AUDIO STYLE TRANSFER WITH RHYTHMIC CONSTRAINTS

Maciek Tomczak, Carl Southall and Jason Hockman

Digital Media Technology Lab (DMT Lab)
Birmingham City University
Birmingham, United Kingdom
`{maciek, carl, jason}@bcu.ac.uk`

ABSTRACT

In this transformation we present a rhythmically constrained audio style transfer technique for automatic mixing and mashing of two audio inputs. In this transformation the rhythmic and timbral features of both input signals are combined together through the use of an audio style transfer process that transforms the files so that they adhere to a larger metrical structure of the chosen input. This is accomplished by finding beat boundaries of both inputs and performing the transformation on beat-length audio segments. In order for the system to perform a mashup between two signals, we reformulate the previously used audio style transfer loss terms into three loss functions and enable them to be independent of the input. We measure and compare rhythmic similarities of the transformed and input audio signals using their rhythmic envelopes to investigate the influence of the tested transformation objectives.

1. INTRODUCTION

In the field of digital audio effects processing, creative transformations of musical audio refer to methods for automated manipulations of temporally-relevant sounds in time. These systems can be seen as part of a larger set of support systems to guide users when they lack inspiration, technical knowledge, musical capability as it relates to melody, harmony, rhythm, structure or style [1]. In recent years, the use of powerful machine learning algorithms, such as convolutional neural networks (CNN), have become an essential component in the development of such intelligent musical expert agents. A step in this direction has recently emerged as a research topic of audio style transfer.

1.1. Background

Audio style transfer (AST) methods use machine learning algorithms to modify the timbral characteristics of musical audio signals. AST was first attempted in [2, 3], which directly extended an algorithm proposed for images in [4]. In AST, a new output is synthesised by minimising the *content* loss with respect to the *content*-contributing audio input and the *style* loss with respect to one or more audio examples of a given *style*. The *content* loss is based on comparing the network activations of features derived from an audio spectrogram. The *style* loss matches the statistics of the Gram matrix (i.e., inner product between neural feature maps) activations in the higher levels of the network. In [5], the authors argue that *content* may refer to the underlying structure of the input music (e.g., note pitches, rhythm) and *style* can refer to timbres of instruments or genres.

Definitions and challenges of style transfer for music are presented in [6]. The appropriateness of the Gram matrix as a representation for *style* remains unclear for both music and images. This challenge is furthered by the ambiguous meaning of the term *style*, which is related to nearly all aspects of music. It has been suggested that the Gram matrix corresponds to a representation of musical timbre [5, 7]. To test the possibilities of creating rhythmically focused transformations varied according to different loss formulations we explore the use of the Gram matrix further and report on the suitability and shortcomings of this approach.

Approaches to AST can be divided into two categories: (1) time-frequency domain (i.e., spectrogram) based, where log-magnitudes of a short-time Fourier transform (STFT) are used as inputs to a CNN that performs the *style* transformation followed by a process of phase reconstruction; and (2) time-domain (i.e., raw audio) based, where the audio samples are directly optimised, removing the need for additional phase reconstruction.

The majority of AST research performs timbral transformation in the time-frequency domain, while preserving the rhythmic characteristics of the *content* recording. Grinstein et al. [5] introduced a spectral filtering method based on a sound texture model to improve the transformation of timbre from *style* directly onto a new audio initialised as *content* sound. The authors experimented with different pre-trained neural networks to aid their transformation. Similarly, Wyse [8] explored the effects of pre-trained weights from a network trained on an audio classification dataset for AST. The presented system appears to generate a more integrated transformation of *content* and *style* with the included pre-trained network. In [7] the authors provide an additional loss term that constrains the temporal envelope of the newly generated spectrogram to match that of the *style* recording. The motivation for the additional loss function was to better portray the temporal dynamics of the *style* recording and diminish the impact of the *content* recording. Audio style transfer was also used in the attempt to change the *style* of prosodic speech by [9]. The authors report success in transferring low-level textural features of the *content* but difficulty in transferring the high-level prosody such as emotion or accent of the *style* voice recording.

In addition to the above spectrogram based methods, AST systems have been proposed that can change rhythmic patterns of the input by applying the transformation directly on the raw audio. Mital [10] combines information from multiple discrete Fourier transform parts and presents them as different concatenated batches (layers) of a convolutional filter. Concatenated real, imaginary, and magnitude features are presented as producing the best results. Barry and Kim [11] implemented a parallel architecture that adds deep specialised networks with reduced frequency channels projected onto constant-Q transform basis, for key invariance capabilities, and Mel basis for representing longer rhythmic patterns. Their approach allows for longer temporal memory over the input features.

While the above methods are capable of timbral transforma-

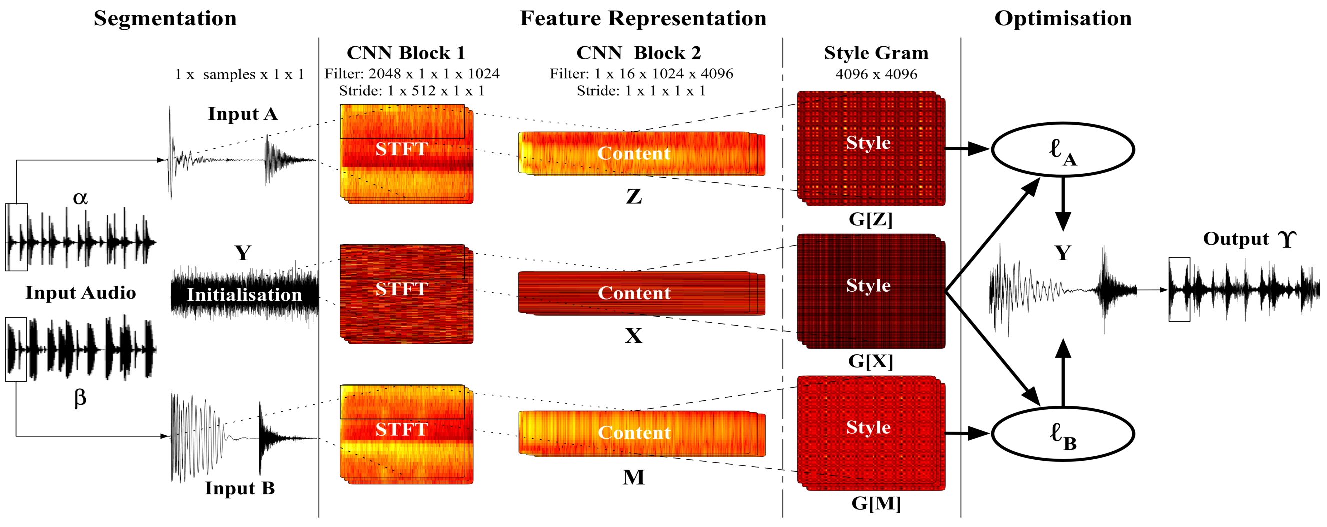

Figure 1: *Audio style transfer with rhythmic constraints in three stages: Segmentation, Feature Representation and Optimisation. A noise signal Υ is iteratively transformed to represent the timbral and rhythmic characteristics of a user-defined mix between two input audio recordings A and B. Solid lines divide the three stages; dotted lines represent convolution, dashed lines represent style Gram computation and the vertical solid-dashed line represents a scaled exponential linear unit (SeLU) activation layer.*

tions, these modifications are not temporally restrictive and therefore do not constrain the elements in a metrically relevant manner. Alternatively, there have been several signal processing approaches to rhythmic transformations, including: percussive swing modification in polyphonic audio recordings [12]; rhythmic pattern manipulation of a drum loop to match that of another [13]; the rhythmic modification of an input polyphonic recording given the intra-measure structure of a model recording [14] and multi-song music mashup creation [15].

1.2. Motivation

In this paper we propose a system that extends the AST method to preserve the meter and the rhythmic structure of the chosen musical signal, while maintaining stylistic elements of both inputs. Our aim in the following is to transform two recordings such that their timbral and rhythmic patterns are merged together, with the presence of each being user-defined. To do this, we alter the original AST formulation to optimise the *style* representations of the input recordings simultaneously. To improve the creative application of this approach we constrain the transformation to act only on beat-length segments and test it on a small corpus of drum performances. This approach ensures that the transformation adheres to a larger rhythmic structure of the recordings with opportunities to generate new music that is both creative and realistic, as well as to uncover musical relationships of familiar audio samples that might otherwise have never been conceptualised.

The remainder of this paper is structured as follows: Section 2 presents our proposed method for AST with rhythmic constraints. Section 3 presents experiments undertaken and the results with discussion. We conclude with suggestions for future work in Section 4.

2. METHOD

Figure 1 presents an overview of our proposed system for AST. The system extends work by [11],[1] in which a noise signal Y is iteratively transformed to embody the timbral characteristics of a target associated with two audio recordings (α and β). In [11], the *content* refers to a network projection of input audio and *style* refers to a statistical representation of the feature map generated from previous layers of the network (as discussed in Section 2.3.1). We add to this kind of transformation through the integration of rhythmic constraints and with the addition of interchangeable loss terms with regards to both inputs.

The proposed model consists of three stages: (1) *segmentation*, where the two audio files (α and β) are divided into beat-length segments (A and B respectively); (2) *feature representation*, in which feature representations (Z, M and X) of A, B and Y are created using a CNN; and (3) *optimisation* in which Y is iteratively transformed to simultaneously match loss functions related to the feature representations of A and B. The resultant transformation Υ is a concatenation of the transformed beat-length segments Y.

2.1. Segmentation

Our motivation for the inclusion of segmentation in AST is to divide the inputs so that they adhere to a larger metrical structure during the transformation, while reducing the computation cost. In our experience, musically-interesting and rhythmically-stable transformations may be obtained when assessing beat-length audio segments. In order for input audio files to be processed by the proposed system, beat and downbeat positions must be first extracted. We compute segment boundaries using a state-of-the-art beat and downbeat tracking algorithm [16] included in the madmom Python

[1] `https://github.com/anonymousiclr2018/Style-Transfer-for-Musical-Audio`

library.[2] We then use the detected beat positions, starting from the first downbeat, as segment boundaries for A and B and generate the new noise segment Y using the same length.

2.2. Feature Representation

The aim of the feature representation stage is to project the input audio segments onto neural feature maps, which results in the creation of *content* and *style* matrices.

2.2.1. Content

To create the *content* matrices, the same two-stage process is performed in separate network branches for A, B and Y, where the weights of signal Y are initialised with random noise and matrices A and B contain input audio data as in [11]. First, feature maps are created by projecting the audio onto STFT bases. Then, the feature maps are projected further onto a larger number of channels as in [2, 5, 10, 11] to create the *content* representation.

The input audio (A, B and Y) is segmented into T frames using a Hanning window of n samples (n = 2048) with a $\frac{n}{4}$ hop-size. A frequency projection of each of the frames is then created with a single CNN layer that uses filters initialised with real and imaginary parts of the discrete Fourier transform resulting in a Tx$\frac{n}{2}$ spectrogram. We convert the created spectrogram to a log-magnitude representation. This transformation is represented in CNN Block 1 in Figure 1, where the filter size is nx1x1x$\frac{n}{2}$ with strides of 1x$\frac{n}{4}$x1x1.

CNN Block 2 (Figure 1) depicts neural feature computation from the STFT projections that becomes the *content* and can be understood as the low-level features of the input. The CNN architecture consists of a single convolutional layer with a filter size of 1xHxFxQ, where H is the number of time frames convolved with the filter, F is the number of frequency bins and Q represents the number of frequency channels that the input spectrogram will be projected onto. The filter size used in this implementation is 1x16x$\frac{n}{2}$x2n. We use a temporal receptive field (i.e., a contextual window modeled by each hidden state of the network) of 16 frames (~370ms) to capture acoustic information about instruments from a context longer than half beat length at 120 beats per minute (BPM). Each network is followed by a scaled exponential linear unit (SeLU) [17] activation layer, represented as vertical solid-dashed line in Figure 1, in place of standard rectified linear units (ReLU), as in [11]. This is done to increase the quality of the synthesised audio and reduce convergence time of the optimisation algorithm. For the rest of the paper, the *content* matrices for A, B and Y are termed Z, M and X respectively.

2.2.2. Style

Style can be understood as high-level information of the input neural features. To obtain a representation of the *style* of an input spectrogram, a Gram matrix G is used as in [4]. This feature space is designed to capture texture or intra-feature map statistics. For each *content* matrix (Z, M and X) G is calculated using the inner product:

$$G[X]_{ij} = \sum_k X_{ik} X_{jk}. \tag{1}$$

[2]`https://github.com/CPJKU/madmom`

2.3. Optimisation

2.3.1. Content and Style Loss Functions

In order to control the contributions of *content* and *style* from the two inputs, the total loss $\mathcal{L}$ is expressed as a sum of *content* ℓ_C and *style* ℓ_S loss functions for the input audio files A and B:

$$\mathcal{L} = \sigma \ell_C^A + \delta \ell_C^B + \theta \ell_S^A + \phi \ell_S^B, \tag{2}$$

where σ, δ, θ and ϕ are proportion parameters that add up to 1 and help configure loss preferences between the input recordings. The individual ℓ terms can be added and changed according to the transformation objective. The *content* loss ℓ_C is a squared error loss between the frame indices i and channels j of the transform *content* matrix X and the input audio *content* matrices (Z or M):

$$\ell_C^A = \frac{1}{2} \sum_{i,j} (X_{ij} - Z_{ij})^2, \tag{3}$$

$$\ell_C^B = \frac{1}{2} \sum_{i,j} (X_{ij} - M_{ij})^2. \tag{4}$$

The *style* loss ℓ_S is the sum of the squared difference between the transformed Gram matrix $G[X]$ and the input Gram matrices ($G[Z]$ or $G[M]$):

$$\ell_S^A = \frac{1}{Q^2} \sum_{i,j} (G[X]_{ij} - G[Z]_{ij})^2, \tag{5}$$

$$\ell_S^B = \frac{1}{Q^2} \sum_{i,j} (G[X]_{ij} - G[M]_{ij})^2. \tag{6}$$

The motivation for using the *style* loss as formulated above was to preserve the statistics about the convolutional representation over the entire input, while losing local information about where exactly different elements are.

2.3.2. Training

We use different combinations of *style* and *content* loss functions to shape the output of the transformation (Section 3.3). Following [11], we normalise the magnitudes of the gradients of loss terms to 1 to moderate the imbalances in weighting of either function. We use the limited-memory BFGS [18] gradient descent-based optimisation algorithm for its appropriateness in non-linear problems related to neural style transfer [4, 19]. Once initialised, the feature map representations of *content* and *style* from inputs A and B do not change throughout the training stage. In each gradient step the *content* and *style* activations are back-propagated all the way to the network output Y. Hence, only weights originating from Y are being manipulated during the optimisation process, while all feature representations remain unchanged for inputs A and B. The optimisation of the concerned weights is stopped after 500 iterations. An NVIDIA Tesla M40 computing processor was used for this project with an average of 3 seconds per algorithm iteration.

2.4. Implementation

Our system is implemented using the Tensorflow Python library.[3] The processing branches of A, B and Y are part of the same CNN in one Tensorflow computation graph. This means that the neural representations of the input time-domain audio Y can be optimised simultaneously in one stage.

[3]`https://www.tensorflow.org/`

Table 1: *Mean cosine similarities from 15 transformed target audio pairs. The cosine similarities are calculated between rhythmic envelopes extracted from full α, β and Υ audio files for loss functions $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$. Mean cosine similarity calculated from all α and β rhythmic envelopes in the experiment is 0.58.*

	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$
Υ to input α	0.32	0.60	0.43
Υ to input β	**0.37**	0.60	**0.52**

3. EXPERIMENTS

3.1. Experimental Setup

We test the rhythmic modification characteristic of our AST approach by assessing the rhythmic similarity of the transformed output to the input audio for three loss term combinations. To achieve this comparison, we generate rhythmic envelopes from the newly created audio files Υ and compare them to those of α and β.

3.2. Dataset

For this experiment we created 30 drum loops (mono .wav sampled at 22.05 kHz with 16-bit resolution) of 4 measures in length, which differ in rhythmic patterns consisting of various kick and snare drums. All transformation examples are created from 15 pairs of input drum loops to reduce computation cost. All drum loops have a fixed-tempo set to 120 BPM in $\frac{4}{4}$ meter. Our motivation for using a fixed-tempo of 120 BPM was to test how our transformation performs on already beat-synchronised inputs essential in the processes of mixing and mashing audio recordings together. The chosen tempo is typical for many genres in popular music as well, as it is the default tempo in various digital audio workstations used in music production. The drum loops used in our tests were generated with twelve different pattern styles defined by the Logic X Drummer virtual instrument.[4]

3.3. Rhythmic Similarity

To test the rhythmic constraints imposed by different transformation objectives within the AST technique, we compare the rhythmic similarity [15] of pairs of transformations. The rhythmic envelopes are calculated from the spectral difference function [20] of new audio Υ with inputs α or β. We calculate the rhythmic envelopes as the sum over frequency bins from the first-order difference between each adjacent magnitude spectra. The STFT parameters from Section 2.2 are used. The resulting rhythmic envelopes were normalised to range from 0 to 1. To determine the rhythmic similarity D between every pair of rhythmic envelopes R we calculate the cosine similarity as:

$$D_{\omega,\Upsilon} = \frac{R_\omega \cdot R_\Upsilon}{\|R_\omega\|\|R_\Upsilon\|}, \tag{7}$$

where ω can represent envelope of either α or β. Thus, the rhythmic similarity will be close to unity for very similar patterns and nearer to zero for dissimilar patterns. The mean of all D values is calculated across 15 transformation audio pairs per loss term formulation.

[4]`https://support.apple.com/kb/PH13070`

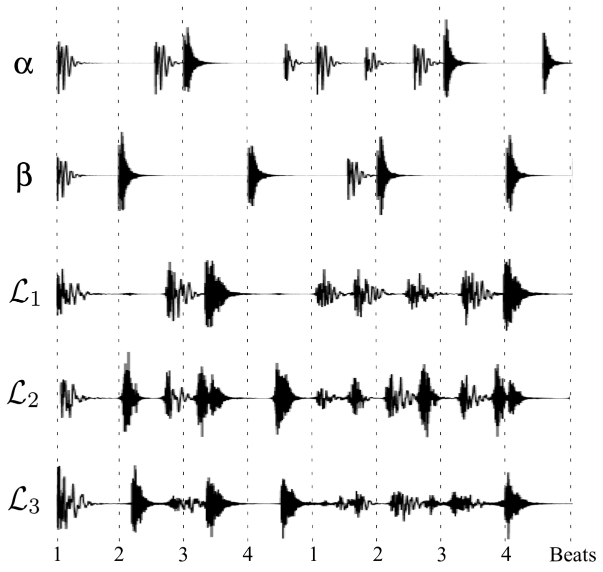

Figure 2: *Example transformations generated from three loss terms $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ from input audio signals α and β.*

We test our approach with three objectives associated with combinations of loss terms with all proportion parameters σ, δ, θ and ϕ set to be equal:

Objective $\mathcal{L}_1$: $\ell_S^A + \ell_C^B$. In this objective we test the ability of our system to move acoustic events to create a rhythmically new performance that is more similar to β through the low-level information from the *content* loss.

Objective $\mathcal{L}_2$: $\ell_S^A + \ell_S^B$. In this objective we test a transformation that solely uses the *style* feature representations to mix high-level characteristics of both recordings. This transformation is akin to a mashup of both audio inputs.

Objective $\mathcal{L}_3$: $\ell_S^A + \ell_S^B + \ell_C^B$. This objective reinforces the mashup transformation with more low-level information from β.

3.4. Results and Discussion

The overall similarity results are summarised in Table 1. The cosine similarities of the transformations Υ compared with input β are higher for objectives $\mathcal{L}_1$ (0.37) and $\mathcal{L}_3$ (0.52), where the B *content* loss (ℓ_C^B) was used. When the B *content* loss was not used ($\mathcal{L}_2$) the transformation similarities to α and β are both 0.60. We believe this is due to both *style* losses having the same weighting, resulting in an equal mix of both inputs that creates a kind of rhythmic and timbral mashup. This is in agreement with the mean similarities of the α and β together (0.58). In addition, when larger proportions of the *content* loss are used the transformations are expected to be more similar to the corresponding *content* loss of the chosen input.

Figure 2 shows transformed waveforms of inputs α and β using the three different loss term combinations. In $\mathcal{L}_1$ the rhythmic pattern of α is recreated at different metrical positions that match the beat pattern of β (e.g., on beat 4 of the second measure). On beat 2 of the first measure the event from β does not appear in the resulting transformation, while in objectives $\mathcal{L}_2$ and $\mathcal{L}_3$ the event is included. Similarly, the transformation in beat 4 of the first measure from $\mathcal{L}_1$ removes an event that is present in objectives $\mathcal{L}_2$ and $\mathcal{L}_3$ as an instrument from β. In this case a kick from α was transformed into a snare from β. The difference between the $\mathcal{L}_2$ and $\mathcal{L}_3$ transformations show the effect of the added *content* loss from B in that drum events that correspond to silences become attenuated in the resulting mashup of both recordings (e.g., beat 3 of the first measure).

Experimental transformations along with other examples are presented using the web-based audio player by [21] and can be found on the supporting website for this project.[5] The resultant audio examples acquired from loss terms $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$ are accompanied by transformation outputs from publicly available algorithms [11, 10, 2]. Our rhythmically-constrained transformation differs in that it is capable of generating new rhythmic patterns from both inputs while preserving the beat pattern of the chosen recording. Challenges faced by all AST transformations are the loss of phase information and the addition of noise, potentially due to the high-level representation of the *style* loss (i.e., Gram matrix).

As in other AST methods to date, we have used the Gram matrix as a representation for *style*, yet it remains questionable whether this feature representation is suitable for transformations based on high-level musical information. Briot and Pachet suggest that this technique presents challenges for audio due to anisotropy of the *content* representation [22]. Anisotropy signifies dependence on directions and here it refers to the nature of the audio spectrogram. In this time-frequency representation the dimensions do not correlate together in the same way a pixel would in an image. A pixel almost always corresponds to one object whereas in music multiple sources overlap causing inherent issues when using the Gram matrix to transform local changes in timbres.

3.5. Attempted Rhythmic Loss Terms

In addition to segmenting the audio and experimenting with different combinations of the existing loss terms, we also tested two new loss terms which aimed to aid the rhythmic aspects of AST. Both terms were formulated to minimise the cosine distance between rhythmic envelopes of the chosen input and the transformation. In the first loss term, each rhythmic envelope was calculated as the sum over frequency bins of the two spectrograms (i.e., feature representations from CNN Block 1 in Figure 1). In the second term, we created a new network branch for the chosen input where the resulting STFT projection was filtered with the first-order difference between each adjacent log-magnitude spectra to then create a detection function focused more on percussive events. The second loss term was formulated to minimise the cosine distance between rhythmic envelopes of the filtered input spectrogram and the transformation. Through informal listening we found that neither term improved the transformation in conjunction with $\mathcal{L}_1$ and $\mathcal{L}_2$ loss terms. The first loss term was causing generated drum events to lose their transient information, whereas the second term removed events created in the silent sections of the rhythmic envelope, while increasing amplitudes of drum event transients.

[5]`https://maciek-tomczak.github.io/maciek.github.io/Audio-Style-Transfer-with-Rhythmic-Constraints`

3.6. Additional Audio Inputs

In our rhythmic extension of AST we are able to create transformations using an arbitrary number of input recordings. In a music composition scenario, once the desired individual recordings are found, it is possible to create their combined transformation. One such purpose would be to mix multiple individual drum recordings together such as hi-hats, kicks and snares with the aim of creating their new rhythmic and timbral interpretation. However, with additional audio input signals the transformation becomes more difficult to control.

4. CONCLUSIONS AND FUTURE WORK

In this work we present a rhythmically constrained audio style transfer technique that explores different loss formulations. Our method utilises a time-domain approach to AST that acts on beat length segments of the input music signals. By constraining the transformation to shorter analysis segments that follow the metrical structure of the chosen input recording, we show that the resulting transformations sound rhythmically coherent, while reducing the computation cost. In the transformation the two input files are mixed together and allow the user to adjust the parameters of each loss term to experiment with the desired objective. The resulting transformation can be formulated as to replicate the exact spectral information of the input or to create a mashup.

Our attempt to measure the transformation similarities compared to their corresponding inputs shows differences in their rhythmic envelopes. From informal listening it can be heard that the beat detection does not need to be accurate for the transformation to produce rhythmically valid examples, however both inputs should have at least some rhythmic agreement when the *content* loss is used. In the case of the loss formulation that uses only the information about *content* and *style* of the inputs, the transformations are more different from both input files.

In future work, we intend to explore transformation objectives related to additional instrumentation and time scales, as well as, improving the phase reconstruction inherent in this kind of sound transformation.

5. REFERENCES

[1] Peter Knees, Kristina Andersen, Sergi Jordà, Michael Hlatky, Günter Geiger, Wulf Gaebele, and Roman Kaurson, "Giantsteps-progress towards developing intelligent and collaborative interfaces for music production and performance," in *Proceedings of the International Conference on Multimedia & Expo Workshops*. IEEE, pp. 1–4, 2015.

[2] Dmitry Ulyanov and Vadim Lebedev, "Audio texture synthesis and style transfer," 2016, Available at: https://tinyurl.com/ybgnsf9h.

[3] Davis Foote, Daylen Yang, and Mostafa Rohaninejad, "Do androids dream of electric beats?," 2016, Available at: https://tinyurl.com/yb5ww2tw.

[4] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge, "A neural algorithm of artistic style," *Computing Research Repository*, vol. abs/1508.06576, 2015.

[5] Eric Grinstein, Ngoc Q. K. Duong, Alexey Ozerov, and Patrick Pérez, "Audio style transfer," *Computing Research Repository*, vol. abs/1710.11385, 2017.

[6] Shuqi Dai, Zheng Zhang, and Gus G. Xia, "Music style transfer issues: A position paper," *arXiv preprint:1803.06841*, 2018.

[7] Prateek Verma and Julius O. Smith, "Neural style transfer for audio spectrograms," *Computing Research Repository*, vol. abs/1801.01589, 2018.

[8] Lonce Wyse, "Audio spectrogram representations for processing with convolutional neural networks," *Computing Research Repository*, vol. abs/1706.09559, 2017.

[9] Anthony Perez, Chris Proctor, and Archa Jain, "Style transfer for prosodic speech," Technical Report, Stanford University, 2017.

[10] Parag K. Mital, "Time domain neural audio style transfer," *Computing Research Repository*, vol. abs/1711.11160, 2017.

[11] Shaun Barry and Youngmoo Kim, "Style transfer for musical audio using multiple time-frequency representations," Unpublished article available at: https://tinyurl.com/y7nu7r9s, 2018.

[12] Fabien Gouyon, Lars Fabig, and Jordi Bonada, "Rhythmic expressiveness transformations of audio recordings: swing modifications," in *Proceedings of the Digital Audio Effects Workshop*, pp. 8–11, 2003.

[13] Emmanuel Ravelli, Juan P. Bello, and Mark Sandler, "Automatic rhythm modification of drum loops," *Signal Processing Letters, IEEE*, vol. 14, no. 4, pp. 228–231, 2007.

[14] Jason A. Hockman, Juan P. Bello, Matthew E. P. Davies, and Mark D. Plumbley, "Automated rhythmic transformation of musical audio," in *Proceedings of the International Conference on Digital Audio Effects*, pp. 177–180, 2008.

[15] Matthew E. P. Davies, Philippe Hamel, Kazutomo Yoshii, and Masataka Goto, "Automashupper: automatic creation of multi-song music mashups," *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol. 22, no. 12, pp. 1726–1737, 2014.

[16] Sebastian Böck, Florian Krebs, and Gerhard Widmer, "Joint beat and downbeat tracking with recurrent neural networks," in *Proceedings of the International Society for Music Information Retrieval*, pp. 255–261, 2016.

[17] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter, "Self-normalizing neural networks," in *Proceedings of the Conference on Neural Information Processing Systems*, pp. 972–981, 2017.

[18] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software*, vol. 23, no. 4, pp. 550–560, 1997.

[19] Kun He, Yan Wang, and John Hopcroft, "A powerful generative model using random weights for the deep image representation," in *Proceedings of the Conference on Neural Information Processing Systems*, pp. 631–639, 2016.

[20] Simon Dixon, "Onset detection revisited," in *Proceedings of the International Conference on Digital Audio Effects*, pp. 133–137, 2006.

[21] Nils Werner, Stefan Balke, Fabian-Robert Stöter, Meinard Müller, and Bernd Edler, "Trackswitch.js: A versatile web-based audio player for presenting scientifc results," in *Proceedings of the Web Audio Conference*, 2017.

[22] Jean-Pierre Briot and François Pachet, "Music generation by deep learning-challenges and directions," *Computing Research Repository*, vol. abs/1712.04371, 2017.