanks to Demucs, we were at least partially able to solve the challenge of data availability and feed our model with appropriate signals. In practice, given an input song, we use Demucs to separate it into vocals, bass, drums, and others, keeping the original mixture.

3.2 Music representation—from raw audio to mel-spectrograms

Our method's distinguishing feature is mel-spectrograms instead of waveforms. Namely, we opted for a two-dimensional time-frequency representation of music rather than a time representation. The spectrum is a common transformed representation for audio, obtained via a Short-Time Fourier transform (STFT) [51]. The discrete STFT of a given signal $x : [0 : L-1] := \{0, 1, \ldots, L-1\} \to \mathbb{R}$ leads to the kth complex Fourier coefficient for the mth time frame:

$$\mathcal{X}(m, k) := \sum_{n=0}^{N-1} x(n + mH) \cdot w(n) \cdot e^{-\frac{2\pi i k n}{N}}$$

With $m \in [0 : M]$ and $K \in [0 : K]$, and where $w(n)$ is a sampled window function of length $N \in \mathbb{N}$, and $H \in \mathbb{N}$ is the hop size that determines the step size the window is to be shifted across the signal [51]. The spectrogram is a two-dimensional representation of the squared magnitude of the STFT, i.e., $\mathcal{Y}(m, k) := \|\mathcal{X}(m, k)\|^2$, with $m \in [0 : M]$ and $K \in [0 : K]$. Figure 1 shows an example of a mel-spectrogram [52] that is treated as a single channel image, representing the sound intensity with respect to time—x-axis—and frequency—y axis [1]. This decision allows us to better deal with long-range dependencies, typical of such kind of data, and to reduce the computational resources and inference time required. Moreover, the mel-scale is based on a mapping between the actual frequency f and perceived pitch $m = 2595 \cdot log_{10}(1 + \frac{f}{700})$, as the human auditory system does not perceive pitch in a linear manner. Finally, using mel-spectrograms of pre-existing songs to train our model potentially enables us to draw sounds for new arrangements from the vast collection of music recordings accumulated in the last century. It is worth recalling that mel-frequency cepstral coefficients are the dominant features used in speech recognition and many music modeling tasks [53].

After the source separation task is carried out on our song dataset, both the bass and drum waveforms are turned into the corresponding mel-spectrograms using PyTorch Audio.[1] PyTorch works very fast and is optimized to perform robust GPU-accelerated conversion. In addition, to reduce the dimensionality of the data, we decided to keep only the magnitude coefficients, discarding the phase information. Finally, to revert the generated mel-spectrograms to the

---

[1] Available at: https://pytorch.org/audio/stable/index.html.

Pre-training pipeline

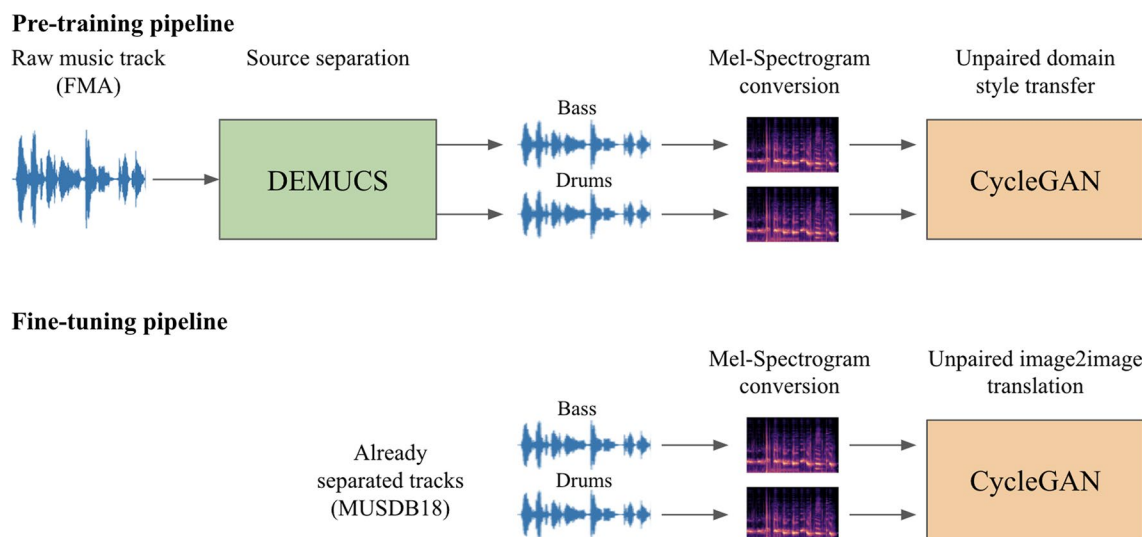**Fine-tuning pipeline**

Fig. 1 To solve the automatic drum arrangement task, we tested an unpaired image-to-image translation strategy known as CycleGAN [7]. In particular, we trained a CycleGAN architecture on 5s bass and drum samples (equivalent to 256×256 mel-spectrograms) coming from both the Free Music Archive (FMA) dataset [8] and the musdb18 dataset [9]. Since the FMA songs lack source-separated channels (i.e., differentiated vocals, bass, drums, etc.), the bass and drum channels were extracted using Demucs [10]. After the source separation task on our song dataset, the bass and drum waveforms are turned into the corresponding mel-spectrograms. The sampling rate was set to 22050 Hz, the window length N to 2048, the number of mel-frequency bins to 256, and the hop size H to 512. To fit our model requirements, we cropped out 256×256 windows from each mel-spectrogram with an overlapping of 50-time frames, obtaining multiple samples from each song (each roughly equivalent to 5 s of music). Since FMA is much larger than musdb18 but also lower quality due to the artificial separation of sources, we used FMA to train the model, and then we fine-tuned it with musdb18, which comes in a source-separated fashion

corresponding time-domain signal: (i) we apply a conversion matrix (using triangular filter banks) that converts the mel-frequency STFT to a linear scale STFT. The matrix is calculated using a gradient-based method [54] that minimizes the Euclidean norm between the original mel-spectrogram and the product between reconstructed spectrogram and filter banks; (ii) we use the Griffin–Lim's algorithm [55] to reconstruct the phase information.

It is worth noticing that the mel-scale conversion and the removal of STFT phases, respectively, discard frequency and temporal information, thus resulting in a distortion in the recovered signal. To minimize this problem, we made use of high-resolution mel-spectrograms [42], whose size can be tweaked with *number of mels* and *STFT hop size* parameters. Thus, here are the hyper-parameters we used: the sampling rate was set to 22050 Hz, the window length N to 2048, the number of Mel-frequency bins to 256, and the hop size H to 512. To fit our model requirements, we cropped out 256×256 windows from each mel-spectrogram with an overlapping of 50-time frames, obtaining multiple samples from each song (each roughly equivalent to 5 s of music).

3.3 Image to image translation—CycleGAN

We cast the automatic drum arrangement generation task as an *unpaired image-to-image translation* task, and we solved it by adapting the CycleGAN model to our purpose. CycleGAN is a framework designed to translate between domains with unpaired input–output examples. The architecture assumes some underlying relationship between domains and tries to learn it. Based on a set of images in the domain X and a different set in the domain Y, the algorithm jointly learns a mapping $G : X \rightarrow Y$ and a mapping $F : Y \rightarrow X$, such that the output $\hat{y} = G(x)$ for every $x \in X$ is indistinguishable from images $y \in Y$, and $\hat{x} = G(y)$ for every $y \in Y$ is indistinguishable from images $x \in X$. Given a mapping $G : X \rightarrow Y$ and another mapping $F : Y \rightarrow X$, then G and F should be one the inverse of the other, and both mappings should be bijections. This property is achieved by training both the mapping G and F simultaneously with a "standard" GAN loss of the form
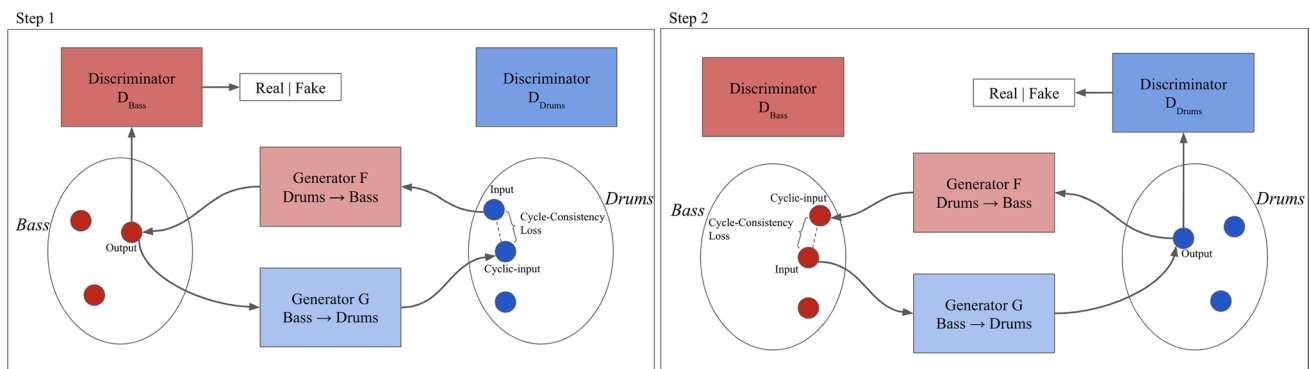
Fig. 2 CycleGAN is a framework designed to translate between domains with unpaired input–output examples. The architecture assumes some underlying relationship between domains and tries to learn it. In our case, based on a set of bass images in the domain X and a set of drum images in the domain Y, the algorithm jointly learns a mapping $G : X \rightarrow Y$—from bass to drums—and a mapping $F : Y \rightarrow X$—from drums to bass, such that the output $\hat{y} = G(x)$ for every $x \in X$ is indistinguishable from images $y \in Y$, and $\hat{x} = G(y)$ for every $y \in Y$ is indistinguishable from images $x \in X$. Given a mapping $G : X \rightarrow Y$ and another mapping $F : Y \rightarrow X$, then G and F should be one the inverse of the other, and both mappings should be bijections. This property is achieved by training both the mapping G and F simultaneously and by adding a cycle-consistency loss that encourages $F(G(x)) \approx x$ and $G(F(y)) \approx y$. Finally, the cycle-consistency loss is combined with the adversarial losses on domains X and Y [7]

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[log D_Y(y)]$$
$$+ \mathbb{E}_{x \sim p_{data}(x)}[log(1 - D_Y(G(x)))],$$

and by adding a cycle-consistency loss that encourages $F(G(x)) \approx x$ and $G(F(y)) \approx y$ according to the following form:

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[\|F(G(x)) - x\|_1]$$
$$+ \mathbb{E}_{y \sim p_{data}(y)}[\|G(F(y)) - y\|_1].$$

Finally, the cycle-consistency loss is combined with the adversarial losses on domains X, and Y [7] to obtain:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{GAN}(G, D_Y, X, Y)$$
$$+ \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda \mathcal{L}_{cyc}(G, F).$$

We adopt the architecture from [56] for our generative networks, which have shown impressive neural style transfer and super-resolution results. For the discriminator networks, we use PathcGANs [11, 57, 58], which aim to classify whether overlapping image patches are real or fakes. Figure 2 shows a schema summarizing the entire architecture.

3.4 Automatic bass to drums arrangement

CycleDRUMS takes as input a set of N music songs in the waveform domain $X = \{\mathbf{x_i}\}_{i=1}^{N}$, where $\mathbf{x_i}$ is a waveform whose number of samples depends on the sampling rate and the audio length. Demucs then separate each waveform into different sources. We only used the bass and drum sources to carry out our experiments. Thus, we ended up having two WAV files for each song, which means a new set of data of the kind: $X_{NEW} = \{\mathbf{d_i}, \mathbf{b_i}\}_{i=1}^{N}$, where $\mathbf{b_i}, \mathbf{d_i}$ represent the bass and drum sources respectively. Each track is then converted into its mel-spectrogram representation.

Since the CycleGAN model takes 256×256 images as input, each mel-spectrogram is chunked into smaller pieces with an overlapping window of 50 time frames, obtaining multiple samples from each song (each equivalent to 5 s of music); finally, in order to obtain one channel images from the original spectrograms, we performed a discretization step in the range [0–255]. In the final stage of our pipeline, we fed CycleGAN architecture with the obtained dataset. Even though the discretization step introduces some distortion—original spectrogram values are floats—the impact on the audio quality is negligible.

At training time, as the model considers two domains, X and Y, we fed the model with drum and bass lines to create credible drums given a bass line. As previously anticipated, this task is an appropriate first step toward fully automated

music arrangement. In the future, for instance, this same approach could be applied to more complex signals, such as voice, guitar, or piano. Nonetheless, we decided to start with drums and bass because they are usually the first instruments to be recorded when producing a song. Their signals are relatively simple compared to more nuanced and harmonic-rich instruments.

4 Experiments

4.1 Dataset

It is important to carefully pick the dataset for the quality of the generated music samples. To train and test our model, we decided to use the Free Music Archive[2] (FMA), and the musdb18[3] dataset [9] that were both released in 2017. The Free Music Archive (FMA) is the largest publicly available dataset suitable for music information retrieval tasks [8]. In its full form, it provides 917 GB and 343 days of Creative Commons-licensed audio from 106,574 tracks, 16,341 artists, and 14,854 albums, arranged in a hierarchical taxonomy of 161 unbalanced genres. Songs come with full-length and high-quality audio and pre-computed features, together with track- and user-level metadata, tags, and free-form text such as biographies. Given the size of FMA, we chose to select only untrimmed songs tagged as either pop, soul-RnB, or indie-rock, for approximately 10,000 songs ($\approx$ 700 h of audio). It is possible to read the full list of songs at the FMA's website,[4] selecting the genres. We discarded all live-recorded songs by filtering out all albums containing the word "live" in the title. Finally, to better validate and fine-tune our model, we decided also to use the full musdb18 dataset. This rather small dataset comprises 100 tracks taken from the DSD100 dataset, 46 tracks from the MedleyDB, two tracks kindly provided by Native Instruments, and two tracks from the Canadian rock band The Easton Ellises. It represents a unique and precious source of songs delivered in a multi-track fashion. Each song comes as five audio files—vocals, bass, drums, others, and full song—perfectly separated at the master level. We used the 100 tracks taken from the DSD100 dataset to fine-tune the model ($\approx$ 6.5 h) and the remaining 50 songs to test it ($\approx$ 3.5 h). We remark that DEMUCS introduces artifacts in the separated sources output. For this reason, our training strategy is to pre-train the architecture with the artificially source-separated FMA dataset and then fine-tune it with musdb18. Intuitively, the former, which is much larger, helps the model to create a good representation of the musical signal; the latter, which is of higher quality, reduces the bias caused by the underlying noise and favors the automatic generation of a base relying on the (clean) input given only. We argue that this training procedure effectively alleviates the effects of the artifacts introduced during the source separation process. A large, clean dataset of separated raw-audio sources remains a research objective. To conclude, since mel-spectrograms are trimmed in 256×256 overlapping windows, we ended up with 600,000 train samples and 14,000 test samples. The hop size, 256, was chosen according to recommendations from [35].

4.2 Training of the CycleGAN model

We trained our model on 2 Tesla V100 SXM2 GPUs with 32 GB of RAM for 12 epochs (FMA dataset) and fine-tuned it for 20 more epochs (musdb18 dataset). As a final step, the mel-spectrograms obtained were converted to the waveform domain to evaluate the music produced. As to the CycleGAN model used for training, we relied on the default network[5]. As a result, the model uses a resnet_9blocks ResNet generator and a basic 70×70 PatchGAN as a discriminator. The Adam optimizer [59] was chosen both for the generators and the discriminators, with betas (0.5, 0.999) and a learning rate equal to 0.0002. The batch size was set to 1. The λ weights for cycle losses were both equal to 10.

4.3 Experimental setting

Even though researchers proposed some effective metrics to predict how popular a song will become [60], there is an intrinsic difficulty in objectively evaluating artistic artifacts such as music. As a human construct, there are no

---

[2] https://github.com/mdeff/fma.
[3] https://sigsep.github.io/datasets/musdb.html.
[4] https://freemusicarchive.org/home.
[5] https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix.

Table 1 Pearson's correlation matrix for all 4 annotators

	Guitarist	Drummer	Prod. 1	Prod. 2
Guitarist	NA	0.82	0.75	0.77
Drummer	0.82	NA	0.76	0.79
Prod. 1	0.75	0.76	NA	0.85
Prod. 2	0.77	0.79	0.85	NA

objective, universal criteria for appreciating music. Nevertheless, in order to establish some forms of benchmark and allow comparisons among different approaches, many generative approaches to raw audio, such as Jukebox [12] or Universal Music Translation Network [61], try to overcome this obstacle by having the results manually tagged by human experts. Although this rating may be the best in quality, the result is still somehow subjective. Thus different people may give different or biased ratings based on their tastes. Moreover, the cost and time required to manually annotate the dataset could become prohibitive even for relatively few samples (over 1000). In light of the limits linked to this human-based approach, we propose a new metric that correlates well with human judgment. This could represent a first benchmark for the tasks at hand. The scores remain somehow subjective, as they mirror the evaluators' criteria and grades, but they are obtained based on a fully automatic and standardized approach.

4.4 Metrics

If we consider as a general objective for a system the capacity to assist composers and musicians, rather than to autonomously generate music, we should also consider as an evaluation criteria the satisfaction of the composer, rather than the satisfaction of the auditors [1].

However, as previously stated, an exclusively human evaluation may be unsustainable in terms of cost and time required. Thus we carried out the following quantitative assessment of our model. We first produced 400 samples—from as many different songs and authors—of artificial drums starting from bass lines that were part of the test set. We then asked a professional guitarist who has been playing in a pop-rock band for more than ten years, a professional drummer from the same band, and two pop and indie-rock music producers with more than four years of experience to manually annotate these samples, capturing the following musical dimensions: sound quality, contamination, credibility, and whether the generated drums followed the beat. More precisely, for each sample, we asked them to rate from 0 to 9 the following aspects: (i) *Sound Quality*: a rating from 0 to 9 of the naturalness and absence of artifacts or noise, (ii) *Contamination*: a rating from 0 to 9 of the contamination by other sources, (iii) *Credibility*: a rating from 0 to 9 of the credibility of the sample, (iv) *Time*: a rating from 0 to 9 of whether the produced drums follow the beat of the bass line. The choice fell on these four aspects after we asked the evaluators to list and describe the most relevant dimensions in the perceived quality of drums. The correlation matrix for all four annotators is shown in Table 1.

Ideally, we want to produce some quantitative measure whose outputs—when applied to generated samples—correlate well (i.e., predict) expert average grades. To achieve this goal, we trained a logistic regression model with features obtained by comparing the original and artificial drums. Here are the details on how we obtained suitable features.

STOI-like features: we created a procedure—inspired by the STOI [62]—whose output vector somehow measures the mel-frequency bins correlation throughout the time between the original sample and the fake one. The obtained vector can feed a multi-regression model whose independent variable is the human score attributed to that sample. Here is the formalization:

$$Human\ Score = \sum_i^{256} a_i \left[\sum_t^{256} (x_i^{(t)} - \bar{x}^{(t)})(y_i^{(t)} - \bar{y}^{(t)}) \right]$$

To simplify, to each pair of samples (original and generated one), a 256 element long vector is associated as follows:

$$\mathcal{S}(\mathcal{X}, \mathcal{Y}, l)^{(i)} = \sum_t^{256} (x_i^{(t)} - \bar{x}^{(t)})(y_i^{(t)} - \bar{y}^{(t)})$$

where (i) $\mathcal{X}$ and $\mathcal{Y}$ are, respectively, the mel-spectrogram matrices of original and generated samples; (ii) a_i is the i-th coefficient for the linear regression; (iii) $x_i^{(t)}$ and $y_i^{(t)}$ the i-th element of the t-th column of matrices $\mathcal{X}$ and $\mathcal{Y}$, respectively; (iv) $\bar{x}^{(t)}$ and $\bar{y}^{(t)}$ are the means along the t-th column of matrices $\mathcal{X}$ and $\mathcal{Y}$, respectively. Each feature i of the regression model is a sort of Pearson correlation coefficient between row i of $\mathcal{X}$ and row i of $\mathcal{Y}$ throughout time.

FID-based features: in the context of GANs result evaluation, the Fréchet Inception distance (FID) is supposed to improve on the Inception Score by actually comparing the statistics of generated samples to authentic samples [63]. This metric leverages the established Inception pre-trained model by getting a vector representation of each mel-spectrogram (i.e. each song), and uses these vectors to compare the distributions of generated and gold examples. This is unlike Inception score which only evaluates the distribution of generated images. In other words, FID measures the probabilistic distance between two multivariate Gaussians, where $X_r = N(\mu_r, \Sigma_r)$ and $X_g = N(\mu_g, \Sigma_g)$ are the 2048-dimensional activations of the Inception-v3 pool3 layer—for real and generated samples respectively—modeled as normal distributions. The similarity between the two distributions is measured as follows:

$$FID = ||\mu_r - \mu_g||^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2})$$

Nevertheless, since we want to assign a score to each sample, we just estimated the $X_r = N(\mu_r, \Sigma_r)$ parameters—using different activation layers of the Inception pre-trained network—and then we calculated the probability density associated to each fake sample. Finally, we added these scores to the regression model predictors.

4.5 Baseline

Since, to the best of our knowledge, we are the first to tackle the drum arrangement task in the audio domain and to treat it as an image-to-image translation problem, we a lack of a suitable baseline. Ultimately, instead of forcing a pre-existing method to work in our specific scenario, we decided to replicate our experiments using the Pix2Pix architecture [11], another image-to-image translation network. Unlike CycleGAN, Pix2Pix learns to translate between domains when fed with paired input–output examples. At training time, we relied on the default network provided by the original authors,[6] we ran it on 2 Tesla V100 SXM2 GPUs with 32 GB of RAM for 50 epochs (FMA dataset), and we fine-tuned it for 30 more epochs (musdb18 dataset).

Finally, after training, we produced 400 drum samples from the same bass lines used for generating the test drums that the evaluators graded. We then asked the same four evaluators to grade the new drum samples according to the principles presented in Sect. 4.4.

4.6 Experimental results

Figure 3 shows the distribution of grades for the 400 test drums for both CycleGAN and Pix2Pix—averaged among all four independent evaluators and over all four dimensions. We rounded the results to the closest integer to make the plot more readable. The higher the grade, the better the sample will sound. Additionally, to fully understand what to expect from samples graded similarly, we discussed the model results with the evaluators. We collectively listened to a random set of samples, and it turned out that all four raters followed similar principles in assigning the grades. Samples with grades 0–3 are generally silent or very noisy. In samples graded 4–5, few sounds start to emerge, but they are usually not very pleasant to listen to, nor coherent. Grades 6–7 identify drums that sound good and are coherent but not continuous: they tend to follow the bass line too closely. Finally, samples graded 8 and 9 are almost indistinguishable from real drums in terms of sound and timing. In labeling non-graded samples, we trained a multi-logistic regression model with STOI-like and FID-based features to predict what of these four buckets the graders would assign the sample to. We trained the model on 300 of the overall 400 graded samples, and kept 100 graded samples as a test set. The model accuracy on this test set was 87% for CycleDRUMS and 93% for Pix2Pix.

Given this pretty good result, we could then use this trained logistic model to label 14,000 different 5s fake drum clips produced from as many real bass lines using both CycleGAN and Pix2Pix. Figure 3 shows the distribution of predicted classes for these samples. At this website[7] a private Sound Cloud playlist of some of the most exciting results is available, while at this link[8] we uploaded some samples obtained with the Pix2Pix baseline architecture.

---

[6] https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix.

[7] https://soundcloud.com/user-639025674/sets/bass2drums/s-jjccrgdXXOi.

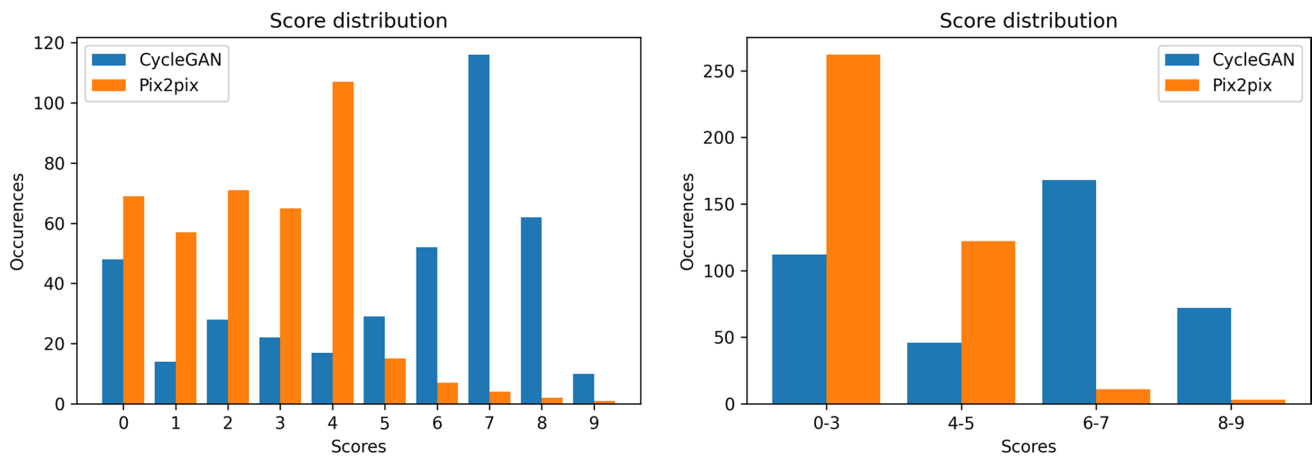[8] https://soundcloud.com/user-639025674/sets/bass2drums_baseline/s-rKqV2S2MFHw.

Fig. 3 (left) The distribution of grades for the 400 test drums for both CycleGAN and Pix2Pix (baseline)—averaged among all four independent evaluators and over all four dimensions. We rounded the results to the closest integer to make the plot more readable. The higher the grade, the better the sample will sound. Samples with grades 0–3 are generally silent or very noisy. In samples graded 4–5, few sounds emerge, but they usually could be more pleasant to listen to and coherent. Grades 6–7 identify drums that sound good and are coherent but not continuous: they tend to follow the bass line too closely. Finally, samples graded 8 and 9 are almost indistinguishable from real drums in terms of sound and timing. (right) In labeling non-graded samples, we, therefore, trained a multi-logistic regression model with both the STOI-like and the FID-based features to predict what of the four grade buckets the graders would assign the sample to. The model accuracy on the test set was 87% for CycleDRUMS and 93% for Pix2Pix. Given this pretty good result, we could then use this trained logistic model to label 14,000 different 5s fake drum clips produced from as many real bass lines using both CycleGAN and Pix2Pix (baseline). The histogram above shows the distribution of the predicted class for these samples

Finally, concerning the computational resources and time required to generate new arrangements, our approach shows several advantages compared to auto-regressive models [12]. Since the output prediction can be fully parallelized, the inference time amounts to a forward pass and a Mel-spectrogram-waveform inverse conversion, whose duration depends on the input length but never exceeds a few minutes. Indeed, it is worth noting that, at inference time, arbitrary long inputs can be processed and arranged. Conversely, this does not apply to other auto-regressive models that can not generate output in a parallel manner at inference time, heavily penalizing computational time that, according to the authors, takes 8 h to generate a 30-s snippet.

5 Conclusions and future work

In this work, we presented a novel approach to automatically producing drums starting from a bass line. We applied CycleGAN to real bass lines, treated as gray-scale images (mel-spectrograms), obtaining good ratings, especially compared to another image-to-image translation approach (Pix2pix). Given the novelty of the problem, we proposed a reasonable procedure to evaluate our model outputs properly. Even with the promising results, some critical issues must be addressed before a more compelling architecture can be developed. First and foremost, a more extensive and cleaner dataset of source-separated songs should be created. Manually separated tracks always contain a big deal of noise. Moreover, the model architecture should be further improved to focus on longer dependencies and consider the actual degradation of high frequencies. For example, our pipeline could be extended to include some recent work on quality-aware image-to-image translation networks [64] and spatial attention generative adversarial networks [65]. Finally, a certain degree of interaction and randomness should be inserted to make the model less deterministic and to give creators some control over the sample generation. Our contribution is nonetheless the first step toward more realistic and valuable automatic music arrangement systems. Further significant steps could be made to reach the final goal of human-level automatic music arrangement production. Moreover, this task moves towards the direction of automatic music arrangement (the same methodology could be extended, in the future, to more complex domains, such as voice or guitar or the whole song). Already now software like Melodyne [66, 67] delivers producers a powerful user interface to directly modify and adjust a spectrogram-based representation of audio signals to correct, perfect, reshape and restructure vocals, samples, and recordings of all kinds. It is not unlikely that in the future, artists and composers will start creating their music almost like they were drawing.

Data availability The datasets generated by the survey research and analyzed during the current study are publicly available at the following addresses: https://freemusicarchive.org/home and https://sigsep.github.io/datasets/musdb.html.

Declarations

Competing interests The authors declare no competing interests.

A hyperparameters

We report here below the hyperparameters used for the proposed method. The window size referers to the size of the each individual mel-spectrogram. The patch size refers to the pixel dimensions of the patches considered for the Patch-GAN, that is used as the building block for the discriminator. λ regulates the trade off between the adversarial losses and the cycle loss, so that a higher value of λ will correspond to higher importance for the latter. Finally, β is used in the Adam optimization algorithm to regulate the strength of the momentum, that is, how much of the previous gradient to keep in the current iteration.

Hyperparameter	Value
Epochs	20
Window size	256
Patch size	70×70
Learning rate	2e−4
λ	10
β	(0.5, 0.999)

References

1. Briot J-P, Hadjeres G, Pachet F-D. Deep learning techniques for music generation. Cham: Springer; 2020.
2. Assayag G, Rueda C, Laurson M, Agon C, Delerue O. Computer-assisted composition at IRCAM: from PatchWork to OpenMusic. Comput Music J. 1999;23(3):59–72.
3. Papadopoulos A, Roy P, Pachet F. Assisted lead sheet composition using flowcomposer. In: International conference on principles and practice of constraint programming. Cham: Springer; 2016. p. 769–85.
4. Jiang N, Jin S, Duan Z, Zhang C. RI-duet: online music accompaniment generation using deep reinforcement learning. In: Proceedings of the AAAI conference on artificial intelligence, vol. 34; 2020. p. 710–8.
5. Ren Y, He J, Tan X, Qin T, Zhao Z, Liu T-Y. Popmag: pop music accompaniment generation. In: Proceedings of the 28th ACM international conference on multimedia; 2020. p. 1198–206.
6. Lee C, Shih J, Yu K, Lin H. Automatic music genre classification based on modulation spectral analysis of spectral and cepstral features. IEEE Trans Multimed. 2009;11(4):670–82. https://doi.org/10.1109/TMM.2009.2017635.
7. Zhu J-Y, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE international conference on computer vision; 2017. p. 2223–32.
8. Defferrard M, Mohanty SP, Carroll SF, Salathé M. Learning to recognize musical genre from audio. In: The 2018 web conference companion. Lyon: ACM Press; 2018. https://doi.org/10.1145/3184558.3192310. https://arxiv.org/abs/1803.05337.

9. Rafii Z, Liutkus A, Stöter F-R, Mimilakis SI, Bittner R. The MUSDB18 corpus for music separation. 2017. https://doi.org/10.5281/zenodo.1117372.

10. Défossez A, Usunier N, Bottou L, Bach F. Demucs: deep extractor for music sources with extra unlabeled data remixed. arXiv preprint. 2019. arXiv:1909.01174.

11. Isola P, Zhu J-Y, Zhou T, Efros AA. Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 1125–34.

12. Dhariwal P, Jun H, Payne C, Kim JW, Radford A, Sutskever I. Jukebox: a generative model for music. arXiv preprint. 2020. arXiv:2005.00341.

13. Yang L-C, Chou S-Y, Yang Y-H. MidiNet: a convolutional generative adversarial network for symbolic-domain music generation. arXiv preprint. 2017. arXiv:1703.10847.

14. Mogren O. C-RNN-GAN: continuous recurrent neural networks with adversarial training. arXiv preprint. 2016. arXiv:1611.09904.

15. Mangal S, Modak R, Joshi P. LSTM based music generation system. arXiv preprint. 2019. arXiv:1908.01080.

16. Jaques N, Gu S, Turner RE, Eck D. Generating music by fine-tuning recurrent neural networks with reinforcement learning. In: Deep reinforcement learning workshop, NIPS; 2016.

17. Makris D, Kaliakatsos-Papakostas M, Karydis I, Kermanidis KL. Combining LSTM and feed forward neural networks for conditional rhythm composition. In: International conference on engineering applications of neural networks. Cham: Springer; 2017. p. 570–82.

18. Yamshchikov IP, Tikhonov A. Music generation with variational recurrent autoencoder supported by history. SN Appl Sci. 2020;2(12):1–7.

19. Roberts A, Engel J, Raffel C, Hawthorne C, Eck D. A hierarchical latent vector model for learning long-term structure in music. In: International conference on machine learning. PMLR; 2018. p. 4364–73.

20. Lattner S, Grachten M. High-level control of drum track generation using learned patterns of rhythmic interaction. In: WASPAA 2019; 2019.

21. Boulanger-Lewandowski N, Bengio Y, Vincent P. Modeling temporal dependencies in high-dimensional sequences: application to polyphonic music generation and transcription. In: Proceedings of the 29th international conference on machine learning; 2012. p. 1881–8.

22. Hadjeres G, Nielsen F. Interactive music generation with positional constraints using anticipation-rnns. arXiv preprint. 2017. arXiv:1709.06404.

23. Johnson DD. Generating polyphonic music using tied parallel networks. In: International conference on evolutionary and biologically inspired music and art. Cham: Springer; 2017. p. 128–43.

24. Hadjeres G, Pachet F, Nielsen F. Deepbach: a steerable model for bach chorales generation. In: International conference on machine learning. PMLR; 2017. p. 1362–71.

25. Lattner S, Grachten M, Widmer G. Imposing higher-level structure in polyphonic music generation using convolutional restricted Boltzmann machines and constraints. J Creat Music Syst. 2018;2(2):1–31.

26. Mao HH, Shin T, Cottrell G. DeepJ: style-specific music generation. In: 2018 IEEE 12th international conference on semantic computing (ICSC). IEEE; 2018. p. 377–82.

27. Hawthorne C, Stasyuk A, Roberts A, Simon I, Huang C-ZA, Dieleman S, Elsen E, Engel J, Eck D. Enabling factorized piano music modeling and generation with the maestro dataset. In: International conference on learning representations; 2018.

28. Hawthorne C, Elsen E, Song J, Roberts A, Simon I, Raffel C, Engel J, Oore S, Eck D. Onsets and frames: dual-objective piano transcription. arXiv preprint. 2017. arXiv:1710.11153.

29. Huang C-ZA, Vaswani A, Uszkoreit J, Simon I, Hawthorne C, Shazeer N, Dai AM, Hoffman MD, Dinculescu M, Eck D. Music transformer: generating music with long-term structure. In: International conference on learning representations; 2018.

30. van den Oord A, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior A, Kavukcuoglu K. Wavenet: a generative model for raw audio. In: 9th ISCA speech synthesis workshop; 2016. p. 125.

31. Zhu H, Liu Q, Yuan NJ, Qin C, Li J, Zhang K, Zhou G, Wei F, Xu Y, Chen E. Xiaoice band: a melody and arrangement generation framework for pop music. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining; 2018. p. 2837–46.

32. Jin C, Tie Y, Bai Y, Lv X, Liu S. A style-specific music composition neural network. Neural Process Lett. 2020;52(3):1893–912.

33. Sánchez Fernández LP, Sánchez Pérez LA, Carbajal Hernández JJ, Rojo Ruiz A. Aircraft classification and acoustic impact estimation based on real-time take-off noise measurements. Neural Process Lett. 2013;38(2):239–59.

34. Khan NM, Khan GM. Real-time lossy audio signal reconstruction using novel sliding based multi-instance linear regression/random forest and enhanced cgpann. Neural Process Lett. 2021;53(1):227–55.

35. Prenger R, Valle R, Catanzaro B. Waveglow: a flow-based generative network for speech synthesis. In: ICASSP 2019–2019 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE; 2019. p. 3617–21.

36. Wang X, Takaki S, Yamagishi J. Neural source-filter waveform models for statistical parametric speech synthesis. IEEE/ACM Trans Audio Speech Lang Process. 2019;28:402–15.

37. Zhao Y, Wang X, Juvela L, Yamagishi J. Transferring neural speech waveform synthesizers to musical instrument sounds generation. In: ICASSP 2020–2020 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE; 2020. p. 6269–73.

38. Mehri S, Kumar K, Gulrajani I, Kumar R, Jain S, Sotelo J, Courville A, Bengio Y. SampleRNN: an unconditional end-to-end neural audio generation model. arXiv preprint. 2016. arXiv:1612.07837.

39. Bhave A, Sharma M, Janghel RR. Music generation using deep learning. In: Soft computing and signal processing. Springer; 2019. p. 203–11.

40. Manzelli R, Thakkar V, Siahkamari A, Kulis B. An end to end model for automatic music generation: combining deep raw and symbolic audio networks. In: Proceedings of the musical metacreation workshop at 9th international conference on computational creativity, Salamanca, Spain; 2018.

41. Wu Y-K, Chiu C-Y, Yang Y-H. JukeDrummer: conditional beat-aware audio-domain drum accompaniment generation via transformer VQ-VA. arXiv preprint. 2022. arXiv:2210.06007.

42. Vasquez S, Lewis M. Melnet: a generative model for audio in the frequency domain. arXiv preprint. 2019. arXiv:1906.01083.

43. Sigtia S, Benetos E, Dixon S. An end-to-end neural network for polyphonic piano music transcription. IEEE/ACM Trans Audio Speech Lang Process. 2016;24(5):927–39.

44. Dong Y, Yang X, Zhao X, Li J. Bidirectional convolutional recurrent sparse network (BCRSN): an efficient model for music emotion recognition. IEEE Trans Multimed. 2019;21(12):3150–63.

45. Brunner G, Wang Y, Wattenhofer R, Zhao S. Symbolic music genre transfer with cyclegan. In: 2018 IEEE 30th international conference on tools with artificial intelligence (ICTAI). IEEE; 2018. p. 786–93.

46. Kereliuk C, Sturm BL, Larsen J. Deep learning and music adversaries. IEEE Trans Multimed. 2015;17(11):2059–71. https://doi.org/10.1109/TMM.2015.2478068.

47. Nistal J, Lattner S, Richard G. Drumgan: synthesis of drum sounds with timbral feature conditioning using generative adversarial networks. In: ISMIR2020; 2020.

48. Dieleman S, van den Oord A, Simonyan K. The challenge of realistic music generation: modelling raw audio at scale. In: Advances in neural information processing systems. San Mateo: Morgan Kaufmann Publishers; 2018. p. 7989–99.

49. Yang L-C, Lerch A. On the evaluation of generative models in music. Neural Comput Appl. 2020;32(9):4773–84.

50. Luo Y, Mesgarani N. Conv-tasnet: surpassing ideal time-frequency magnitude masking for speech separation. IEEE/ACM Trans Audio Speech Lang Process. 2019;27:1256–66.

51. Müller M. Fundamentals of music processing: audio, analysis, algorithms, applications. Cham: Springer; 2015.

52. Stevens SS, Volkmann J, Newman EB. A scale for the measurement of the psychological magnitude pitch. J Acoust Soc Am. 1937;8(3):185–90.

53. Logan B, Robinson T. Adaptive model-based speech enhancement. Speech Commun. 2001;34(4):351–68.

54. Decorsière R, Søndergaard PL, MacDonald EN, Dau T. Inversion of auditory spectrograms, traditional spectrograms, and other envelope representations. IEEE/ACM Trans Audio Speech Lang Process. 2015;23(1):46–56. https://doi.org/10.1109/TASLP.2014.2367821.

55. Griffin D, Lim J. Signal estimation from modified short-time Fourier transform. IEEE Trans Acoust Speech Signal Process. 1984;32(2):236–43. https://doi.org/10.1109/TASSP.1984.1164317.

56. Johnson J, Alahi A, Fei-Fei L. Perceptual losses for real-time style transfer and super-resolution. In: European conference on computer vision. Cham: Springer; 2016. p. 694–711.

57. Ledig C, Theis L, Huszár F, Caballero J, Cunningham A, Acosta A, Aitken A, Tejani A, Totz J, Wang Z. Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017. p. 4681–90.

58. Li C, Wand M. Precomputed real-time texture synthesis with Markovian generative adversarial networks. In: European conference on computer vision. Cham: Springer; 2016. p. 702–16.

59. Kingma DP, Ba J. Adam: a method for stochastic optimization. In: Bengio Y, LeCun Y, editors. 3rd international conference on learning representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, conference track proceedings; 2015. http://arxiv.org/abs/1412.6980.

60. Lee J, Lee J. Music popularity: metrics, characteristics, and audio-based prediction. IEEE Trans Multimed. 2018;20(11):3173–82. https://doi.org/10.1109/TMM.2018.2820903.

61. Mor N, Wolf L, Polyak A, Taigman Y. A universal music translation network. arXiv preprint. 2018. arXiv:1805.07848.

62. Andersen AH, de Haan JM, Tan Z-H, Jensen J. A non-intrusive short-time objective intelligibility measure. In: 2017 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE; 2017. p. 5085–9.

63. Heusel M, Ramsauer H, Unterthiner T, Nessler B, Hochreiter S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in neural information processing systems. San Mateo: Morgan Kaufmann Publishers; 2017. p. 6626–37.

64. Chen L, Wu L, Hu Z, Wang M. Quality-aware unpaired image-to-image translation. IEEE Trans Multimed. 2019;21(10):2664–74. https://doi.org/10.1109/TMM.2019.2907052.

65. Emami H, Aliabadi MM, Dong M, Chinnam RB. Spa-gan: spatial attention gan for image-to-image translation. IEEE Trans Multimed. 2021;23:391–401. https://doi.org/10.1109/TMM.2020.2975961.

66. Neubäcker P. Sound-object oriented analysis and note-object oriented processing of polyphonic sound recordings. Google Patents. US Patent 8,022,286; 2011.

67. Senior M. Celemony melodyne DNA editor. Sound on sound; 2009.