Preprint, Accept for publication, August 2020

Groove2Groove: One-Shot Music Style Transfer with Supervision from Synthetic Data

Ondřej Cífka, *Graduate Student Member, IEEE,* Umut Şimşekli, *Member, IEEE,* Gaël Richard, *Fellow, IEEE*

***Abstract*—Style transfer is the process of changing the style of an image, video, audio clip or musical piece so as to match the style of a given example. Even though the task has interesting practical applications within the music industry, it has so far received little attention from the audio and music processing community. In this paper, we present Groove2Groove, a one-shot style transfer method for symbolic music, focusing on the case of accompaniment styles in popular music and jazz. We propose an encoder-decoder neural network for the task, along with a synthetic data generation scheme to supply it with parallel training examples. This synthetic parallel data allows us to tackle the style transfer problem using end-to-end supervised learning, employing powerful techniques used in natural language processing. We experimentally demonstrate the performance of the model on style transfer using existing and newly proposed metrics, and also explore the possibility of style interpolation.**

***Index Terms*—Style transfer, symbolic music, synthetic data, deep learning, recurrent neural networks**

I. Introduction

STARTING with the work of Gatys et al. [1] on neural style transfer for images, style manipulation has become a very popular research topic and has attracted numerous attempts to extend it to other modalities, namely video [2], speech [3], music [4] and text [5]. In its original definition, a style transfer algorithm expects two inputs – a 'content image' and a 'style image' – and produces an image depicting the same content as the first input, but bearing the artistic style of the second one. In other words, the goal is to *transfer the style* from one image to another. A successful extension of this concept to music would offer musicians and music producers a new tool for repurposing existing material in a creative way.

More recently, especially in the context of language and music processing, the term 'style transfer' came to be used more generally to refer to any form of style transformation. In particular, this includes a task which might be more aptly described as *style conversion* or *translation* (as in [4], [6]–[10]), where the goal is to convert an input to a target style known in advance (or one of several such styles). The fundamental difference is that the target style is no longer transferred from one or a handful of examples, but usually learned from a large database. Such methods therefore typically suffer from a lack of generalization to new target styles.

In the present work, we adhere to the traditional definition of style transfer – dubbing it 'one-shot[1] style transfer' for clarity – and apply it within the domain of symbolic music (i.e. music represented as a set of events rather than a waveform). Our goal is then to transform a song X so as to match the style of a given song fragment Z, where both X and Z are provided as MIDI.[2] We therefore understand style not as a category (e.g. hip hop, jazz fusion, death metal), but rather as a set of characteristics unique to a particular artist or even an individual song (e.g. Africa by Toto).

Musical style is a broad term that may refer to any of the numerous aspects of music, e.g. composition, arrangement, performance or timbre [13]. This leads to the need for a clear definition of style (i.e. what is transferred from X) and content (what is retained from Z) in the style transfer task. In this work, we are interested in *accompaniment styles* in the context of jazz and popular music. This type of music is often notated using *chord charts*, providing the basic harmonic and rhythmic information for a song (an example can be found at the top of Fig. 2). When performing the song, the rhythm section of the band may use the chart as a basis for an improvised accompaniment ('comping'), choosing a bass line, chord voicings, slight harmonic alterations, rhythmic patterns, ornamentation and even instrumentation (e.g. piano vs. keyboard) as appropriate for the style. Herein, we consider the chord chart to be the *content*, and *style* then characterizes the process of converting this chord chart to an accompaniment. The underlying assumption is that the chord chart is independent of style; while this is not always realistic (for instance, jazz songs typically use more complex harmonies than some popular music genres), it allows us to develop a principled approach.

Our task can then be more precisely formulated as follows: *Generate a new accompaniment for X in the style of Z*. Note that even though we expect the output to follow the same chord chart as X, we do not assume this chart to be available. Also, we assume Z to be a *song fragment* approximately 8 measures long. While such a fragment might not fully capture the style of the entire song, it should manifest enough of its key features to allow for meaningful extrapolation. Indeed,

This work was supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 765068 (MIP-Frontiers).

All three authors are with the Information Processing and Communication Laboratory (LTCI), Télécom Paris, Institut Polytechnique de Paris, 19 place Marguerite Perey, 91120 Palaiseau, France. E-mail: {ondrej.cifka, umut.simsekli, gael.richard}@telecom-paris.fr.

Manuscript received March 5, 2020; revised June 15, 2020, and July 28, 2020.

[1] *One-shot learning* is the process of learning the concept of a class from a single example, in order to either perform classification [11], or to generate new samples from the class [12]. Style transfer can then be viewed as a one-shot learning problem, where we aim to 'learn' the style from the style input and transfer it to the content input.

[2] Musical Instrument Digital Interface; https://www.midi.org/

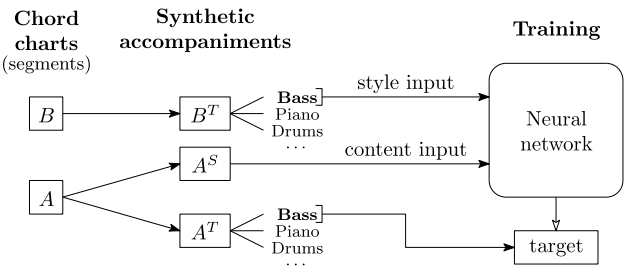


Fig. 1. The core idea of our approach. Starting from chord charts (here A, B), we create synthetic accompaniments in different styles (here S, T) and use them as training data for a neural network. The network is given the content input A^S (i.e. accompaniment for A in style S) and a single track of the style input B^T, and is trained to generate the corresponding track of the target accompaniment A^T.

employing a previously heard accompaniment pattern in a creative, improvised way is a skill possessed by many human musicians, and one that we aim to mimic here.

Our work represents one of the first attempts at one-shot style transfer for symbolic music accompaniments. We propose Groove2Groove[3] – or Grv2Grv for short – a novel end-to-end approach based on a neural network trained to map the input fragments X and Z directly to the output. The approach is summarized in Fig. 1.

Our major contributions are as follows:

- Unlike previous approaches to music style transfer, our method relies exclusively on *synthetic parallel training data*, which allows us to take advantage of the power of supervised learning. Moreover, our new synthetic dataset is unique in that it includes an unusually large number – thousands – of cleanly labeled styles, which is essential for generalization *across* styles. The dataset is available from the project's website.[4]
- We propose a novel *beat-relative* sequential encoding for our model's outputs, designed to be more robust in terms of timing than previously used sequential representations.
- We carefully evaluate the proposed method in the one-shot setting, showing that Grv2Grv is able to produce accompaniments even in styles unknown ahead of time.
- We introduce a new set of objective evaluation metrics to be used alongside existing metrics.
- We explore the possibility of controlling the output by directly manipulating the learned style representation, and demonstrate that it can be used for style interpolation.
- We are releasing the source code of our system.[5]

We also invite the reader to take a look at the live demo and other resources available at the supplementary website.[4]

The rest of the paper is structured as follows. Related work is briefly discussed in Section II; Section III describes the synthetic data generation scheme which underpins our approach; we present our proposed model and evaluation methods in Sections IV and V, respectively; finally, experimental results are given in Section VI.

[3]The term *groove* is difficult to define, but often refers to the characteristic rhythmic 'feel' of a piece, arising from the patterns employed by the rhythm section. Hence, it encompasses many of the key style features which we are considering here.

[4]https://groove2groove.telecom-paris.fr/

[5]https://github.com/cifkao/groove2groove

II. Related Work

A. Music style transformations

Music style transformations can take many different forms depending on the definition of style. Moreover, they may differ in whether and how they are conditioned on style.

The present work is most closely related to what we refer to as *style conversion* or *translation*, where a piece of music is converted to a given target style. This task is the focus of several recent works,[6] covering melodies [9], instrumentation [14], accompaniment [10], [15], [16], general arrangement style [17]–[19], expressive performances [8] as well as timbre [4], [20], [21]. These methods generally only allow for conversion to a limited set of target styles (often a single one); [14] should in principle support one-shot instrumentation style transfer, yet unlike the present work, it is not validated in this setting.

Another type of style transformation is the *harmonization* of a given melody, or more generally, what could be referred to as *arrangement completion* or *inpainting*, where a new track is generated to complement a set of tracks given as input. This sort of transformation has the advantage that it can be learned from an unlabeled music corpus. For example, the models proposed by [22], [23] enable harmonizing a given melody in the style of Bach chorales. More recent works achieve a form of one-shot style transfer by conditioning on an example of the target style, e.g. to harmonize a melody in a given piano performance style [24] or to generate a new kick drum track for a given song based on patterns extracted from a different recording [25]. While the focus of our work is also on generating accompaniments, a key difference is that the input to our Grv2Grv model is a full accompaniment rather than a partial one, and the output does not retain any of the original accompaniment tracks.

Finally, one-shot style transfer methods have been proposed for audio, but these focus mostly on low-level acoustic features ('sound textures'). Grinstein et al. [26] adapt the classic image style transfer method of Gatys et al. [1] for audio, but without specific focus on music. Approaches proposed for music so far [27], [28] employ more traditional signal processing techniques, combining signal decomposition with *musaicing* [29] (concatenative synthesis).

B. Related supervised methods

One-shot style transfer is seldom addressed using end-to-end supervised learning as in the present work, presumably due to the lack of suitable parallel training data. To our knowledge, the only existing example of such an approach is the work of Zhang et al. [30] on Chinese typeface transfer, exploiting the large number of Chinese characters combined with a relatively wide range of available typefaces. In our case, this approach is made possible by our synthetic data generation scheme,

[6]Even though these works often use the term 'style transfer', we avoid doing so for the reason stated in the introduction.

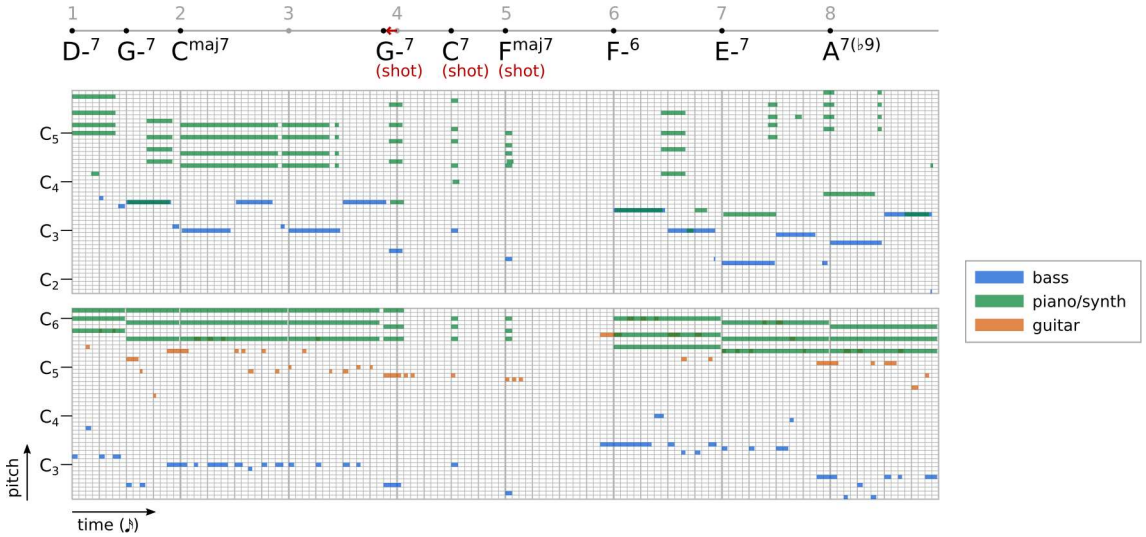


Fig. 2. 8-bar excerpts of BIAB-generated accompaniments in two different styles (top: Jazz Swing, bottom: Progressive Rock) visualized as piano rolls (non-quantized). Both correspond to the chord chart displayed above them. Drums are not included. A 16th note grid is shown for reference.

allowing to obtain an unlimited number of examples in a large number of styles.

The present work builds and extends upon our previous (non-one-shot) style translation work [10], which relied on synthetic data in a similar manner. The crucial difference between the two works lies in the fact that the former was limited to *translation* to a fixed set of target styles (only those included in the synthetic training dataset). In contrast, the present work adds the possibility to condition the model on unseen (*non-synthetic*) target styles via short examples, and introduces a data generation scheme necessary for training and evaluating a model with such conditioning.

Other, minor limitations of [10] were that:

1) it only considered accompaniments consisting of bass and piano and required training a separate model for each;
2) it ignored velocity information (dynamics).

Grv2Grv removes these limitations 1) by introducing the capability to produce any given combination of tracks – including drums – with a single model, and 2) by modeling velocity.

III. Data Generation

In order to apply supervised learning to the style transfer task, we need a parallel dataset consisting of triplets of musical fragments (A^S, B^T, A^T) where A^S denotes the *content input* in some source style S, B^T is the *style input*, serving as an example of the target style T, and A^T is the *target*, combining the content A with the target style T. We obtained such a dataset by generating accompaniments using RealBand from the Band-in-a-Box (BIAB) package.[7] However, unlike in prior style conversion work, care must be taken here to ensure that the training, validation and testing sections of the dataset contain disjoint sets of styles, which allows to monitor and evaluate the performance of the model on the one-shot style transfer task.

[7]https://www.pgmusic.com/

A. Chord chart generation

The first step was to acquire chord charts to use for accompaniment generation. Although chord charts in the BIAB format are available, we chose to create a new set of synthetic chord charts to use as input for BIAB. The main, purely practical motivation is that of complete control over the generation procedure, allowing to create a dataset that is balanced and diverse at the same time. The same cannot be easily achieved with existing BIAB files, as the closed file format effectively prevents automatic analysis and manipulation. Moreover, using a fully synthetic dataset enables us to release it publicly for reproducibility and to foster future research on musical styles.

We obtain the chord charts by sampling from a chord language model (LM) estimated on the iRb corpus [31], which contains chord charts of over a thousand jazz standards. For this purpose, each chord symbol is represented as a token composed of the chord's root expressed in relation to the song's main key (e.g. I, ♭VII), the chord's quality (e.g. min6, 7♭9), and its duration. We separate songs in major keys from songs in minor keys, and train a smoothed bigram LM on each of the two. A bigram LM models the conditional probability of a token given the previous token, and hence allows for sampling new token sequences in a Markovian fashion. This yields chord sequences similar to those from the iRb corpus, but the smoothing allows for producing unexpected chord transitions occasionally, increasing the diversity of the data.

Although the distribution of chords generated by the LM may not be appropriate for all musical styles, we assume that it will be sufficiently diverse to cover most styles thanks to the harmonic variability of jazz, and the LM smoothing. Nevertheless, expanding the dataset to chord charts from other genres could provide some benefits in future work.

After sampling a token sequence from the LM, we convert it to a chord chart (choosing the key at random from the distribution of keys in iRb) and add rhythmic variations randomly chosen from those available in BIAB (see e.g. bar 4 in Fig. 2,

featuring a G minor 7th chord with a 'shot' modifier and an eighth note 'push' – refer to Appendix B for a more detailed explanation). This is necessary in order for the trained system to be able to handle such variations; notably, we observed that the model from [10], trained without this kind of data enhancement, always produces continuous accompaniments even when the inputs contain prominent breaks.

This procedure resulted in 1,200 chord charts of 252 bars each for training, plus another 2×600 chord charts of 16 bars each as a validation and test set, respectively.

More details about the generation process can be found in Appendix B.

B. Accompaniment generation

The general procedure for accompaniment generation is adapted from [10]. BIAB allows generating a MIDI accompaniment for a given chord chart in one of the available styles. A style is essentially a set of human-defined patterns and rules for accompaniment generation which allow for some degree of freedom (randomness); one input can thus yield many different results. Each style typically consists of two substyles (A and B) with slightly different patterns intended for different sections of a song. The overall range of patterns in each style is relatively small, and thus corresponds to a specific *groove* rather than a broad category like genre. For instance, over 150 BIAB styles are categorized as Blues, bearing such names as 'Texas Blues – 12/8 Slow Blues' and 'Elvis1 – 50s Rock Shuffle-Blues'. Each style may contain up to 5 tracks (drums and up to 4 other instruments).

To enable generalization to unseen styles, the training data should contain as many styles as possible. We picked 1,476 MIDI styles from BIAB (their list can be found on the supplementary website) and reserved 2×20 of them for the validation and test set, respectively. All selected styles are in $\frac{4}{4}$ or $\frac{12}{8}$ time (i.e. with 4 beats per measure) and contain between 3 and 5 instruments, one of which is always drums. In this work, we treat the A and B substyles as separate styles, effectively doubling the number of styles in each part of the dataset. We used BIAB to render each generated chart in a few randomly chosen styles, so as to obtain about 500 measures of MIDI per style in each subset (train, val, test). We then split the files into fragments of 8 bars.

An example from the training set is visualized in Fig. 2.

C. Data feeding

After preparing the data, we need to form the triplets (A^S, B^T, A^T) required to train our style transfer model. As illustrated in Fig. 3a, this is done by looping over all possible pairs (A^S, A^T) such that $S \neq T$ for every 8-bar segment A, and for each such pair, choosing the fragment B^T at random from all segments in style T.

IV. Proposed Model

Our model is a neural network following the encoder-decoder pattern. It consists of two encoders – one for content and one for style – that process the two respective inputs, and a decoder that subsequently generates the output.

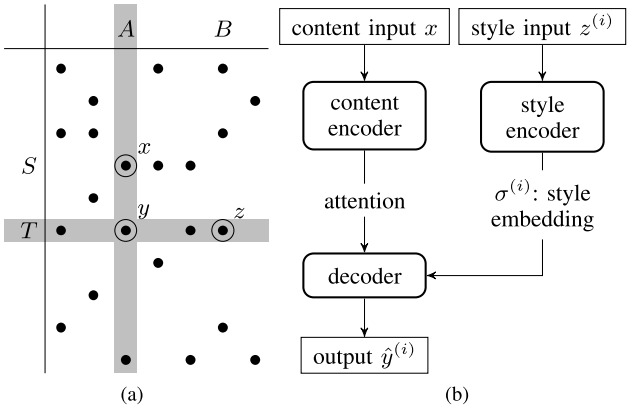


Fig. 3. (a) An illustration of how the training triplets $(x, z, y) = (A^S, B^T, A^T)$ are formed. Rows correspond to styles, columns to chord chart segments. Each dot represents an available example. (b) A high-level view of the model architecture. The model generates the i^{th} track of the output $\hat{y}$ given the content input x and the i^{th} track of the style input z. The overall output $\hat{y}$ is obtained by combining all the $\hat{y}^{(i)}$'s.

Since the output consists of several tracks corresponding to different instruments, we chose to break down the problem by modeling each track separately. While this could be done by training a dedicated model for each instrument as in [10], here we instead propose a single model allowing to generate all the accompaniment tracks, one at a time.

The model, depicted in Fig. 3b, operates as follows. The *content encoder* receives the content input x (containing all tracks), while the *style encoder* receives $z^{(i)}$, the i^{th} track of the style input. The decoder then combines the representations computed by the two encoders to generate the corresponding i^{th} output track, $\hat{y}^{(i)}$. Note that neither the index i nor any other information about the expected output track is given explicitly; instead, the model must infer all the necessary information from $z^{(i)}$ itself.

This design has the advantage that it allows to process any style input regardless of the number of tracks and without the need for any additional knowledge about them, and generalizes easily to instrument combinations unseen in the training data. On the other hand, it does not allow for interaction between tracks in the output, but this is only a minor limitation in our setting where the generation is densely conditioned on the content input.

A. Data representation

The style input $z^{(i)}$ and the output $\hat{y}^{(i)}$ use an event-based representation similar to MIDI, which has proven effective for neural music generation [10], [32], [33]. Unlike these previous works, where timing is encoded as time differences between events, we propose a *beat-relative encoding*, which we found to be more robust to timing errors during generation. In this representation, each track is encoded as a sequence of event tokens, each with one integer argument:

- `NoteOn(pitch)`, `DrumOn(pitch)`: begin a new note at the given pitch (0–127);[8]

[8]For `DrumOn` and `DrumOff`, the `pitch` argument (MIDI note number) encodes the percussion instrument (e.g. snare drum, hi-hat).

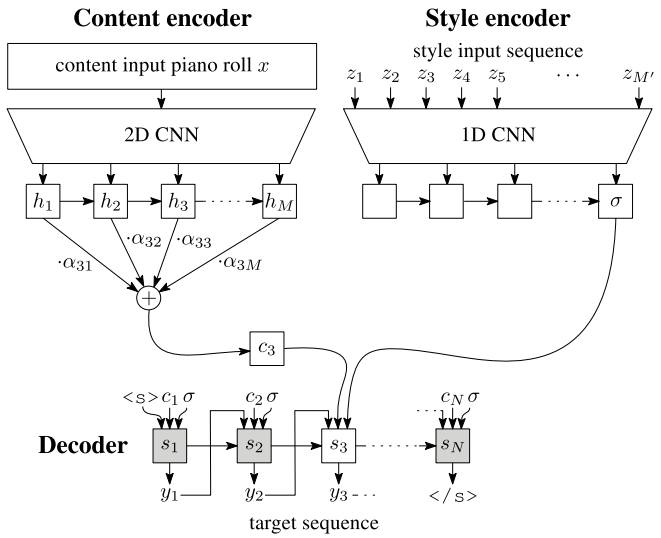


Fig. 4. A detailed view of the model architecture with focus on the 3rd decoding step, i.e. $n = 3$ (for illustration purposes). Each attention weight α_{nm} is calculated from the states s_{n-1} (here s_2) and h_m. The upper index (i) is omitted everywhere for simplicity.

- `NoteOff(pitch)`, `DrumOff(pitch)`: end the note at the given pitch (0–127, or $\star$ to end all active notes);
- `SetTime(units)`: set the time within the current beat, quantized to 12ths of a beat (0–11);
- `SetTimeNext(units)`: move to the next beat and set the time within that beat (0–11);
- `SetVelocity(units)`: set the velocity value, quantized to 8 levels (1–8).

The content input, on the other hand, may contain an arbitrary number of tracks that need to be processed simultaneously, which motivates a more compact representation. We use a piano roll matrix with 128 rows (all pitches allowed by the MIDI standard) and with 4 columns per beat. Each value in this matrix equals the total velocity of all notes with a given pitch active at a given time, normalized by the maximum velocity (127).

The main advantage of the piano roll representation over event-based representations is that its size depends only on the duration of the input (which is at most 8 measures in our setting), and not on the number of instruments or the style of the content input. This prevents excessive computational complexity, and should also aid the content encoder in deriving a representation which is style- and instrument-independent. For simplicity, we do not include drums in our piano rolls; this is a limitation, though in practice, we observe that the model is often able to use rhythmic cues from the other instruments for generating the target-style drum track.

B. Architecture details and training

A detailed view of the architecture is displayed in Fig. 4. Both encoders consist of a convolutional neural network (CNN) with exponential linear unit (ELU) activations [34], followed by a recurrent neural network (RNN) with a gated recurrent unit (GRU) [35]. The convolutional layers reduce the dimensionality of the initial representations, and the subsequent RNN serves to integrate information from a wider temporal context. The encoder architectures differ as follows, mostly due to the different input representations:

- The content encoder applies two consecutive 2D convolutional layers to the piano roll matrix, then flattens the resulting 3D feature map to obtain a sequence of 1024-dimensional vectors with two vectors per measure. This sequence is then fed to a GRU layer with 200 units, resulting in a sequence of 200-dimensional state vectors $h_1, \ldots, h_M$.
- The style encoder starts with a sequence of embeddings of the input tokens and applies three consecutive convolutional layers, compressing the sequence eight times, followed by a GRU with 500 units. Only the last GRU state is kept and used as a summary vector of the style (*style embedding*), denoted $\sigma^{(i)}$.

Note that since the style encoder only sees one track $z^{(i)}$ at a time, the style embedding $\sigma^{(i)}$ encodes the style of that specific track (which is to be transferred to the corresponding output track), rather than the style of the accompaniment as a whole.

The architecture of the decoder is adapted from sequence-to-sequence *(seq2seq)* models with attention [36] (originally developed for machine translation), and also uses a GRU. It works by always predicting the next token $y_n^{(i)}$ in the target sequence, conditioned on all the previous tokens $y_1^{(i)}, \ldots, y_{n-1}^{(i)}$, on the style embedding $\sigma^{(i)}$ and, via the attention mechanism, on the sequence of content encoder states $h_1, \ldots, h_M$. More precisely, the n-th decoder state $s_n^{(i)}$ is computed as

$$s_n^{(i)} = \mathrm{GRU}([Ey_{n-1}^{(i)}, c_n^{(i)}, \sigma^{(i)}], s_{n-1}^{(i)}),$$

where $[\cdot]$ denotes concatenation, E is the token embedding matrix (shared with the style encoder) and $c_n^{(i)}$ is the context vector (a weighted average of the content encoder states) computed by attention. The attention weights are predicted by a feed-forward neural network (trained jointly with the rest of the model) based on all the encoder states $h_1, \ldots, h_M$ and the previous decoder state $s_{n-1}^{(i)}$.

The role of the attention mechanism is to provide a *soft alignment* between the states of the content encoder and those of the decoder. In other words, it allows the decoder to *attend to* different encoder states as it generates the output. The need for such a mechanism arises mainly from the different time representations in the content input and the output. However, it also gives the decoder the flexibility to focus on different contexts (e.g. next measure, previous beat) as needed.

The output of the decoder cell is a softmax probability distribution over $y_n^{(i)}$. The model is trained end-to-end to minimize the cross entropy loss:

$$-\log P(y^{(i)} \mid x, z^{(i)}) = -\sum_{n=1}^{N} \log P(y_n^{(i)} \mid x, z^{(i)}, y_1^{(i)}, \ldots, y_{n-1}^{(i)}).$$

Note that we do not explicitly enforce mutually disentangled representations of content and style during training (e.g. by

means of additional terms in the loss function). Instead, the model is expected to learn to perform style transfer simply due to the way the training examples are constructed (as described in Section III).

Once trained, the decoder can be run in two modes:

a) *sampling* with temperature τ: sampling the token $\hat{y}_n^{(i)}$ at random from the softmax distribution. To control the randomness of the outputs, the logits are divided by a temperature parameter τ before applying the softmax.
b) *greedy decoding*: taking the most likely token $\hat{y}_n^{(i)}$ at every step (this is the limit sampling behavior when $\tau \to 0$).

We train the model using Adam [37] with a batch size of 64 and with exponential learning rate decay, halving the learning rate every 3 k batches (192 k training triplets). This somewhat aggressive strategy prevents overfitting by forcing the validation loss to plateau. We stop training in the middle of the first epoch (after 1.6 M triplets) where the improvement to the validation loss is already very small.

The complete hyperparameter settings are included with the source code.

V. Evaluation

In this section, we describe the objective metrics used in our experiments to evaluate style transfer performance. Functions for computing these metrics are included in the source code.

The goal of style transfer is to produce an output that matches both the content input and the style input. This leads to two complementary evaluation criteria: *content preservation* and *style fit*.

A. Content preservation

Given our definition of content, we would like the content preservation metric to capture the agreement in harmonic structure, which is the most important piece of information conveyed in a chord chart. We compute this metric as the frame-wise cosine similarity between chroma features of the content input and those of the output as proposed in [19]. Following [10], the chroma features are computed at a rate of 12 frames per beat and averaged over a 2 beat window with a stride of 1 beat. Unlike in [10], the metric is computed on the combined output rather than on individual tracks. This is because the harmony may not be fully captured by each individual instrument. Also, as the content input and the output may use completely different sets of instruments, it is not possible to pair them up meaningfully for this purpose.

B. Style fit

It has been previously proposed to characterize musical styles using statistics computed on musical events [16], [38], [39]. Based on this idea, we evaluate style fit by collecting so-called *style profiles* (inspired by the features proposed in [38]), and then measure how well they are matched by the style transfer outputs. Since the different accompaniment tracks within a given style are generated conditionally independently given the chord charts, we compute the style fit metrics for each track separately.

Metric	Observations	Bins
pw. time-pitch	$(\text{start}(b) - \text{start}(a),\ \text{pitch}(b) - \text{pitch}(a))$ $\in [0, 4) \times \{-20, -19, \ldots, 20\},\ a \neq b$	24×41
onset-duration	$(\text{start}(a) \bmod 4,\ \text{end}(a) - \text{start}(a))$ $\in [0, 4) \times [0, 2)$	24×12
onset-velocity	$(\text{start}(a) \bmod 4,\ \text{velocity}(a))$ $\in [0, 4) \times \{1, 2, \ldots, 127\}$	24×8
onset-drum	$(\text{start}(a) \bmod 4,\ \text{pitch}(a))$ $\in [0, 4) \times \{0, 1, \ldots, 127\}$	24×128

TABLE I
OBJECTIVE STYLE FIT METRIC DEFINITIONS. EACH METRIC IS COMPUTED AS A COSINE SIMILARITY BETWEEN FLATTENED 2D HISTOGRAMS OF THE OBSERVATIONS DEFINED IN THE MIDDLE COLUMN (VALUES THAT FALL OUTSIDE THE GIVEN RANGES ARE IGNORED). 'START' AND 'END' ARE THE ONSET AND OFFSET TIME IN BEATS, RESPECTIVELY, OF A GIVEN NOTE. 'PITCH' AND 'VELOCITY' DENOTE THE RESPECTIVE MIDI VALUES.

The statistics used to compute each metric are summarized in Table I. Firstly, we adopt our previously proposed metric [10], herein referred to as the *(pairwise) time-pitch* metric. To calculate it, we consider all pairs of notes a, b less than 4 beats apart present in a given set of outputs, then plot a 2D histogram with the time difference between the onsets of a and b on the x axis and the interval between a and b on the y axis. We then measure the cosine similarity between this style profile (flattened to a 984-dimensional vector) and a reference one computed on examples of the target style.

Clearly, the pairwise time-pitch metric is invariant to time shifts, does not account for note duration or velocity and is not suitable for drums. For this reason, we complement it with 3 additional metrics, computed on statistics of single notes: *onset-duration*, *onset-velocity* and *onset-drum*. These are defined analogously to the pairwise time-pitch metric, but instead relate the position of a note's onset within the measure to some other attribute of the same note (duration, velocity, and percussion instrument, respectively).

We measure the time-pitch and onset-duration metrics for non-drum instruments only; onset-drum is computed on drums only. Plots of example style profiles can be found in Fig. 12 at the end of the paper.

To evaluate a model using a given metric, we compute an aggregate style profile on all outputs of the model in each style, measure the cosine similarities to the reference profiles, and report the mean and standard deviation over all styles in the dataset. On non-synthetic datasets, where neither ground truth nor fine-grained style labels are available, we compute a separate profile for each output and measure its cosine similarity to the profile of the corresponding style input. We refer to these two ways of computing the metrics as *macro* and *nano* metrics, respectively.

VI. Experiments

We test the model on our synthetic test set and the Bodhidharma dataset [40], containing 950 MIDI recordings. The latter

dataset was picked on the grounds of being stylistically diverse and balanced, containing an equal number of recordings from 38 different genres. We filtered Bodhidharma to keep only music in $\frac{4}{4}$ time (660 files) and pre-processed it in the same way as the synthetic data, obtaining a total of 8,934 eight-bar segments. Additionally, we performed a certain kind of dynamic range compression by standardizing the velocity values in each segment, then scaling and shifting them to match the mean and variance computed on the training data; this is to compensate for a skewed distribution of velocity annotations in this dataset.

Note that both Bodhidharma and the synthetic test set contain styles unseen during training, and hence test the one-shot style transfer capabilities (i.e. the generalization to new styles).

We construct triplets (A^S, B^T, A^T) from the synthetic test set in the same way as during training (see Section III-C). On Bodhidharma, where targets are not available, we form input pairs (A^S, B^T) by choosing B^T randomly (with replacement) from the entire dataset.

The model is tested in both the greedy decoding mode and the sampling mode with $\tau = 0.6$ (observed to yield good results in preliminary experiments on the validation set).

For comparison, we also evaluate the following trivial systems:

- CP-CONTENT: copies the content input to the output; expected to have perfect performance on content preservation, but poor on style fit.
- CP-STYLE: copies the style input to the output; expected to have perfect performance on style fit, but poor on content preservation.
- ORACLE: retrieves the 'ground-truth' target-style segment generated by BIAB, if available; this should provide a more realistic upper bound on all metrics.

Evaluating CP-CONTENT for style fit presents two pitfalls. The first is that the content inputs for one target style may themselves have several different styles. To avoid conflating them, we aggregate the style profiles over each of them separately; we then have one data point for each source-target style pair. The second problem is that, as the content input may contain a different set of instruments than the target, we do not know which reference to use for each track. For this reason, we evaluate each track of the content input against each track of the target style and report the maximum value for each target-style track.

We note that a direct comparison of our approach with prior style conversion work (especially [10]) is unfortunately not possible. The main reason is that a style translation (conversion) system cannot be conditioned on unseen styles since it has no style encoder.

In the rest of this section, we present the main evaluation results (Section VI-A) and an ablation study (Section VI-C), provide some observations about practical use of the proposed system (Section VI-B), evaluate the proposed style similarity metrics (Section VI-D), and explore the properties of the style embedding space (Sections VI-E and VI-F).

A. Evaluation results

We evaluate our model using the metrics described in Section V. First, we present in Fig. 5 the results on the synthetic test set. In terms of the content preservation metric, Grv2Grv achieves perfect results (on par with ORACLE), and the gap with respect to CP-STYLE is large (for this metric, CP-STYLE can be thought of as a 'random baseline', since the style input is chosen independently of the content input).

Even though on style fit metrics (*macro* version), Grv2Grv is not able to reach the performance of ORACLE or CP-STYLE (close to 1), it scores higher than CP-CONTENT. This means that the output is, on average, closer to the target style than the content input, and hence the style transfer is at least partly successful. The large range of values of CP-CONTENT is explained by the fact that the content input may (or may not) already be in a style which is similar to the target.

We may notice that the performance of Grv2Grv on the onset-duration metric is considerably lower than on the other style fit metrics, which suggests that it does not model note duration well. However, note duration is arguably perceptually less important than other features (in particular, those related to onset time and pitch).

The two decoding modes of our model (*greedy* and *sampling*) achieve similar results on all metrics, but the sampling mode consistently performs slightly better on style fit. This is not unexpected, given the fact that greedy decoding always picks the most likely event, whereas sampling draws events randomly from the learned conditional distribution. This means that sampling should allow the model to better cover the distribution of features of the style, leading to a higher score on our metrics.

We now turn to the results on the Bodhidharma dataset. In this case, we need to use the *nano* style fit metrics (as explained in Section V). To allow for comparison, we compute the *nano* metrics on both datasets (synthetic and Bodhidharma) and display the results alongside in Fig. 6. First of all, we can see the style metrics drop and become more 'blurred' with respect to their *macro* versions (Fig. 5). For example, on the synthetic dataset, ORACLE decreases from 1.00 to 0.75 on average on time-pitch, and Grv2Grv (sampling) drops from 0.84 to 0.62; moreover, sampling no longer seems consistently better than greedy decoding on either dataset. This is probably due to the fact that a single 8-bar example cannot capture how the characteristic patterns of the style manifest in all the different contexts (i.e. in different chord progressions); this will often lead to mismatching style profiles, and therefore noisy results.

On Bodhidharma, the scores are generally substantially lower than on the synthetic test set, which indicates that the dataset is more challenging for our model. The performance of CP-CONTENT on style fit metrics is lower as well; this means that the differences between styles in this dataset are larger, making the task more difficult. However, our model still beats the baselines – CP-STYLE on the content preservation metric and CP-CONTENT on the style fit metrics – the former being outperformed by a large margin.

One other factor to consider when reading the results is that Bodhidharma contains full arrangements including melodies,

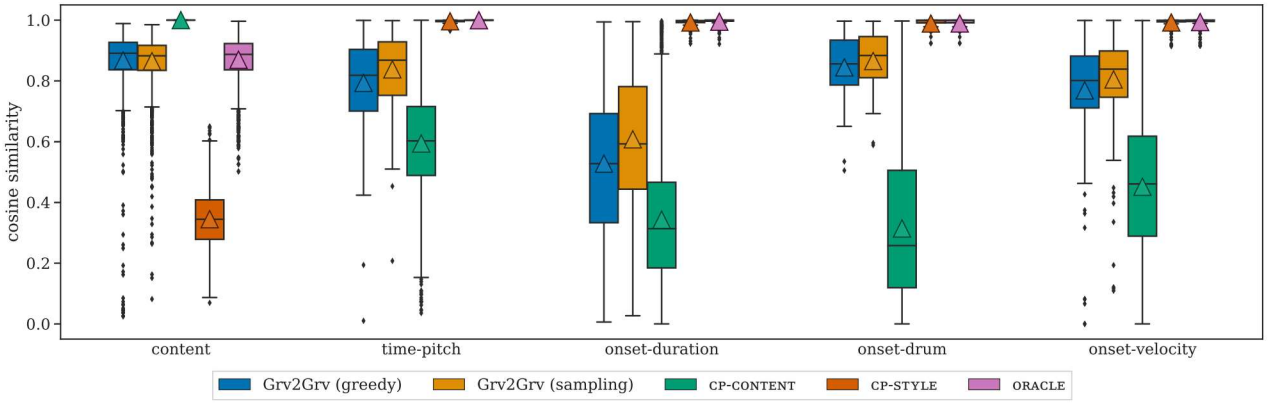


Fig. 5. Evaluation results on the synthetic test set. The two leftmost results in each group are those of our main proposed model. The triangles indicate the mean. We use the *macro* variant of the style metrics, i.e. each data point corresponds to one of the target styles.

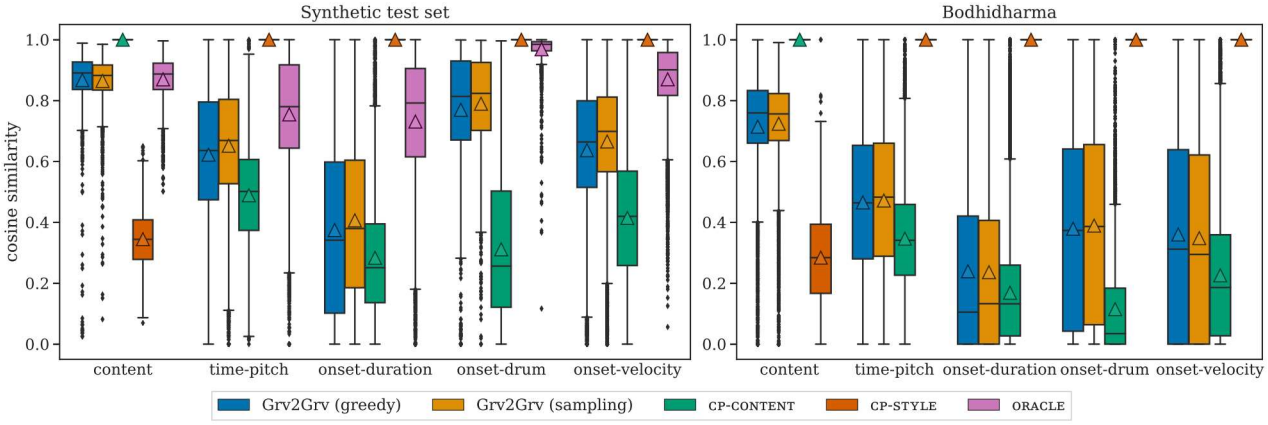


Fig. 6. Evaluation results on both test sets. Style metrics are computed in the *nano* variant, i.e. each data point corresponds to a single example. This results in higher variance than in Fig. 5, but enables us to evaluate on Bodhidharma.

as well as polyphonic music. This leads to the following issues which may, in part, also be responsible for the different results between the synthetic test set and Bodhidharma:

1) When presented with a melody line as its style input, Grv2Grv – being trained on accompaniments – will instead attempt to generate a pseudo-accompaniment track in the style of the melody. Such tracks are generally unwanted and should, in practical applications, be removed in a manual pre- or post-processing step.
2) The additional (non-accompaniment) tracks in the content input can make the reference chroma features more noisy, which could contribute to the drop in the content preservation metric.

B. User perspective

Upon listening to the outputs, we note that they are, for the most part, musically meaningful, and follow the harmony of the content input very accurately (this is true even on the Bodhidharma dataset, despite the somewhat lower content preservation values). They also generally match the overall 'feel' of the target style, especially the rhythmic feel, pitch ranges and voicing types of the different tracks, but sometimes fail to reproduce some of the exact patterns characteristic for the style. We also observe that the outputs produced by random sampling tend to sound more interesting than those resulting from greedy decoding, which are often too simplistic and do not capture the real variability of the target style. This is consistent with the results in Fig. 5.

We also note that for best results, human selection and/or pre-processing of the inputs is often required. Firstly, entire pieces cannot be used as style inputs; instead, one needs to select a short segment (ideally 8 bars), and not every such segment is equally representative of the style of the piece. Secondly, as mentioned in the previous section, some tracks should usually be excluded from the style input (or, equivalently, the output) because they are not part of the accompaniment. This is also true for heavily interdependent tracks (e.g. instruments playing in unison or creating parallel harmonies), which, if generated independently, will not have the desired effect.

Finally, to create a complete arrangement (cover), the generated accompaniment needs to be, at the least, combined with the melody of the content input. Even though it is conceivable to extract the melody automatically, it is a non-trivial task that lies outside the scope of our work.

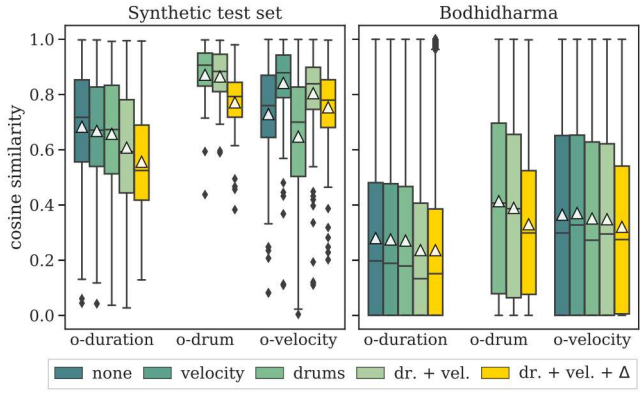


Fig. 7. Results of the ablation study. 'Dr. + vel.' ('drums + velocity') is the full Grv2Grv model; 'none' models neither drums nor velocity. Δ stands for the Δ-encoding. All models are evaluated in sampling mode. The synthetic test set uses *macro* metrics as in Fig. 5, Bodhidharma uses *nano* metrics as in Fig. 6.

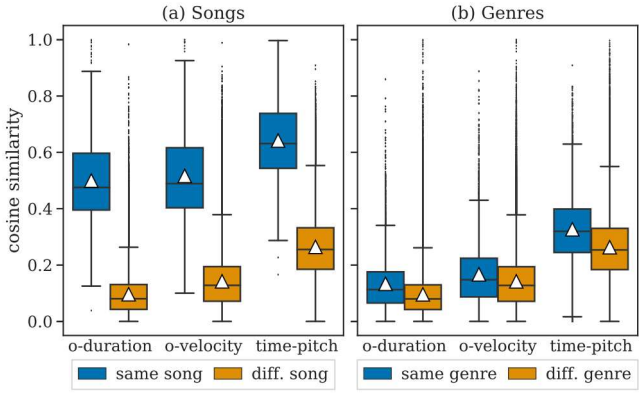


Fig. 8. Style similarities between pairs of segments from the Bodhidharma dataset. Plot (a) contrasts similarities within and across songs, while plot (b) contrasts similarities within and across the 38 genres in the Bodhidharma dataset. The values have been averaged so that every data point corresponds to a single pair of songs.

C. Ablation study

Drums and velocity: Compared to our previous style translation work [10], Grv2Grv adds the ability to generate drums and to model velocity. In this section, we attempt to answer the question whether these additional tasks affect the performance of the model in other areas. To this end, we perform an ablation study where we re-train and evaluate the model while

(a) excluding the drum track,
(b) omitting the `SetVelocity` tokens and making the content input piano roll binary (containing only the values 0 and 1 indicating whether a note is present), or
(c) both of the above.

In cases (b) and (c), we post-process the output by setting the velocity of all notes equal to the average velocity of the style input notes.

Fig. 7 (four leftmost bars in each group) shows the results on three selected metrics on both of our test sets. Firstly, removing the capability to generate drums obviously causes the onset-drum metric to become undefined. However, it slightly improves the performance on the other metrics as the task becomes simpler.

Similarly, eliminating velocity seems to slightly improve the performance on the metrics unrelated to velocity (onset-duration and onset-drums). This may be explained by the fact that removing the velocity tokens makes the sequences shorter, reducing the length of the context that needs to be considered by the decoder, and hence making the problem easier overall.

In terms of the onset-velocity metric, the velocity-enabled models outperform the velocity-free ones on both datasets (although the latter still yield relatively good results thanks to our heuristic, which copies the average velocity from the style input).

Beat-relative encoding: We are also interested in validating our proposed beat-relative encoding (see Section IV-A), designed to overcome the limitations of representing timing as time differences between consecutive events, such as in [10]. For this reason, we include in our ablation study a version of the Grv2Grv model which employs the encoding from [10], which we will refer to as the Δ-*encoding*. In practice, this means that all `SetTime` and `SetTimeNext` tokens are replaced with `TimeShift` tokens, which encode the time difference to the previous event, rather than the last beat.

The results, displayed in Fig. 7 as the rightmost bar in each group, show that the Δ-encoding is mostly outperformed by the beat-relative encoding. On inspection, we confirm that the outputs generated with the Δ-encoding are prone to rapidly accumulating timing errors (as noted in [10]). This frequently results in the individual output tracks getting completely desynchronized from the content input, as well as from each other. On the other hand, tracks generated with the beat-relative encoding are mostly rhythmically coherent, even with high sampling temperatures.

D. Validation of style fit metrics

In this section, we aim to validate the proposed style similarity metrics. To this end, we compute the similarities between all pairs of 8-bar segments in Bodhidharma and compare inter- vs. intra-class similarities. For simplicity, we limit the analysis to pitched instruments, and therefore include only the onset-duration, onset-velocity and time-pitch metrics (the latter is adopted from prior work, but has not yet been evaluated on non-synthetic data). Since we apply the metrics to each instrument track individually and it may not be possible to unambiguously match the tracks of two given segments, we simply compute the similarities on all pairs of instruments belonging to the same MIDI instrument group[9] and average the results.

First, we compare in Fig. 8 (a) the similarities between segments from the same song to similarities between segments from different songs (regardless of genre). The same-song similarities are substantially higher for all three metrics, which is in line with our understanding of style as a set of characteristics pertaining to a particular artist or song. Secondly,

[9]The General MIDI specification [41] defines 16 instrument groups such as *Piano*, *Bass*, *Strings* or *Reed*, each comprising 8 instruments. This grouping does not include drums, which exist on a dedicated MIDI channel.

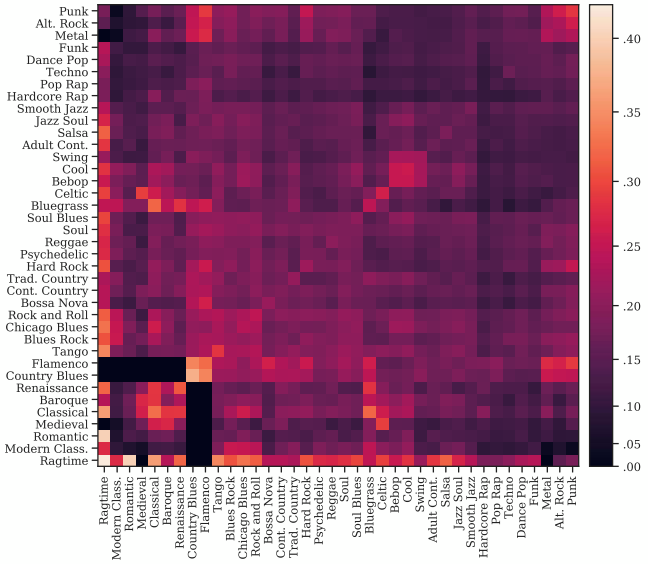


Fig. 9. Pairwise similarities between genres from the Bodhidharma dataset. The values are averaged over all 3 metrics and all pairs of segments. Values on the diagonal do not include pairs of segments from the same song. The order of rows and columns has been determined using hierarchical clustering.

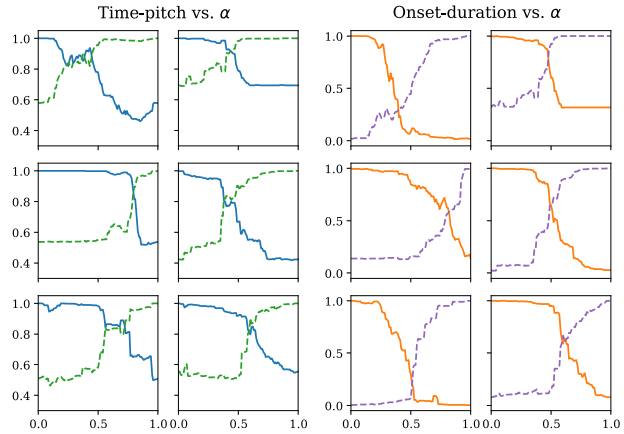
we would also expect our metrics to capture at least some characteristics of genres; this is demonstrated in plot (b), which shows that the average similarity of segments from the same genre (*excluding* segments from the same song) is higher than the average similarity across genres, again on all three metrics.

The results on genres are further detailed in Fig. 9, showing the similarity values – averaged over all 3 metrics – between all pairs of genres (again, pairs of segments from the same song are excluded). Despite the generally low and noisy values, we can find clear clusters of similar genres, e.g.: Swing, Cool and Bebop; Metal, Alternative Rock and Punk; as well as a classical music cluster.

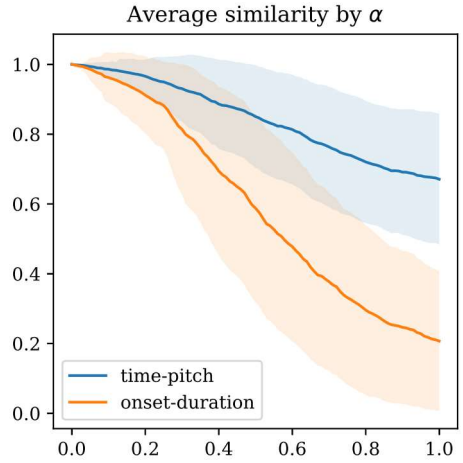
E. Style interpolation

The learned style embedding space enables us to blend styles by linearly interpolating their embeddings. We sampled 100 pairs of bass tracks from the synthetic test set and encoded them using the style encoder to obtain their respective embeddings. For each embedding pair σ_0, σ_1, we conditioned the decoder, in turn, on vectors of the form $(1-\alpha)\cdot\sigma_0+\alpha\cdot\sigma_1$ for α evenly spaced between 0 and 1. Each time, we ran the model over a batch of 20 content inputs in greedy decoding mode and computed the style similarities of the outputs to those obtained at the endpoints σ_0, σ_1 (i.e. for $\alpha = 0, 1$).

Fig. 10 shows the results as a function of α. Interestingly, the similarity curves in (a) are rather monotonic, yet staircase-like (with continuous but steep transitions). This suggests that the style space is divided into soft regions with little internal variation (manifesting as plateaux in the plots), and that regions closer to each other correspond to more similar styles. Plot (b) in Fig. 10 then displays the behavior on average, showing that, consistently with the above observations, the similarity to the initial style decreases with increasing α.



(a) Examples for specific style pairs; the solid line and the dashed line show the similarity to the outputs generated from σ_0 (i.e. for $\alpha = 0$) and σ_1 (for $\alpha = 1$), respectively.



(b) α-wise average and standard deviation of the metrics plotted in (a) (solid line) over 100 style pairs.

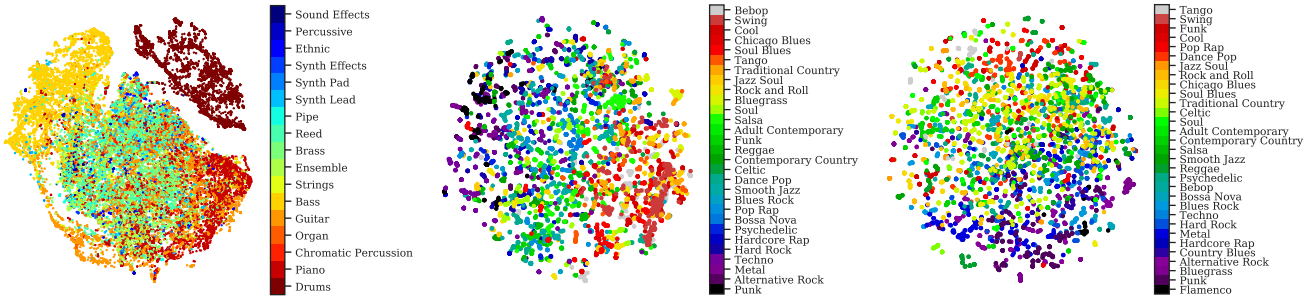
Fig. 10. Style similarity to interpolation endpoints as a function of the interpolation coefficient α.

Example outputs from this experiment are provided on the supplementary website.[4]

F. Style embedding visualization

To further explore the properties of the style embedding space, we visualize in Fig. 11 embeddings of segments from the Bodhidharma dataset, using PCA followed by t-SNE [42] for dimensionality reduction. Since the style embeddings encode the characteristics of individual tracks, we may expect the embedding space to be primarily clustered by instrument. This is confirmed by plot (a), showing that drums and bass are clearly separated from the rest. Other instruments do not form such clear-cut clusters, arguably because a single instrument may have different functions (e.g. playing chords vs. melody).

Plots (b) and (c) then show the distributions of genres for two selected instrument groups.[9] Even though there are no pronounced clusters, we can observe that the individual genres are fairly localized.



(a) All embeddings by instrument (b) Bass embeddings by genre (c) Guitar embeddings by genre

Fig. 11. PCA + t-SNE projections of style embeddings. Each point corresponds to a single track of a single segment from Bodhidharma. The colors in (a) represent MIDI instrument groups with drums added as an extra category. The colors of the genres in (b) and (c) are determined by the locations of their centroids.

VII. Limitations and Future Directions

While our supervised approach to music style transfer has proven effective, some limitations remain. Possibly the most important shortcoming is the fact that it is limited to accompaniments and does not account for interaction between different instruments. This arises from the nature of the synthetic training data, which is generated purely from chord charts and without strong inter-track dependencies. An approach capable of overcoming this limitation will likely need to be able to take advantage of the available non-parallel 'real-world' music data by means of unsupervised or semi-supervised learning (possibly still using parallel synthetic data for supervision). It will also need to employ a model capable of generating all tracks jointly – without strong independence assumptions – in order to capture the interactions between them; this is in principle possible (as shown e.g. in [33], [43], [44]), albeit more computationally demanding.

Moreover, it is apparent that there is, generally speaking, still room for improvement in the quality of the outputs – in particular, in the generalization to novel styles. This sub-optimal one-shot generalization capability may be due to an insufficient number of styles in the training set (in spite of our efforts to make this number as large as possible) or a discrepancy between BIAB styles and the test inputs (which likely exists, despite BIAB styles being fairly realistic and diverse). We believe that both problems may be alleviated by training at least partially on open-domain, non-BIAB data as outlined above, which we leave as future work.

Lastly, the applicability of Grv2Grv is limited by the fact that it works with symbolic music only. An extension capable of processing audio inputs, or even producing audio end-to-end, would certainly be interesting and is left as another natural next step.

VIII. Conclusion

In this paper, we presented Groove2Groove, a one-shot style transfer method for accompaniment styles in the symbolic music domain. Atypically for the style transfer task, we approached it using end-to-end supervised learning, proposing an encoder-decoder neural network along with a parallel training data generation scheme. We have demonstrated the performance of our model on both synthetic and non-synthetic inputs in new styles, and shown that it behaves meaningfully when its style representation is manipulated. We have also conducted an ablation study, which highlighted some strengths and limitations of our approach.

We hope that our work will help attract more attention to the challenging problem of music style transfer and inspire new research on the subject.

Appendix

A. Online demo

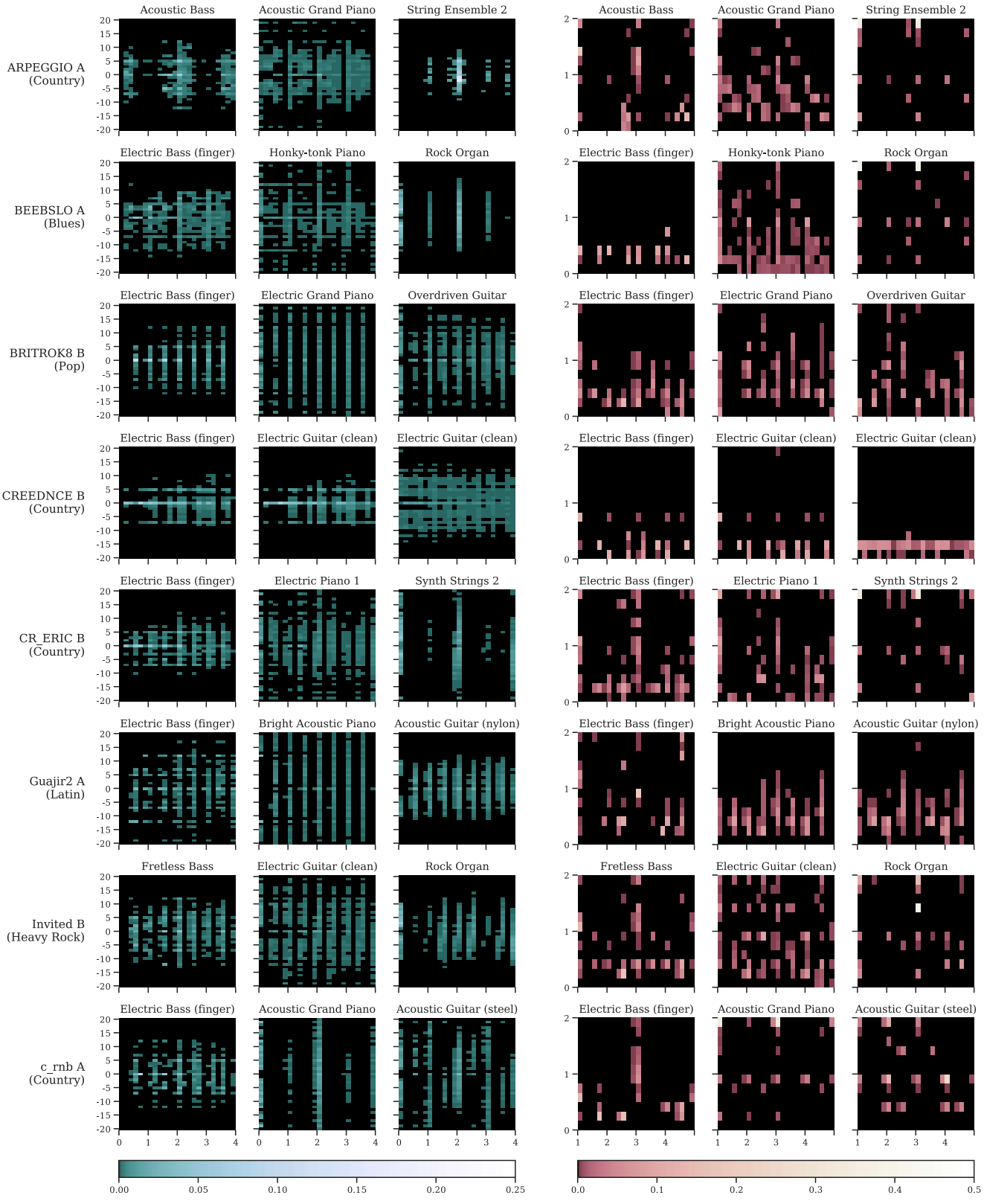
Our supplementary website[4] contains an interactive demo, which allows to run Grv2Grv on arbitrary MIDI files uploaded by the user. To take into account the considerations mentioned in Section VI-B, we enable the user to select appropriate sections and tracks from the input files, and also provide a facility to recombine the output with any of the tracks of the content input. We also provide pre-generated examples, created using the velocity-free model from the ablation study (Section VI-C) with inputs from the Bodhidharma dataset.

B. Data generation details

To obtain the training data for the bigram language models mentioned in Section III-A, we extract the necessary features from the iRb corpus using the `jazzparser.sh` script provided by [31]. The LMs use Lidstone (add-ε) smoothing with $\varepsilon = 0.01$. We sample sequences from the LM repeatedly and concatenate them until we reach the maximum number of measures. We then convert each token to a chord symbol. We optionally add some of the following modifiers (defined by BIAB) to each chord: (a) *push*: creates an 8$^{\text{th}}$ note anticipation; (b) *hold*: all instruments hit the chord simultaneously and hold it until the next chord symbol; (c) *shot*: all instruments play the chord *staccato*, followed by silence; (d) *rest*: all instruments are silent until the next chord. These modifiers are turned on and off at random, with probabilities chosen so that they are scattered sparsely throughout the chord charts.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions, which aided us in improving the manuscript.



(a) Time-pitch (x: time difference, y: pitch difference)

(b) Onset-duration (x: onset time, y: duration)

Fig. 12. Style profile examples from the synthetic test set. Each row corresponds to one style, with the short name and genre of the style displayed on the left. (More information about the styles can be found on the supplementary website.) Each plot is labeled with the MIDI instrument of the track on which it was computed.

REFERENCES

[1] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *CVPR*, 2016.

[2] D. Chen, J. Liao, L. Yuan, N. Yu, and G. Hua, "Coherent online video style transfer," in *ICCV*, 2017.

[3] R. J. Skerry-Ryan, E. Battenberg, Y. Xiao, Y. Wang, D. Stanton, J. Shor, R. J. Weiss, R. Clark, and R. A. Saurous, "Towards end-to-end prosody transfer for expressive speech synthesis with Tacotron," in *ICML*, 2018.

[4] A. P. Noam Mor, Lior Wold and Y. Taigman, "A universal music translation network," in *ICLR*, 2019.

[5] T. Shen, T. Lei, R. Barzilay, and T. S. Jaakkola, "Style transfer from non-parallel text by cross-alignment," in *NIPS*, 2017.

[6] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *CVPR*, 2017.

[7] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*, 2017.

[8] I. Malik and C. H. Ek, "Neural translation of musical style," *ArXiv*, vol. abs/1708.03535, 2017.

[9] E. Nakamura, K. Shibata, R. Nishikimi, and K. Yoshii, "Unsupervised melody style conversion," in *ICASSP*, 2019.

[10] O. Cífka, U. Şimşekli, and G. Richard, "Supervised symbolic music style translation using synthetic data," in *ISMIR*, 2019.

[11] F.-F. Li, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 594–611, 2006.

[12] D. J. Rezende, S. Mohamed, I. Danihelka, K. Gregor, and D. Wierstra, "One-shot generalization in deep generative models," in *ICML*, 2016.

[13] S. Dai, Z. Zhang, and G. Xia, "Music style transfer: A position paper," in *Proceedings of the 6th International Workshop on Musical Metacreation (MUME)*, 2018.

[14] Y.-N. Hung, I. P. Chiang, Y.-A. Chen, and Y.-H. Yang, "Musical composition style transfer via disentangled timbre representations," in *IJCAI*, 2019.

[15] F. Pachet and P. Roy, "Non-conformant harmonization: the Real Book in the style of Take 6," in *ICCC*, 2014.

[16] G. Hadjeres, J. Sakellariou, and F. Pachet, "Style imitation and chord invention in polyphonic music with exponential families," *ArXiv*, vol. abs/1609.05152, 2016.

[17] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "MIDI-VAE: Modeling dynamics and instrumentation of music with applications to style transfer," in *ISMIR*, 2018.

[18] G. Brunner, Y. Wang, R. Wattenhofer, and S. Zhao, "Symbolic music genre transfer with CycleGAN," in *ICTAI*, 2018.

[19] W.-T. Lu and L. Su, "Transferring the style of homophonic music using recurrent neural networks and autoregressive models," in *ISMIR*, 2018.

[20] C.-Y. Lu, M.-X. Xue, C.-C. Chang, C.-R. Lee, and L. Su, "Play as you like: Timbre-enhanced multi-modal music style transfer," in *AAAI*, 2018.

[21] S. Huang, Q. Li, C. Anil, X. Bao, S. Oore, and R. B. Grosse, "TimbreTron: A WaveNet(CycleGAN(CQT(Audio))) pipeline for musical timbre transfer," in *ICLR*, 2019.

[22] G. Hadjeres and F. Pachet, "DeepBach: a steerable model for Bach chorales generation," in *ICML*, 2017.

[23] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. C. Courville, and D. Eck, "Counterpoint by convolution," in *ISMIR*, 2017.

[24] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel, "Encoding musical style with transformer autoencoders," *ArXiv*, vol. abs/1912.05537, 2019.

[25] S. Lattner and M. Grachten, "High-level control of drum track generation using learned patterns of rhythmic interaction," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2019.

[26] E. Grinstein, N. Q. K. Duong, A. Ozerov, and P. Pérez, "Audio style transfer," in *ICASSP*, 2018.

[27] J. Driedger, T. Prätzlich, and M. Müller, "Let it Bee – towards NMF-inspired audio mosaicing," in *ISMIR*, 2015.

[28] C. J. Tralie, "Cover song synthesis by analogy," in *ISMIR*, 2018.

[29] A. Zils and F. Pachet, "Musical mosaicing," in *COST G-6 Conference on Digital Audio Effects (DAFX-01)*, 2001.

[30] Y. Zhang, W. Cai, and Y. Zhang, "Separating style and content for generalized style transfer," in *CVPR*, 2018.

[31] Y. Broze and D. Shanahan, "Diachronic changes in jazz harmony," *Music Perception: An Interdisciplinary Journal*, vol. 31, no. 1, pp. 32–45, 2013.

[32] I. Simon and S. Oore, "Performance RNN: Generating music with expressive timing and dynamics," Magenta Blog, 2017. [Online]. Available: https://magenta.tensorflow.org/performance-rnn

[33] C. Payne, "MuseNet," OpenAI, 2019. [Online]. Available: https://openai.com/blog/musenet/

[34] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *ICLR*, 2016.

[35] K. Cho, B. van Merrienboer, Çaglar Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *EMNLP*, 2014.

[36] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[38] C. McKay, "Automatic genre classification of MIDI recordings," M.A. Thesis, McGill University, 2004.

[39] J. Sakellariou, F. Tria, V. Loreto, and F. Pachet, "Maximum entropy models capture melodic styles," *Scientific Reports*, 2017.

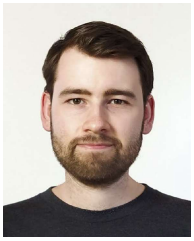
[40] C. McKay and I. Fujinaga, "The Bodhidharma system and the results of the MIREX 2005 symbolic genre classification contest," in *ISMIR*, 2005.

[41] "General MIDI system level 1," MIDI Manufacturers Association, 1991.

[42] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[43] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *ICML*, 2018.

[44] J. Thickstun, Z. Harchaoui, D. P. Foster, and S. M. Kakade, "Coupled recurrent models for polyphonic music composition," in *ISMIR*, 2019.



Ondřej Cífka received his Bachelor's degree in Computer Science in 2016 and his Master's degree in Computational Linguistics in 2018, both from Charles University, Prague, Czechia. Currently a Marie Skłodowska-Curie fellow within the MIP-Frontiers training network, he is pursuing a Ph.D. at Telecom Paris on the topic of context-driven music transformation. His research interests include music information processing, audio processing, natural language processing, and machine translation.

Umut Şimşekli received his PhD degree in 2015 from Boğaziçi University, İstanbul, Turkey. His current research interests are in large-scale Bayesian machine learning, diffusion-based Markov Chain Monte Carlo, non-convex optimization, and audio/music processing. He was a visiting faculty at the University of Oxford, Department of Statistics and currently he is an assistant professor within the Signals, Statistics, and Machine Learning Group at Telecom Paris, and a member of the IEEE Audio and Acoustic Signal Processing technical committee.

Gaël Richard (SM'06, F'17) received the State Engineering degree from Telecom Paris, France in 1990, and the Ph.D. degree from University of Paris-Sud, in 1994. He then spent two years at the CAIP Center, Rutgers University, Piscataway, NJ, in the Speech Processing Group of Prof. J. Flanagan. From 1997 to 2001, he successively worked for Matra-Nortel, France, and for Philips, France. He then joined Telecom Paris, where he is now a Full Professor and Head of the Image, Data, Signal department. Co-author of over 250 papers and inventor in 10 patents, his research interests are mainly in the field of speech and audio signal processing and include source separation, machine learning methods for speech/audio/music signals and music information retrieval.