Self-Supervised VQ-VAE for One-Shot Music Style Transfer

Ondřej Cífka, Alexey Ozerov, Umut Şimşekli, Gael Richard

 DOI: 10.1109/ICASSP39728.2021.9414235

SELF-SUPERVISED VQ-VAE FOR ONE-SHOT MUSIC STYLE TRANSFER

Ondřej Cífka[⋆†] *Alexey Ozerov*[†] *Umut Şimşekli*[‡⋆] *Gaël Richard*[⋆]

[⋆]LTCI, Télécom Paris, Institut Polytechnique de Paris, France
[†]InterDigital R&D, Cesson-Sévigné, France [‡]Inria/ENS, Paris, France

ABSTRACT

Neural style transfer, allowing to apply the artistic style of one image to another, has become one of the most widely showcased computer vision applications shortly after its introduction. In contrast, related tasks in the music audio domain remained, until recently, largely untackled. While several style conversion methods tailored to musical signals have been proposed, most lack the 'one-shot' capability of classical image style transfer algorithms. On the other hand, the results of existing one-shot audio style transfer methods on musical inputs are not as compelling. In this work, we are specifically interested in the problem of *one-shot timbre transfer*. We present a novel method for this task, based on an extension of the vector-quantized variational autoencoder (VQ-VAE), along with a simple self-supervised learning strategy designed to obtain disentangled representations of timbre and pitch. We evaluate the method using a set of objective metrics and show that it is able to outperform selected baselines.

Index Terms— Style transfer, music, timbre, self-supervised learning, deep learning

1. INTRODUCTION

Neural style transfer techniques, originally proposed for images [1, 2], allow applying the 'artistic style' of one image to another. Recently, there has been increased interest in developing similar methods for music, and promising works in this domain have begun to emerge. Especially compelling are results achieved by several recent works on timbre conversion [3, 4, 5, 6], leading to entertaining applications.[1] However, a common property of these deep learning-based methods is that they require training for each individual target instrument. Consequently, the set of target instruments available in these systems is typically small, as adding new ones is a time-consuming process which depends on the availability of clean training data.

In the present work, we instead propose to tackle a more general task, which we refer to as *one-shot timbre transfer*.[2] Borrowing the terminology of image style transfer, our goal is to transfer the timbre of a *style input* onto a *content input* while preserving the pitch content of the latter. To this end, we develop a single generic model capable of encoding pitch and timbre separately and then combining their representations to produce the desired output.

Unlike many previous music style transformation works (e.g. [7, 5, 8, 9]), we neither assume the training data to be paired or otherwise annotated, nor do we rely on existing models or algorithms to create artificial annotations (e.g. pitch contours or timbre-related descriptors). This leads to the need for data-driven *disentanglement* of the pitch and timbre representations learned by the model. In this work, we propose to perform this disentanglement using a combination of discrete representation learning (via an extension of the vector-quantized variational autoencoder, or VQ-VAE [10]), self-supervised learning, and data augmentation.

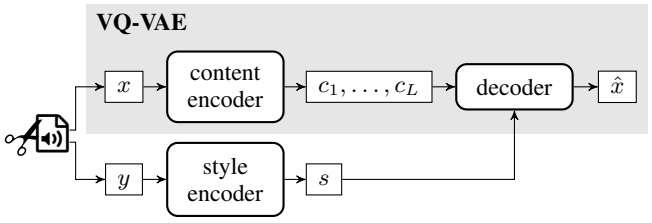

Fig. 1. A high-level depiction of the proposed method. We extract pairs of segments from audio files and use them for self-supervised learning of a VQ-VAE with an additional style encoder. The content representation $c_1, \ldots, c_L$ is discrete, the style representation s is continuous.

Our contributions can be summarized as follows:

- We present the first neural model for one-shot instrument timbre transfer. The model operates via mutually disentangled pitch and timbre representations, learned in a self-supervised manner without the need for annotations.
- We train and test our model on a dataset where each recording contains a single, possibly polyphonic instrument. Using a set of newly proposed objective metrics, we show that the method constitutes a viable solution to the task, and is able to compete with baselines from the literature. We also provide audio examples for perceptual comparison by the reader.[3]
- Since our approach to disentanglement is largely data-driven, it should be extensible to other music transformation tasks, such as arrangement or composition style transfer.
- Our source code is available online.[4]

2. RELATED WORK

Prior work on our topic is rather limited. To our knowledge, most existing works that fall under our definition of one-shot music timbre transfer [11, 12, 13] are based on non-negative matrix factorization (NMF) combined with *musaicing* [14] (a form of concatenative synthesis). Other works on audio style transfer [15, 16] adapt the original image style transfer algorithm [1] to audio, but focus on

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).

[1] `https://g.co/tonetransfer`

[2] Similarly to [7], we supplement the somewhat ambiguous term 'timbre transfer' with the attribute 'one-shot' to specify that we aim to imitate the timbre of *one single example* presented at test time.

[3] `https://adasp.telecom-paris.fr/s/ss-vq-vae`

[4] `https://github.com/cifkao/ss-vq-vae`

'sound textures' rather than timbre. In particular, the style representation used in these works is heavily pitch-dependent, which makes it unsuitable for representing musical timbre. Finally, [9] focuses on singing voice conversion, adopting vocoder-based techniques developed for speech.

Speaking more generally of *timbre* or *style conversion*, several methods were recently proposed for musical audio [3, 4, 5, 6]. While these approaches achieve remarkable output quality, they cannot be considered one-shot as they only allow for conversion to the (small) set of styles present in the training data. Moreover, unlike our methods, they require training a separate decoder for each target style; in particular, [3] report unsuccessful attempts to train a single decoder conditioned on the identity of the target instrument.

Other recent works [17, 18, 19, 20] are related to ours in that they also learn a continuous timbre representation which allows for audio generation, but are limited to the simple case of isolated notes.

3. BACKGROUND

3.1. Vector-quantized autoencoder (VQ-VAE)

The VQ-VAE [10] is an autoencoder with a discrete latent representation. It consists of an encoder, which maps the input x to a sequence z of discrete codes from a codebook, and a decoder, which tries to map z back to x. Using discrete latent codes places a limit on the amount of information that they can encode. The authors successfully exploit this property to achieve voice conversion. In this work, we follow a similar path to achieve music style transfer.

Formally, the encoder first outputs a sequence $E(x) \in \mathbb{R}^{L\times D}$ of D-dimensional feature vectors, which are then passed through a quantization (discretization) operation Q which selects the nearest vector from a discrete embedding space (codebook) $e \in \mathbb{R}^{K\times D}$:

$$z_i = Q(E_i(x)) = \underset{e_j, 1\leq j\leq K}{\arg\min} \big\|E_i(x) - e_j\big\|. \tag{1}$$

The model is trained to minimize a reconstruction error $\mathcal{L}_{\text{ae}}$ between the input x and the output of the decoder $D(Q(E(x)))$. The backpropagation of its gradient through the discretization bottleneck Q to the encoder is enabled via *straight-through estimation*, where the gradient with respect to $Q(E(x))$ received from the decoder is instead assigned to $E(x)$. To ensure the alignment of the codebook e and the encoder outputs $E(x)$, two other terms appear in the VQ-VAE objective – the codebook loss and the commitment loss:

$$\mathcal{L}_{\text{cbk}} = \big\|\text{sg}\big[Q(E(x))\big] - E(x)\big\|^2, \tag{2}$$

$$\mathcal{L}_{\text{cmt}} = \big\|Q(E(x)) - \text{sg}\big[E(x)\big]\big\|^2. \tag{3}$$

Here $\text{sg}[\cdot]$ stands for the 'stop-gradient' operator, defined as identity in the forward computation, but blocking the backpropagation of gradients. The two losses are therefore identical in value, but the first only affects (i.e. has non-zero partial derivatives w.r.t.) the codebook e (via Q), while the second only affects the encoder E. A weighting hyperparameter β is applied to $\mathcal{L}_{\text{cmt}}$ in the total loss:

$$\mathcal{L} = \mathcal{L}_{\text{ae}} + \mathcal{L}_{\text{cbk}} + \beta\mathcal{L}_{\text{cmt}} \tag{4}$$

3.2. Self-supervised learning

Self-supervised learning is a technique for learning representations of unlabeled data. The basic principle is to expose the inner structure of the data – by splitting each example into parts or by applying simple transformations to it – and then exploit this structure to define an artificial task (sometimes called the *pretext task*) to which supervised learning can be applied. Notable examples include predicting context (e.g. the neighboring words in a sentence [21] or a missing patch in an image [22]), the original orientation of a rotated image [23] or the 'arrow of time' in a (possibly reversed) video [24]. In this work, we extract pairs of excerpts from audio files and rely on them to learn a style representation as detailed in the following section.

4. METHOD

Given the goal of mapping two inputs – the content input x and the style input y – to an output, it is natural to define an encoder-decoder model with two encoders (one for each input) and a single decoder. It remains to describe how to train this model, and in particular, how to ensure the mutual disentanglement of the style and content features. Our proposal, illustrated in Fig. 1, rests on two key points:

- (i) We use a *discrete* representation $c_1, \ldots, c_L$ for content and train the model to reconstruct the content input, x; hence, the content encoder together with the decoder form a VQ-VAE. This is motivated by the success of the VQ-VAE on voice conversion as mentioned in Section 3.1.
- (ii) The output of our style encoder is a single *continuous-valued* embedding vector s. To ensure that the style encoder only encodes style (i.e. to make it content-independent), we employ a simple self-supervised learning strategy where we feed a different input y to the style encoder such that x and y are different segments of the same audio recording (with some data augmentation applied; see Section 4.1 for details).

These choices are complementary to each other, as we will now see.

Firstly, (i) necessarily means that the content encoder will drop some information from the content representation c. Since this alone does not guarantee that only content information will be preserved, (ii) is introduced to guide the encoder to do so. Our reasoning is that providing a separate style representation, not constrained by the discretization bottleneck, should make it unnecessary to also encode style information in c.

Secondly, it can be expected that in a trained model, only information useful for reconstructing x will influence the output. Hence, due to (ii) and provided that x and y do not share any content information, we expect s to only encode style. Also note that the discretization bottleneck in (i) is key for learning a useful style representation s: without it, y may be completely ignored by the model.

Once trained, the model is used for inference simply by feeding the content input and the style input to the respective encoders.

4.1. Data

Our self-supervised learning strategy consists in training on pairs of segments x, y where each such pair comes from a single recording. The underlying assumption is that such x and y have the same style (timbre) but different content. We combine data from two different sources, chosen to easily satisfy this assumption:

1. **LMD.** The 'full' version of the Lakh MIDI Dataset[5] [25] (`LMD-full`), containing 178 k MIDI files (about a year's worth of music in a symbolic representation). We pick a random non-drum part from each file, sample two 8-second segments of this part and render them as audio using a sample-based synthesizer (FluidSynth), with the SoundFont picked randomly out of 3 options.[6]

[5] `https://colinraffel.com/projects/lmd/`

[6] *Fluid R3 GM*, *TimGM6mb*, and *Arachno SoundFont*; see [26]

2. **RT.** A set of audio tracks from PG Music;[7] specifically, the 1526 RealTracks included with Band-in-a-Box UltraPAK 2018. Each RealTrack (RT) is a collection of studio recordings of a single instrument playing either an accompaniment part or a solo in a single style. We extract pairs of short segments totalling up to 20 min per RT, and clip each segment to 8 s after performing data augmentation (see below).

We perform two kinds of data augmentation. Firstly, we transpose each segment from LMD up or down by a random interval (up to 5 semitones) prior to synthesis; this ensures that the two segments in each pair have different content, but does not affect their timbre.

Secondly, we apply a set of random timbre-altering transformations to increase the diversity of the data:

- *(LMD only.)* Randomly changing the MIDI program (instrument) to a different one from the same broad family of instruments (keyboards & guitars; basses; winds & strings; ...) prior to synthesis.
- *(RT only.)* Audio resampling, resulting in joint time-stretching and transposition by up to ± 4 semitones.
- 0–4 audio effects, drawn from reverb, overdrive, phaser, and tremolo, with randomly sampled parameters.

An identical set of transformations is applied to both examples in each pair to ensure that their timbres do not depart from each other.

After this procedure, we end up with 209 k training pairs (119 k from LMD[8] and 90 k from RT).

4.2. Model and training details

We represent the audio signal as a log-scale magnitude STFT (short-time Fourier transform) spectrogram with a hop size of $1/32$ s and 1025 frequency bins. To obtain the output audio, we invert the STFT using the Griffin–Lim algorithm [27].

The model architecture is depicted in Fig. 2. The encoders treat the spectrogram as a 1D sequence with 1025 channels and process it using a series of 1D convolutional layers which serve to downsample it (i.e. reduce its temporal resolution). The last layer of the style encoder is a GRU (gated recurrent unit [28]) layer, whose final state s (a 1024-dimensional vector) is used as the style representation. This vector s is then fed to the 1st and the 4th decoder layer by concatenating it with the preceding layer's outputs at each time step.

The decoder consists of 1D transposed convolutional layers which upsample the feature sequence back to the original resolution. GRU layers are inserted to combine the content and style representations in a context-aware fashion.

We train the model using Adam [29] to minimize the VQ-VAE loss from Eq. (4), defining the reconstruction loss $\mathcal{L}_{\text{ae}}$ as the mean squared error between x and $\hat{x}$. We train for 32 epochs, taking about 20 hours in total on a Tesla V100 GPU.

5. EXPERIMENTS

As in previous music style transformation work [31, 7], we wish to evaluate our method on two criteria: (a) content preservation and (b) style fit. In timbre transfer, these should express (a) how much of the pitch content of the content input is retained in the output, and (b) how well the output fits the target timbre. To this end, we propose

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

[8] The final number is lower than the number of files in LMD due to corrupt MIDI files and parts with insufficiently many notes being discarded.

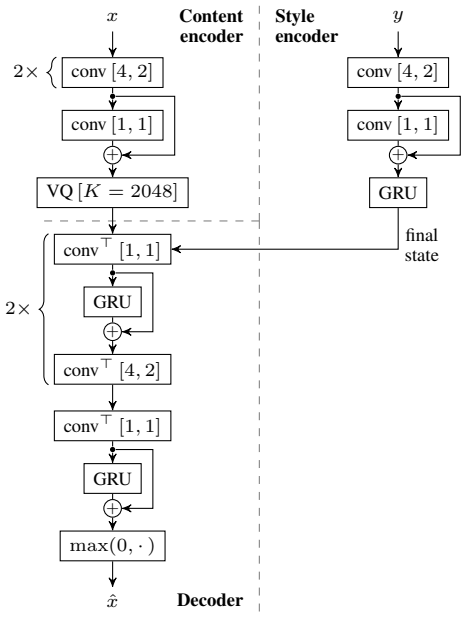

Fig. 2. The model architecture. All convolutions are 1D, with the kernel size and stride shown. All layers have 1024 channels, except for the last two (conv$^\top$ & GRU), which have 1025 (the number of frequency bins). All layers except for the input layers and the VQ are preceded by batch normalization and a Leaky ReLU activation [30]. conv$^\top$ stands for transposed convolution.

the following objective metrics for measuring pitch and timbre dissimilarity, respectively, between an output and a reference recording:

(a) **Pitch:** We extract pitch contours from both recordings using a multi-pitch version of the MELODIA algorithm [32] implemented in the Essentia library [33]. We round the pitches to the nearest semitone and express the mismatch between the two pitch sets A, B at each time step as the Jaccard distance:

$$d_J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

We report the mean value of this quantity over time.

(b) **Timbre:** Mel-frequency cepstral coefficients (MFCCs) 2–13 are generally considered to be a good approximate timbre representation [34]. Since they are computed on a per-frame basis, we train a triplet network [35] on top of them to aggregate them over time and output a single dissimilarity score. More details can be found on the supplementary website.[3]

We compare our method to 2 trivial baselines and 2 baselines from the literature:

- CP-CONTENT: Copies the content input to the output.
- CP-STYLE: Copies the style input to the output.
- *U+L*: The algorithm of Ulyanov and Lebedev [15] (not specifically designed for timbre transfer), consisting in optimizing the output spectrogram for a content loss and a style loss. We tune the ratio of the weights of the two losses on a small synthetic validation set to minimize the log-spectral distance (LSD) to the ground truth (see Section 5.1).
- *Musaicing*: A freely available implementation[9] of the musaicing algorithm of Driedger et al. [11].

[9] https://github.com/ctralie/LetItBee/

	Artificial			Real	
System	LSD_T	Timbre_T	Pitch_T	Timbre_S	Pitch_C
CP-CONTENT	14.62	0.3713	0.5365	0.4957	———
CP-STYLE	20.36	0.2681	0.8729	———	0.9099
U+L [15]	14.50	0.3483	**0.5441**	0.4792	**0.1315**
Musaicing [11]	14.51	0.2933	0.6445	0.2319	0.6297
This work	**12.16**	**0.2063**	0.5500	**0.2278**	0.6197

Table 1. Evaluation results. Distances marked S, C, and T are computed w.r.t. the style input, the content input, and the synthetic target, respectively. Results that are trivially 0 are omitted.

5.1. Artificial benchmark

First, we evaluate our method on a synthetic dataset generated from MIDI files. Although such data is not completely realistic, it enables conducting a completely objective benchmark by comparing the outputs to a synthetic ground truth.

We generate the data from the Lakh MIDI Dataset (LMD) similarly as in Section 4.1, but using a set of files held out from the training set, and with no data augmentation. We use the *Timbres Of Heaven* SoundFont (see [26]), not used for the training set.

We randomly draw 721 content–style input pairs and generate a corresponding ground-truth target for each pair by synthesizing the content input using the instrument of the style input. More details are given on the supplementary website.[3]

Both the pitch and timbre distance are measured with respect to the ground-truth target. Additionally, we measure an overall distance to the target as the root-mean-square error computed on dB-scale mel spectrograms; this is known as the *log-spectral distance* or **LSD**.

5.2. 'Real data' benchmark

We create a more realistic test set based on the 'Mixing Secrets' audio library [36], containing over 400 multi-track recordings from various (mostly popular music) genres. After filtering out multi-instrument, vocal and unpitched percussion tracks, we extract 690 content-style input pairs similarly as in Section 5.1. As no ground truth is available in this dataset, we compute the pitch and timbre metrics with respect to the content and style input, respectively.

5.3. Results

The results of both benchmarks are shown in Table 1. First, our system outperforms all baselines on LSD and the timbre metric. The difference to the CP-CONTENT baseline is negative in more than $75\,\%$ of examples on both of these metrics and in both benchmarks. Hence, viewing our system as a timbre transformation applied to the content input, we can conclude that, informally speaking, the transformation is at least partly successful in more than $75\,\%$ of cases. We may also notice that the result of CP-STYLE on timbre is, somewhat counter-intuitively, outperformed by our system. This may be a sign that the timbre metric is still somewhat influenced by pitch.

Turning to the pitch distance metric, we note that its values seem rather high (> 0.5 on a scale from 0 to 1). However, most of this error should be attributed to the pitch tracking algorithm rather than to the systems themselves. This is documented by the fact that the pitch distance of CP-CONTENT to the ground-truth target is 0.54 instead of 0. Another useful value to look at is the result of CP-STYLE: as the style input is selected randomly, its pitch distance value should be high, and is indeed close to 0.9. Using these two points of reference, we observe that our system's result is much closer to the former than to the latter in both benchmarks, which is the desired outcome. Moreover, it outperforms the musaicing baseline in both cases, albeit only slightly on real inputs.

6. DISCUSSION

Our subjective observations upon examining the outputs[3] mostly match the objective evaluation. We find that, although the sound quality of our outputs is not nearly perfect, their timbre typically does sound much closer to the style input than to the content input. (Low synthesis quality and various artifacts are somewhat expected, as they are a common occurrence with the Griffin-Lim algorithm, as well as decoders based on transposed convolutions [37]. However, synthesis quality is not the main focus of this preliminary work.)

The pitch of the content input is generally well preserved in the output, yet faster notes and polyphony seem to pose a problem. We believe this is caused by a low capacity of the discrete content representation. Even though a codebook size of 2048 seems more than sufficient in theory, we found that on both of our test sets combined, only 81 of the codebook vectors are actually used in practice. This means, for example, that at a tempo of $120\,\text{BPM}$, only 25.4 bits of information can be encoded per beat. This 'codebook collapse' [38] is a known issue with VQ-VAEs.

We also observe that our method works better on target instruments with a temporally 'stable' sound, e.g. piano; this might also explain why our method achieves better evaluation results on synthetic inputs (generated using samples) than on real ones, which are less predictable. A likely culprit is our use of a deterministic model, which cannot possibly capture the acoustic variability of instruments like saxophone or violin while being able to convert from an instrument that lacks this variability. This could be remedied by replacing our decoder with a probabilistic one which models a fully expressive conditional distribution, such as WaveNet [39].

The musaicing baseline, which uses fragments from the style input to construct the output, generally matches the target timbre very precisely, but is often less musically correct than ours. For example, note onsets tend to lack clear attacks; pitch errors and spurious notes occur, especially when the style input is non-monophonic or fast.

Finally, let us comment on the U+L baseline. Although its results on pitch are excellent, this is caused by the fact that the style weight obtained by tuning is very low (about 100 times lower than the content weight), causing the algorithm to behave much like CP-CONTENT. This is also reflected by the timbre metric. Experimenting with higher weights, we notice that the algorithm is able to transfer fragments of the style input to the output, but cannot transpose (pitch-shift) them to match the content input.

7. CONCLUSION

We have proposed a novel approach to one-shot timbre transfer, based on an extension of the VQ-VAE, along with a simple self-supervised learning strategy. Our results demonstrate that the method constitutes a viable approach to the timbre transfer task and is able to outperform baselines from the literature.

The most important shortcoming of our method seems to be the use of a deterministic decoder. We believe that a more expressive decoder such as WaveNet should allow improving the performance especially on instruments with great temporal variability, and perhaps enable extensions to more challenging style transfer tasks, such as arrangement or composition style transfer.

8. REFERENCES

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

[2] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *ICCV*, 2017.

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

[4] 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.

[5] J. Engel, L. H. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *ICLR*, 2020.

[6] A. Bitton, P. Esling, and T. Harada, "Vector-quantized timbre representation," *arXiv preprint arXiv:2007.06349*, 2020.

[7] O. Cífka, U. Şimşekli, and G. Richard, "Groove2Groove: One-shot music style transfer with supervision from synthetic data," *IEEE/ACM Trans. on Audio, Speech, and Lang. Proc.*, vol. 28, pp. 2638–2650, 2020.

[8] Z. Wang, D. Wang, Y. Zhang, and G. Xia, "Learning interpretable representation for controllable polyphonic music generation," in *ISMIR*, 2020.

[9] S. Nercessian, "Zero-shot singing voice conversion," in *ISMIR*, 2020.

[10] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *NIPS*, 2017.

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

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

[13] H. Foroughmand and G. Peeters, "Music retiler: Using NMF2D source separation for audio mosaicing," in *Audio Mostly 2018 on Sound in Immersion and Emotion (AM'18)*, 2018.

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

[15] D. Ulyanov and V. Lebedev, "Audio texture synthesis and style transfer," online (accessed Sep 29, 2020): `https://dmitryulyanov.github.io/audio-texture-synthesis-and-style-transfer/`.

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

[17] P. Esling, A. Chemla-Romeu-Santos, and A. Bitton, "Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces," in *ISMIR*, 2018.

[18] Y.-J. Luo, K. Agres, and D. Herremans, "Learning disentangled representations of timbre and pitch for musical instrument sounds using Gaussian mixture variational autoencoders," in *ISMIR*, 2019.

[19] A. Bitton, P. Esling, A. Caillon, and M. Fouilleul, "Assisted sound sample generation with musical conditioning in adversarial auto-encoders," in *Proceedings of the 22nd International Conference on Digital Audio Effects (DAFx-19)*, 2019.

[20] Y.-J. Luo, K. W. Cheuk, T. Nakano, M. Goto, and D. Herremans, "Unsupervised disentanglement of pitch and timbre for isolated musical instrument sounds," in *ISMIR*, 2020.

[21] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR*, 2013.

[22] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *CVPR*, June 2016.

[23] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *ICLR*, 2018.

[24] D. Wei, J. J. Lim, A. Zisserman, and W. T. Freeman, "Learning and using the arrow of time," in *CVPR*, June 2018.

[25] C. Raffel, *Learning-based methods for comparing sequences, with applications to audio-to-MIDI alignment and matching*, Ph.D. thesis, Columbia University, 2016.

[26] "SoundFonts and SFZ files," in *Handbook for MuseScore 3*. MuseScore, online (accessed Sep 26, 2020): `https://musescore.org/en/handbook/soundfonts-and-sfz-files`.

[27] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Trans. on Acoustics, Speech, and Signal Proc.*, vol. 32, no. 2, pp. 236–243, 1984.

[28] 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.

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

[30] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML*, 2013.

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

[32] J. Salamon and E. Gómez, "Melody extraction from polyphonic music signals using pitch contour characteristics," *IEEE Trans. on Audio, Speech, and Lang. Proc.*, vol. 20, no. 6, pp. 1759–1770, 2012.

[33] D. Bogdanov, N. Wack, E. Gómez Gutiérrez, S. Gulati, H. Boyer, O. Mayor, G. Roma Trepat, J. Salamon, J. R. Zapata González, X. Serra, et al., "Essentia: An audio analysis library for music information retrieval," in *ISMIR*, 2013.

[34] G. Richard, S. Sundaram, and S. Narayanan, "An overview on perceptually motivated audio indexing and classification," *Proceedings of the IEEE*, vol. 101, no. 9, pp. 1939–1954, 2013.

[35] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*. Springer, 2015, pp. 84–92.

[36] "The 'Mixing Secrets' free multitrack download library," online (accessed Sep 25, 2020): `https://www.cambridge-mt.com/ms/mtk/`.

[37] J. Pons, S. Pascual, G. Cengarle, and J. Serrà, "Upsampling artifacts in neural audio synthesis," in *ICASSP*, 2021.

[38] S. Dieleman, A. van den Oord, and K. Simonyan, "The challenge of realistic music generation: modelling raw audio at scale," in *NeurIPS*, 2018.

[39] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," in *SSW*, 2016.