Deep learning methods for music style transfer

Ondřej Cífka

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2021IPPAT029

Thèse de doctorat

Deep learning methods for music style transfer

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris

École doctorale n°626
École Doctorale de l'Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat: Signal, Images, Automatique et robotique

Thèse présentée et soutenue à Palaiseau, le 17/11/2021, par

ONDŘEJ CÍFKA

Composition du Jury :

Geoffroy Peeters Professeur, Télécom Paris (IDS, LTCI), IP Paris, France	Président
Gerhard Widmer Professor, JKU Linz (ICP), Autriche	Rapporteur
Jean-Pierre Briot Directeur de recherche, CNRS (LIP6), Sorbonne Univ., France	Rapporteur
Rachel Bittner Senior Research Scientist, Spotify, France	Examinatrice
Antoine Liutkus Chargé de recherche, INRIA (LIRMM), Univ. Montpellier, France	Examinateur
Gaël Richard Professeur, Télécom Paris (IDS, LTCI), IP Paris, France	Directeur de thèse
Umut Şimşekli Chargé de recherche, INRIA (SIERRA), ENS (DI), France	Co-directeur de thèse

Deep learning methods for music style transfer

Ondřej Cífka

Abstract

In music, composers, arrangers, performers and producers often adapt existing pieces to different contexts and audiences. Recently, deep learning methods have enabled transforming musical material in a data-driven manner, setting the ground for tools which could partially automate this process. The research performed in this area so far has focused largely on conversion between a small set of musical genres or instrument timbres, and on tasks that involve completing a partial arrangement in a desired style. The focus of this thesis, on the other hand, is on a family of tasks which we refer to as *(one-shot) music style transfer*, where the goal is to transfer the style of one musical piece or fragment onto another. We propose two specific tasks in this direction: (1) *accompaniment* style transfer for symbolic music representations (i.e. digital scores or MIDI files), and (2) *timbre* transfer for audio recordings. For each of these tasks, we propose novel methods based on deep learning, as well as evaluation protocols. Additionally, we present a broader contribution related to the processing of sequences (music or otherwise) using *Transformer* neural networks.

In the first part of this work, we focus on supervised methods for *symbolic music accompaniment style transfer*, aiming to transform a given piece by generating a new accompaniment for it in the style of another piece. The method we have developed is based on supervised sequence-to-sequence learning using recurrent neural networks (RNNs) and leverages a synthetic *parallel* (pairwise aligned) dataset generated for this purpose using existing accompaniment generation software. We propose a set of objective metrics to evaluate the performance on this new task and we show that the proposed system is successful in generating an accompaniment in the desired style while following the harmonic structure of the input. We also present additional analyses aimed at a better understanding of the system.

In the second part, we investigate a more basic question: the role of *positional encodings* (*PE*) in music generation using *Transformers*. In particular, we propose *stochastic positional encoding* (*SPE*), a novel form of PE capturing *relative positions* while being compatible with a recently proposed family of efficient Transformers. The main theoretical contribution of this work is to draw a connection between positional encoding and cross-covariances of correlated stochastic processes. We demonstrate that SPE allows for better extrapolation beyond the training sequence length than the commonly used *absolute PE*. We follow up on this work with an experiment studying how PE can be better exploited for music generation by making it encode more musically meaningful information.

Finally, in the third part, we turn from symbolic music to audio and address the problem of *timbre transfer*. Specifically, we are interested in transferring the timbre of an audio recording of a single (but not necessarily monophonic) musical instrument onto another such recording while

preserving the pitch content of the latter. 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. As in the first part, we design a set of objective metrics for the task. We show that the proposed method is able to outperform existing ones.

We believe that our contributions open interesting directions for follow-up work. Firstly, our approach to timbre transfer is promising, but may benefit from more advanced audio synthesis techniques to improve the sound quality of the outputs. We are also interested in investigating whether the approach could be adapted to symbolic music by combining it with efficient Transformers; this could lead to a more robust system for accompaniment or arrangement style transfer. Finally, regarding positional encodings in Transformers, we see a need for a more careful investigation of their role not only in music generation, but in sequence generation in general.

Méthodes d'apprentissage profond pour le transfert de style musical

Ondřej Cífka

Résumé

Les compositeurs, les arrangeurs, les interprètes et les producteurs de musique adaptent souvent des morceaux existants à des contextes et publics différents. Récemment, les méthodes d'apprentissage profond ont permis d'effectuer des transformations du matériel musical basées sur les données (data-driven), ce qui pourrait aider à créer des outils permettant d'automatiser une partie de ce processus. Les travaux antérieurs dans ce domaine se sont concentrés principalement sur la conversion entre un petit nombre de genres musicaux ou de timbres. L'objet de cette thèse est d'étendre ce cas de figure et de considérer le problème plus général du *transfert de style musical*, dont le but est de transférer de manière automatique le style d'un morceau à un autre. Nous proposons deux tâches différentes dans ce sens : (1) le transfert de style des *accompagnements* dans une représentation *symbolique* (c'est-à-dire sous forme d'une partition numérique ou un fichier MIDI), et (2) le *transfert de timbre* des enregistrements audio. Pour chacune de ces tâches, nous proposons une approche basée sur l'apprentissage profond, ainsi qu'un protocole d'évaluation. Nous apportons également une contribution plus large liée au traitement de séquences (musicales ou autres) à l'aide de réseaux de neurones appelés les *Transformers*.

Dans la première partie de la thèse, nous nous concentrons sur les méthodes supervisées pour le transfert de style des *accompagnements* dans une représentation symbolique. Plus précisément, l'objectif de ce travail est de transformer un morceau en lui générant un nouvel accompagnement dans le style d'un morceau différent. La méthode proposée est basée sur l'apprentissage supervisé de séquence à séquence à l'aide de réseaux de neurones récurrents (RNN), une technique développée pour la traduction automatique. Le système est entraîné sur une base de données synthétiques *parallèle* (alignée par paires) générée à cet effet à l'aide d'un logiciel existant de génération d'accompagnement. Nous proposons ainsi un ensemble de mesures objectives pour évaluer la performance sur cette nouvelle tâche et nous montrons que le système proposé réussit à générer un accompagnement dans le style souhaité (pas forcément connu pendant l'entraînement) tout en suivant la structure harmonique de l'entrée. En plus, nous présentons des analyses supplémentaires visant à mieux comprendre le fonctionnement du système proposé.

Dans la deuxième partie, nous étudions une question plus fondamentale : le rôle des *encodages positionnels* dans la génération de musique à l'aide des *Transformers*. Nous proposons *l'encodage positionnel stochastique* (*SPE*), un nouvel encodage positionnel capable de coder des *positions relatives* et compatible avec une classe récemment proposée de Transformers efficaces (*Transformers à complexité linéaire*). La principale contribution théorique de ce travail est l'établissement d'un lien entre l'encodage positionnel et la covariance croisée de processus gaussiens corrélés. Nous

montrons expérimentalement que le SPE permet, mieux que la méthode conventionnelle (*l'encodage positionnel absolu*), de modéliser des séquences plus longues que celles rencontrées pendant l'entraînement. Nous poursuivons ce travail avec une expérience étudiant comment l'encodage positionnel peut être mieux exploité pour la génération de musique en le faisant coder des informations plus significatives musicalement.

Enfin, dans la troisième partie, nous passons de la musique symbolique à l'audio et abordons le problème du *transfert de timbre*. Plus précisément, étant donnés deux enregistrements audio, chacun d'un seul instrument (mais pas forcément monophonique), nous cherchons à transférer le timbre de l'un à l'autre, tout en préservant le contenu mélodique et harmonique du dernier. Nous présentons une nouvelle méthode pour cette tâche, basée sur une extension de *l'autoencodeur variationnel quantifié* (*VQ-VAE*), ainsi qu'une stratégie *d'apprentissage auto-supervisé*. La méthode est conçue pour obtenir des représentations séparées (démêlées) du timbre et de la hauteur, ce qui permet d'effectuer le transfert de timbre. Comme dans la première partie, nous concevons un ensemble de métriques objectives pour la tâche. Nous montrons que la méthode proposée est capable de surpasser des méthodes existantes.

Notre travail ouvre des pistes intéressantes pour le futur. D'abord, notre approche au transfert de timbre est prometteuse, mais bénéficierait d'une méthode plus avancée de synthèse sonore afin d'améliorer la qualité des résultats. Nous envisageons aussi la possibilité d'adapter cette approche à la musique symbolique en la combinant avec les Transformers efficaces ; cela permettrait d'obtenir un système plus robuste de transfert de style d'accompagnement ou d'arrangement. Enfin, concernant les encodages positionnels, nous voyons un besoin d'étudier davantage leur rôle non seulement dans la génération de musique, mais plus généralement dans la génération de séquences.

Acknowledgments

I would like to express my gratitude to:

- my supervisors Gaël and Umut for their guidance, support and kindness, for always being there when I needed them – as well as knowing when to leave me alone – and in general for making my PhD an enjoyable experience;
- colleagues from Télécom Paris and the ADASP group – including both of my supervisors – not only for creating a pleasant workplace environment, but also for the fun we've had playing music, even virtually[1] during the COVID-19 lockdown;
- my love Kateřina for her companionship, in particular for agreeing to move to Paris, a decision she continues to lament to this day;
- Giorgia, Javier, Karim and Kilian for the mutual support in the face of the French bureaucracy;
- Alexey for hosting (unfortunately only remotely) my internship at InterDigital and for his adamance in pushing our paper and code through the company's internal approval process;
- Antoine for the fruitful and stimulating collaborations and for being an endless source of ideas;
- Eric (Yi-Hsuan) and Shih-Lun for their valuable advice and contributions;
- Marie Skłodowska-Curie Actions and the MIP-Frontiers network[2] for making my PhD possible:

> This work was supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 765 068 (MIP-Frontiers).

[1] https://youtu.be/hFWqzEQw4Ow
[2] https://mip-frontiers.eu/

Contents

List of publications

⟨**ISMIR2019**⟩
Ondřej Cífka, Umut Şimşekli, and Gaël Richard. Supervised symbolic music style translation using synthetic data. In *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR 2019)*, pages 588–595, Delft, The Netherlands, November 2019. ISMIR. doi: 10.5281/zenodo.3527878. URL `https://doi.org/10.5281/zenodo.3527878`.

⟨**TASLP2020**⟩
Ondřej Cífka, Umut Şimşekli, and Gaël Richard. Groove2Groove: One-shot music style transfer with supervision from synthetic data. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 28:2638–2650, August 2020. IEEE. doi: 10.1109/TASLP.2020.3019642. URL `https://hal.archives-ouvertes.fr/hal-02923548`.

⟨**ICASSP2021**⟩
Ondřej Cífka, Alexey Ozerov, Umut Şimşekli, and Gaël Richard. Self-supervised VQ-VAE for one-shot music style transfer. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2021)*, Toronto, Canada, June 2021. IEEE. doi: 10.1109/ICASSP39728.2021.9414235. URL `https://hal.telecom-paris.fr/hal-03132940`.

⟨**ICML2021**⟩
Antoine Liutkus,* Ondřej Cífka,* Shih-Lun Wu, Umut Şimşekli, Yi-Hsuan Yang, and Gaël Richard. Relative positional encoding for Transformers with linear complexity. In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*, July 2021. PMLR. URL `http://proceedings.mlr.press/v139/liutkus21a.html`.

*Equal contribution.

Notation

$\mathbf{v} = [v_n]_n$	column vector with entries v_n
$\mathbf{A} = [a_{mn}]_{mn}$	matrix with entries a_{mn}
$\mathbf{a}_m = [a_{mn}]_n$	m-th row of matrix $\mathbf{A}$
$\mathbf{a}_{*n} = [a_{mn}]_m$	n-th column of matrix $\mathbf{A}$
x	scalar or unspecified object
$[\mathbf{u}, \mathbf{v}]$	concatenation of vectors $\mathbf{u}, \mathbf{v}$
$[a\ b]$	closed interval from a to b
$(a\ b)$	open interval from a to b
$(v_1, v_2, \ldots, v_N)$	tuple or row vector
$\mathbf{0}_N$ or $\mathbf{0}$	zero vector $(\underbrace{0, 0, \ldots, 0}_{N})^\top$
$\mathbf{1}_N$ or $\mathbf{1}$	all-ones vector $(\underbrace{1, 1, \ldots, 1}_{N})^\top$
$\mathbb{1}_n$	one-hot vector $(\underbrace{0, 0, \ldots, 0}_{n-1}, 1, 0, 0, \ldots)^\top$
$\mathrm{diag}(\mathbf{v})$	diagonal matrix with the entries of $\mathbf{v}$ on the diagonal

1. Introduction

In music, composers, arrangers, performers and producers often adapt existing pieces to different contexts and audiences. For instance, music of the Renaissance period was often based on popular melodies, such as *L'homme armé*, a Medieval secular tune that gave rise to dozens of masses. Mussorgsky's *Pictures at an Exhibition* exists in hundreds of different arrangements for various instrumentations, with Ravel's orchestration being known to a wider audience than the original piano version. In modern popular music, cover songs are a common phenomenon; if sufficiently different in style from the original, a cover version may make the song accessible to a different audience, like Jimi Hendrix's classic rendition of *All Along the Watchtower* by Bob Dylan. More broadly, any performance of a musical piece can be viewed as the artist's personal interpretation of the piece, adapted to some extent to the context in which it is performed.

With technological advances of the late 20$^{\text{th}}$ century, it became possible to combine and transform existing recordings, leading to practices such as *remixing* and *sampling*, which became especially widespread with the advent of digital audio. A common example are dance remixes, altering the sound of a song to make it better suited for the dance floor, mainly by adding or replacing tracks or applying audio effects. Recordings may also be edited (shortened) to make them suitable for radio broadcasting or a TV advertisement.

More recently, developments in algorithms, signal processing and machine learning have brought the promise of directly manipulating more high-level, musically meaningful features. For example, it has become possible to automatically harmonize melodies [Ebcioğlu, 1990, Simon et al., 2008, Huang et al., 2017], generate sophisticated accompaniments or 'improvised' solos (see e.g. Band-in-a-Box[1]), turn scores into expressive performances [Dixon et al., 2005, Widmer et al., 2009] or transfer sound textures from one recording onto another [Driedger et al., 2015, Grinstein et al., 2018].

One area where such automated music transformations can be applied is the very act of music making. In a practice called *human-AI co-creation*, artists[2] experiment with integrating machine learning models into their creative process, using them to transform or complete existing musical fragments or generate new musical ideas from scratch [Roberts, 2019, Huang et al., 2020]. Neural audio synthesis methods [Engel et al., 2017, 2020] enable artists to explore new timbres and, when combined with pitch estimation, to employ them as virtual instruments controlled by audio input.[3]

Another application lies in generating music for short videos, such as advertisements or tutorials. Finding music that matches a video may be difficult and time-consuming and is complicated by copyright issues [Frid et al., 2020]. This has inspired services[4] that offer 'AI-generated music' based on different user-specified parameters. There has also been work on the retrieval of music based on video content [Prétet et al., 2021]. An interesting opportunity for innovation then lies

[1] `https://www.pgmusic.com/`

[2] e.g. `https://yacht.bandcamp.com/album/chain-tripping`

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

[4] e.g. `https://www.aiva.ai/`, `https://www.ampermusic.com/`

in automatically generating or transforming music in order to match the content of a video or mimic the style of a given song.

A particularly interesting situation arises in video game soundtracks, which need to be sufficiently varied to avoid tiring the player during hours of gameplay. Moreover, the soundtrack needs to change dynamically in response to the player's actions and the state of the environment. Consequently, video games lend themselves to *procedural* music [Collins, 2009] that adapts in real time to various in-game variables, typically by using pre-composed building blocks that are recombined in a randomized way, controlled by game-dependent logic. Such soundtracks require large amounts of specialized engineering effort, creating an exciting opportunity for methods capable of automating this process, e.g. to adapt existing music to a gameplay context.

1.1 Music style conversion

After a broad introduction to context-based music transformations, let us focus on a specific kind of transformation which we call **music style conversion** and which is the main topic of this thesis. In general terms, given a piece x with *content* C and *style* S, the aim of style conversion is to produce a piece y with the same content C, but a different style T (the *target style*). For this definition to be useful, we must also define what we mean by style and content. However, as we are about to see, there is no universal definition of these terms that could cover all possible use cases, and their meaning will instead depend on the concrete type of music style conversion task at hand.

Variety of style conversion tasks. Conceivable examples of music style conversion include:

(a) Changing the instrumentation of a given recording.

(b) Changing the lead vocal track to resemble a different singer.

(c) Changing the dynamics, timing or articulation to alter the mood of a performance or to imitate a given performer.

(d) Generating a cover of a song in the style of another song or artist, or in a given genre.

It should be apparent from these examples that style conversion can happen on a number of different levels. Xia and Dai [2018] define three such levels: *timbre*, *performance control* and *composition style*. For example, in (a) and (b), we modify timbre while preserving the information on the other two levels; in (c), we only modify performance control. However, in (d), the style features we wish to modify may be spread across all three levels.

Style transfer and style translation. We may also categorize style conversion tasks based on how the target style T is specified to the system. In this work, we consider two basic options:

(i) The set of possible styles is finite (and typically small) and fixed in advance. The target style T may be represented as a discrete label (ID) or may not have an explicit representation (e.g. if a separate (sub-)system is trained for each target style). We refer to this case as **style translation**.

(ii) The set of possible styles is potentially infinite. The target style T is represented by a single *example*, i.e. by a musical fragment z in that style. This may be understood as **style transfer** [Gatys et al., 2016], and borrowing its terminology, we can call x the *content input* (since it bears the content C) and z the *style input* (since it bears the target style T).

As we will detail in Chapter 2, most prior work on music has focused on style translation (i). Note that although some of these prior works also use the expression 'style transfer', this conflicts with how the term is traditionally understood [Efros and Freeman, 2001, Xie et al., 2007, Gatys et al., 2016], and the term 'translation' [Isola et al., 2017, Zhu et al., 2017, Malik and Ek, 2017, Mor et al., 2019] is in our opinion more appropriate and helps us draw the distinction between (i) and (ii).

To further highlight this distinction, we will occasionally refer to (ii) as ***one-shot* style transfer**. This is by analogy to *one-shot learning*, the problem of learning the concept of a class from a single example (in order to perform classification [Li et al., 2006] or to generate new samples from the class [Rezende et al., 2016]). In our case, we use the term 'one-shot' to emphasize that the system must extract the (potentially previously unseen) style T from a single example z.

Tasks of interest. We are now ready to define the tasks that are the focus of this thesis. The first one is **accompaniment style transfer**, a task where the inputs and outputs are popular music or jazz accompaniments in a symbolic representation (in our case, MIDI[5] files). We define the content of an accompaniment as the harmonic structure of the song – i.e. the information represented in a chord chart – and style refers to the way musicians produce an actual accompaniment based on this information. Note that we do not consider styles to be broad classes such as *genres* – instead, we consider a style to represent the set of patterns (e.g. riffs, voicings, rhythmical patterns) characteristic of a given artist or even an individual song. The task is illustrated in Fig. 1.1.

Ultimately, accompaniment style transfer enables creating a cover version of a given song by generating a new accompaniment for it in a given style. If sufficiently reliable, a system with this capability could be applied to aforementioned game or video soundtracks in order to increase the stylistic variability of existing material or even for personalization (adapting music to the listener's taste). In any case, accompaniment style transfer systems could be used by music creators to quickly try out different musical styles or even to create remixes or mashups in a human-AI co-creation setup.

The second task of interest is **audio timbre transfer**, illustrated in Fig. 1.2, where the inputs are single-instrument (but not necessarily monophonic) audio recordings. In this case, content is defined as pitch and style is defined as timbre. Again, we consider timbre in a very broad sense, including not only the identity of the instrument, but also any audio effects applied on top of it.

[5] `https://www.midi.org`

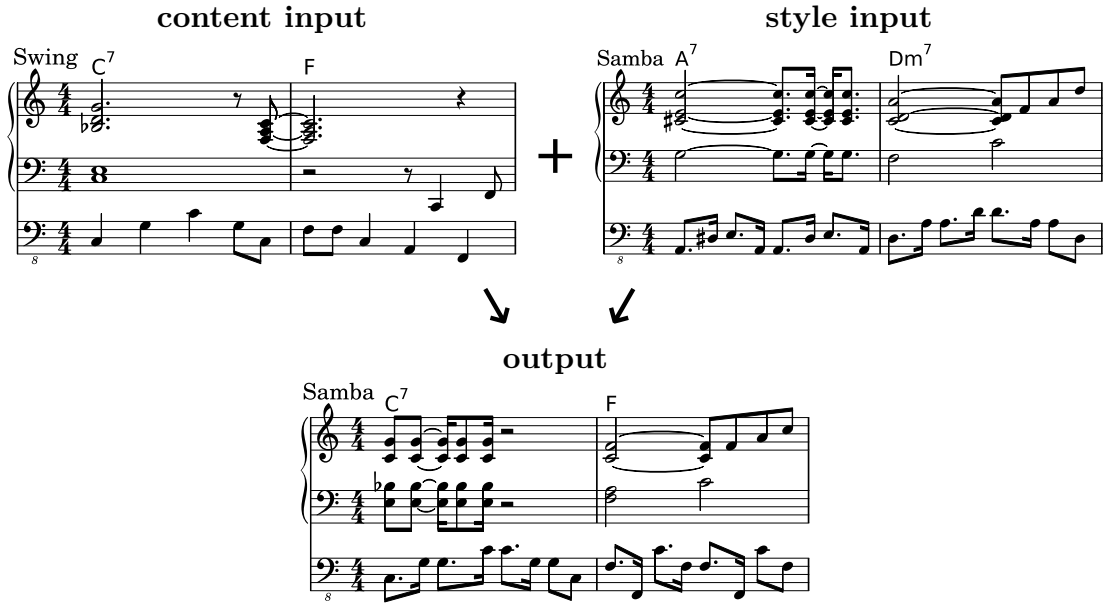


Figure 1.1 – An example of accompaniment style transfer. The accompaniments in this example consist of a piano track and a bass track. The output follows the harmony of the content input (the chords C^7 and F) while employing the samba-like rhythmic and melodic patterns of the style input. In general, the two inputs need not be the same length. The chord symbols are shown for illustration only and are not part of the data for the task.

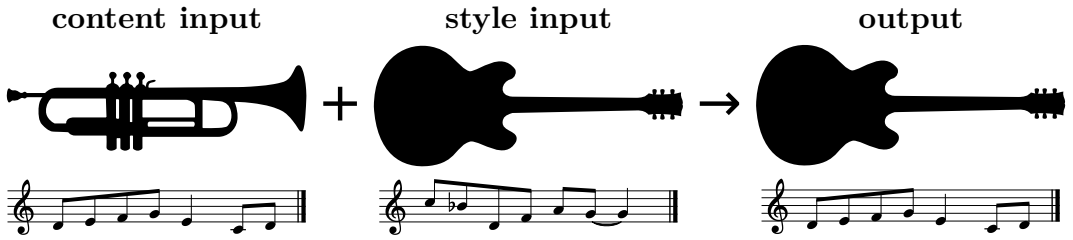


Figure 1.2 – An example of timbre transfer. The inputs and outputs are audio recordings. The output combines the pitch content of the content input with the timbre of the style input.

This task has obvious applications in music making. In particular, a real-time, high-fidelity timbre transfer system would allow to turn a short audio sample into a fully realistic virtual instrument controllable e.g. by singing or playing an acoustic instrument.

1.2 Structure and contributions

The rest of the thesis is structured in the following manner:

- In Chapter 2, we review **related work** on music style conversion.
- In Chapter 3, we present some background on **deep learning techniques for sequence generation**, which are used extensively in this thesis.
- Chapter 4 concerns **supervised methods for accompaniment style**

conversion, in particular one-shot accompaniment style transfer and, as an intermediate, simpler task, accompaniment style translation. The chapter includes the following major contributions of this thesis:

  - We propose a common framework for accompaniment style conversion tasks based on *supervised learning from synthetic data*.
  - We design a common evaluation protocol for the tasks, consisting of objective metrics of *content preservation* and *style fit*.
  - As a first step, we approach the **accompaniment style translation** task and develop a method capable of translating bass-and-piano accompaniments between 70 different styles.
  - We then extend the method to the **one-shot accompaniment style transfer** setting while also proposing a more robust input/output representation and an efficient strategy for working with accompaniments consisting of arbitrary combinations of instruments.
  - Our experimental results demonstrate the performance of the proposed systems on the respective tasks. We also present additional analyses such as an ablation study and visualizations of the learned style representations.

  These contributions have been published in the papers ⟨**ISMIR2019**⟩ and ⟨**TASLP2020**⟩ (see the List of publications).

- In Chapter 5, we investigate a more basic question: the role of **positional encodings** in symbolic music generation using **Transformers**. The major contributions presented here are as follows:

  - We propose *stochastic positional encoding* (*SPE*), a novel form of positional encoding capturing *relative positions* (considered important in music generation) while being compatible with recently proposed *linear complexity Transformers*.
  - We demonstrate that when applied to music generation, SPE allows for better extrapolation beyond the training sequence length than the commonly used *absolute positional encoding* (*APE*).
  - We follow up on this work with an experiment studying how positional encodings can be better exploited for music generation by making them encode more musically meaningful information.

  The first two of these contributions have been published in ⟨**ICML2021**⟩, along with some results on non-music data, additionally presented in Appendix A.2.

- In Chapter 6, we turn from symbolic music to audio and present our contributions addressing the problem of **timbre transfer**:

  - We propose 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 design an evaluation protocol for the task, consisting of objective metrics of *content preservation* and *style fit*.
  - Our experimental results show that the proposed method is able to outperform baselines from the literature.

  These contributions have led to the publication ⟨**ICASSP2021**⟩.

- Chapter 7 concludes the thesis, summarizes its contributions and offers some directions for future research.

The above contributions are accompanied by websites with listening examples and/or interactive demos, as well as source code and other resources. Their list can be found in Appendix C.

Moreover, three open-source software packages, HTML MIDI Player, NoPdb and Confugue, described in Appendix D, have been released as a by-product of this work.

2. Related work

This chapter will discuss prior work closely related to the style transfer task. In general, we are interested in learning style from data, and we therefore mostly omit methods that are rule-based and/or tailored to specific styles. We also focus more on deep learning-based approaches, as they are the topic of this thesis.

We open in Section 2.1 with some elementary background on deep learning for music and audio processing. In Section 2.2, we review methods that are specific to audio data (especially as opposed to symbolic music representations) and usually rely on theoretical understanding of audio signals and traditional digital signal processing methods. On the other hand, Sections 2.3 and 2.4 give an account of methods that are more strongly data-driven and are applicable across data modalities. We leave various other (more loosely related) works for Section 2.5. Section 2.6 concludes the chapter with some final remarks.

2.1 Deep learning preliminaries

In this section, we briefly introduce some deep learning concepts which will appear in the rest of this chapter. For a more complete overview of deep learning techniques for audio and music processing, see Peeters and Richard [2021].

Convolutional neural networks (CNN). CNNs have become widespread as a means to process data presented as a regular grid, e.g. images or regularly sampled time series. Their core building block is a *convolutional layer*, which convolves its input with a set of learnable *filters*, essentially applying the same linear operation to different 'patches' of the input.

In the case of images, the input to each layer is usually a 3D tensor with 2 spatial dimensions (width, height) and 1 channel dimension; in this case, the filters of each layer make up a 4D tensor (*kernel*) with dimensions corresponding to width, height, input channels and output channels (i.e. number of filters); 2D convolution along the 2 spatial dimensions is performed. When processing audio, the input to the network is usually a *spectrogram*, i.e. a representation with a time dimension and a frequency dimension. One option is to treat the spectrogram as an image with a single channel and use 2D convolutions. However, depending on the type of spectrogram and the task, it may be more appropriate to use 1D convolution along the temporal dimension only.

A CNN then consists of a series of multiple such convolutional layers, interlaced with non-linearitites (activation functions) and pooling operations, which downsample (i.e. reduce the spatial dimensions of) the features. A popular pooling operation is max-pooling, which takes the maximum value in each spatial region. An alternative to pooling is strided convolution, where the convolutional filter skips some locations as it slides over the input, also leading to downsampling.

Recurrent neural networks (RNN). An RNN is a suitable architecture whenever the input is a sequence of feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$. It works by

processing the input sequence from left to right and using each input $\mathbf{x}_n$ to update its internal ('hidden') state vector. The resulting sequence of state vectors $\mathbf{h}_1, \ldots, \mathbf{h}_N$ can then be used for further processing. For example, if we wish to classify the input sequence, we can use the last hidden state $\mathbf{h}_N$ as input to a linear classifier. We can also use the whole sequence of hidden states to predict a value for each position.

We will describe RNNs in more detail in Chapter 3, which will also explain how RNNs can be used for sequence generation.

Autoencoders. An autoencoder is any neural network that is trained to *reconstruct* its input, i.e. to copy its input to its output. It consists of an *encoder*, which maps the input x to a lower-dimensional intermediate representation z, often called the *latent code*, and a *decoder*, which then maps the latent code z to a reconstruction $\hat{x}$. The utility of an autoencoder lies precisely in its latent code, which may be useful as features for a downstream task. In some cases, it is also possible to use the autoencoder to generate new data samples or to alter inputs by manipulating the corresponding latent codes.

Different types of autoencoders exist, imposing different constraints on the latent code space in order to obtain some desirable properties. Arguably the most popular kind is the *variational autoencoder* (*VAE*, Kingma and Welling [2014]), which assumes a particular *prior distribution* $p(z)$ over the latent codes, typically the standard Gaussian.

2.2 Audio-specific methods

We start our literature review with a method which was in fact originally proposed for images and only later adapted to audio, but is still somewhat domain-specific. The method in question was introduced by Gatys et al. [2016] and constitutes the first neural approach to **style transfer**. It is based on the observation that in a CNN trained for image recognition, the deeper layers encode high-level information about the composition of the image (*content*), while information about textures present in the image (*style*) is captured by summary statistics of different layers.

The method is illustrated in Fig. 2.1. The input to the algorithm are two images, the content image I_c and the style image I_s, which are then used to extract content and style information, respectively, in the following way:

(a) The content image I_c is processed by the CNN and the activations $\mathcal{F}^{l_c}(I_c)$ of one of the deeper (higher) layers l_c are used as the content representation.

(b) The style image I_s is processed by the same CNN and the Gram matrix of each layer of the network is computed. This Gram matrix $\mathcal{G}^l(I_s)$ of the layer l captures the correlations between the activations in the different feature maps at this layer. The Gram matrices $\mathcal{G}^1(I_s), \ldots, \mathcal{G}^L(I_s)$ of all the layers are used together as a style representation.

Once the content representation and the style representation are extracted, we wish to find an image I that matches both simultaneously, i.e. one that minimizes a weighted sum of the following losses (Euclidean distances to the target

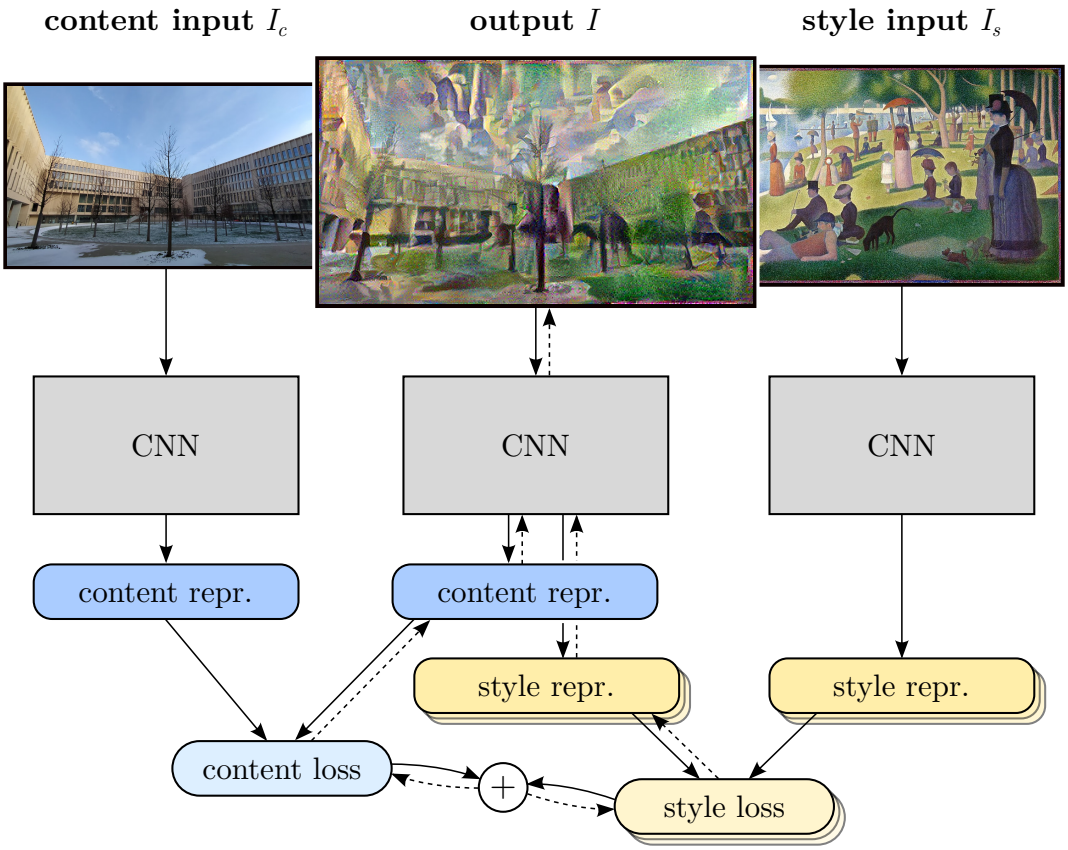


Figure 2.1 – Image style transfer. **Left:** the content representation of the image I_c is obtained as the features produced by a specific layer of the pre-trained CNN. **Right:** the style representation of the image I_s is computed as the Gram matrices of the features produced by different layers of the same CNN. **Middle:** the content and style features of the (initially random) image I are computed analogously. The total loss (weighted sum of distances between representations) is calculated and its gradient is propagated back to I.

representations):

$$\begin{aligned} \left\| \mathcal{F}^{l_c}(I_c) - \mathcal{F}^{l_c}(I) \right\|^2 & \qquad \text{the content loss,} \\ \left\| \mathcal{G}^{l}(I_s) - \mathcal{G}^{l}(I) \right\|^2 & \qquad \text{the style loss at layer } l \ (\forall l). \end{aligned}$$

This is done by initializing the image with random noise and then optimizing it using gradient descent. In each step, the representations $\mathcal{F}^{l_c}(I)$, $\mathcal{G}^{l}(I)$ are computed with the use of the pre-trained CNN and the gradient is back-propagated to update the image.

The technique has subsequently been improved and extended, as well as adapted to specific kinds of style transfer. For a taxonomy of image and video style transfer algorithms, see Jing et al. [2020].

Ulyanov and Lebedev [2016] and Grinstein et al. [2018] adapted and generalized this framework in order to transfer '**sound textures**'. As a proof-of-concept, Ulyanov and Lebedev simply replace the image with the spectrogram of an audio recording and use an *untrained* (randomly initialized) single-layer one-dimensional CNN to extract content and style features. On the other hand,

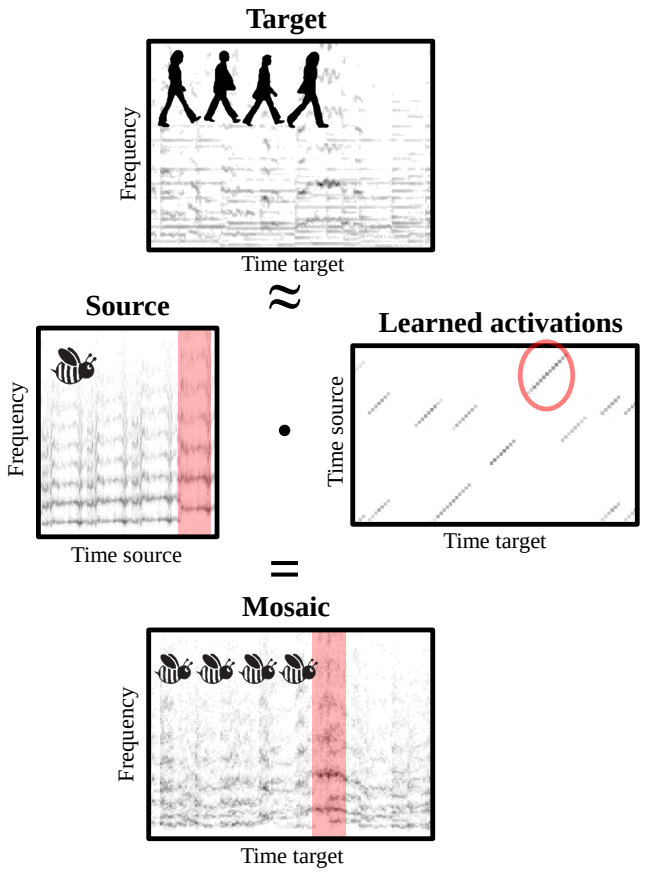


Figure 2.2 – The NMF-driven musaicing method (a.k.a. 'Let It Bee') of Driedger et al. [2015]. The 'source' (the buzzing of a bee) and the 'target' (*Let It Be* by The Beatles) correspond to our style input and content input, respectively. Reproduced from Driedger et al. [2015]. © 2015 Jonathan Driedger, Thomas Prätzlich, Meinard Müller.

Grinstein et al. drop the content loss and instead use the content input for initialization, letting the optimization converge to a nearby local optimum. They also propose an alternative way to compute the style loss, based on a model emulating the human auditory system. While both works achieve interesting results, the approach is not specifically tailored to music and does not aim to transfer musical *timbre*.

On the other hand, a method designed for music, and one that can be considered to perform music ***timbre* transfer**, is *musical mosaicing* or *musaicing* [Zils and Pachet, 2001]. Musaicing is a form of *concatenative synthesis* which splits a 'source' audio recording into short frames (each a fraction of a second in duration) and concatenates them so as to match the characteristics of another ('target') recording.

Driedger et al. [2015] propose to combine this method with *non-negative matrix factorization* (*NMF*), as illustrated in Fig. 2.2. The frames from the source recording are used as a fixed *dictionary* W of spectral templates, and an *activation matrix* H is then obtained using an iterative update process so that the product WH approximates the spectrogram of the target recording. The Griffin-Lim algorithm [Griffin and Lim, 1983] is then applied to this approximation WH

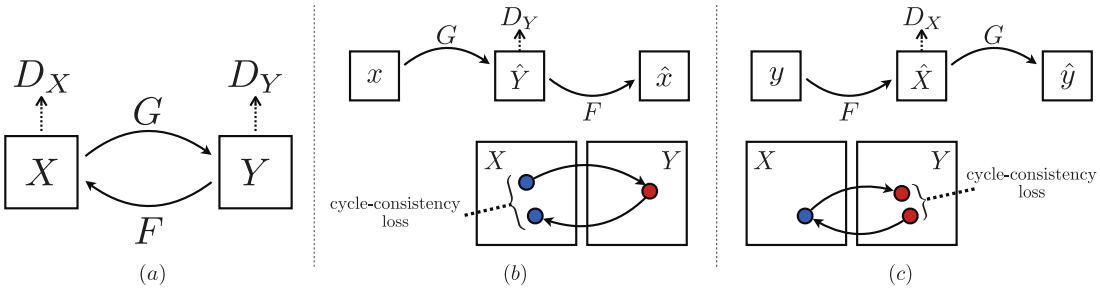


Figure 2.3 – Cycle-consistent generative adversarial network (CycleGAN). (a) The model contains two mappings $G\colon \mathcal{X} \to \mathcal{Y}$ and $F\colon \mathcal{Y} \to \mathcal{X}$. The two corresponding discriminators $D_{\mathcal{X}}$ and $D_{\mathcal{Y}}$ encourage the outputs of the two mappings to be indistinguishable from samples from the target domains $\mathcal{X}$ and $\mathcal{Y}$, respectively. (b) The forward cycle consistency loss, enforcing $F(G(x)) \approx x$. (c) The backward cycle consistency loss, enforcing $G(F(y)) \approx y$. Reproduced from Zhu et al. [2017]. © 2017 IEEE.

in order to obtain a time-domain signal, which is the final output of the algorithm. Compared to the original NMF algorithm, Driedger et al. propose an extended set of update rules that promotes sparsity and diagonal structures in the activation matrix, which leads to better timbre preservation.

Subsequent works [Aarabi and Peeters, 2018, Tralie, 2018] instead use *non-negative matrix factor 2D deconvolution* (*NMF2D*, Schmidt and Mørup [2006]) in an attempt to improve the performance in cases where the source audio is not monophonic.

In a radically different, learning-based approach, Engel et al. [2020] perform music **timbre conversion** using a trainable synthesis model conditioned on pitch and loudness. The model is composed of a neural network that controls the parameters of a *harmonic-plus-noise synthesizer* (combining additive and subtractive synthesis) with explicitly controlled pitch (fundamental frequency). Timbre conversion is achieved by extracting the pitch and loudness information from a given recording (the former using a pre-trained pitch estimator) and feeding it to the synthesis model, trained for a specific target instrument.

Similarly, Bitton et al. [2021] combine a neural network with a subtractive synthesizer, but their model is trained end-to-end as a waveform autoencoder. The latent representation is *discrete* (see VQ-VAE in Sections 2.4 and 6.1.1) and loudness-invariant (i.e. disentangled from loudness, which is included as a separate feature). Timbre conversion is done simply by feeding an arbitrary audio input to this end-to-end model, which has been trained for a specific target instrument as in Engel et al. Interestingly, the method allows not only preserving the pitch and loudness of the input, but also translating within-instrument timbre variations.

Both methods achieve impressive audio quality, but require training for each individual target instrument, unlike the method of Driedger et al. [2015] and our own method laid out in Chapter 6.

2.3 Domain translation with cyclic consistency

When styles are thought of as *classes* or *data domains*, style conversion may be framed as a **domain translation** task, a concept which is not limited to music, but may be applied to any data modality.

From a machine learning perspective, we can approach the task in a supervised or an unsupervised manner, depending on the kind of available training data. A classical example of a translation task where *aligned* (*parallel*) data is readily available, and therefore a supervised approach is feasible, is machine translation (MT) between natural languages. This field has a relatively long tradition, with the first data-driven machine translation systems dating back to the early 1990s [Brown et al., 1990]. On the other hand, music style translation is a task where a sufficient amount of aligned data is typically difficult or impossible to collect. This leads to a need for unsupervised domain translation methods, which have only become available in recent years thanks to deep learning.

A major advantage of unsupervised translation methods is that they usually do not require a prior notion of content (what is shared across domains) and style (what should be changed by translation) – these concepts are learned completely from data. However, the goal is poorly defined, and unless sufficient constraints are imposed, the learned mapping may not be meaningful.

In this section, we focus on domain translation using *cycle-consistent generative adversarial networks* (*CycleGAN*s). CycleGANs were developed for unsupervised image-to-image translation by Zhu et al. [2017]. The learning algorithm observes unpaired examples from two domains $\mathcal{X}$ and $\mathcal{Y}$ and learns two mutually inverse mappings $G\colon \mathcal{X} \to \mathcal{Y}$ and $F\colon \mathcal{Y} \to \mathcal{X}$. The authors propose to employ two techniques, displayed in Fig. 2.3, to constrain these mappings:

(1) An *adversarial objective* to force the outputs of F and G to be indistinguishable from real examples from the domains $\mathcal{X}$ and $\mathcal{Y}$, respectively.

    Specifically, the mapping G is coupled with a discriminator network $D_{\mathcal{Y}}$, which aims to distinguish the generated samples $G(x)$ from real samples y; in the same way, F is coupled with a discriminator $D_{\mathcal{X}}$. The mappings F and G are trained to fool the respective discriminators, which ensures a match between the distributions of their outputs and the real data distributions. Each mapping together with its associated discriminator hence constitutes a *generative adversarial network* (*GAN*; Goodfellow et al. [2014]).

(2) Two *cycle consistency losses*, enforcing the constraint that F and G are each other's inverse, i.e. $F(G(x)) \approx x$ and $G(F(y)) \approx y$.

Applying CycleGANs (and GANs in general) to sequential data such as music is more challenging due to the autoregressive nature of most sequence generation models, such as recurrent neural networks (RNNs), Transformers or WaveNets. In these models, generation proceeds sequentially, always conditioned on the outputs sampled in all previous steps, which typically makes the process non-differentiable, especially in the case of symbolic music, which is itself a discrete, non-differentiable object. This prevents from training the model by straightforward gradient back-propagation.

Huang et al. [2019b] use a CycleGAN for **audio timbre translation**, but avoid generating raw audio by applying the CycleGAN to spectrograms. A conditional WaveNet, trained on spectrogram-waveform pairs, is then used to synthesize the audio. Brunner et al. [2018b] employ a CycleGAN for **symbolic music style translation**, representing the music as a piano roll (a binary matrix of note activations along time). In both cases, the music representation can be essentially treated as a monochrome image, which allows to use a standard convolution-based architecture.

2.4 Representation disentanglement

Another popular concept used for unsupervised learning of transformations is that of *representation disentanglement*. The aim of disentanglement is to learn features that are *interpretable* in the sense that each feature or group of features represents a particular attribute ('semantic factor') and can be manipulated independently of the other features without affecting other attributes. This is especially useful in the context of autoencoders and generative models, since it allows transforming data samples along these attributes by manipulating the corresponding features. To perform style transformations, it is desirable to disentangle *style* from *content*; for example, style transfer can then be achieved simply by combining one example's content representation with the style representation of another.

Methods attempting to achieve disentanglement purely using unlabeled data (i.e. by automatically discovering the factors to be disentangled) have been proposed, e.g. by Chen et al. [2016] and Karras et al. [2019], but doing so for high-level attributes, such as style and content in music, remains a hard problem. For this reason, we focus on settings where these attributes are known and labels are available.

One of the first works in this area [Lample et al., 2017] succeeded in disentangling specific attributes of human faces (e.g. age, gender, glasses) in an image autoencoder. The method is inspired by GANs, but the adversarial training is performed in the autoencoder's *latent space* to enforce disentanglement. Specifically, the encoder maps the input x to a latent representation z, which is then used not only by the decoder to reconstruct x, but also by a discriminator to predict the associated attributes y. The encoder and the discriminator are trained adversarially, meaning that the goal of the encoder is to make it impossible for the discriminator to identify the attributes y from z.

Similar techniques are used by Lample et al. [2018] to perform unsupervised natural language translation, bringing the task closer to the one discussed in Section 2.3. This time, the attribute y is the language (e.g. English or French) and the goal is for z is to encode the meaning of the sentence x in a language-independent way.

A different method that we wish to include here is the *vector-quantized variational autoencoder* (*VQ-VAE*), proposed by van den Oord et al. [2017]. A VQ-VAE is an autoencoder with a *discrete* latent space. Its latent representation is a sequence of discrete symbols from a (learned) finite dictionary, placing an explicit limit on its capacity. While the authors' goal is not representation disentanglement, they show experimentally (without theoretical guarantees) that after training this model on speech with conditioning on the speaker identity, it is pos-

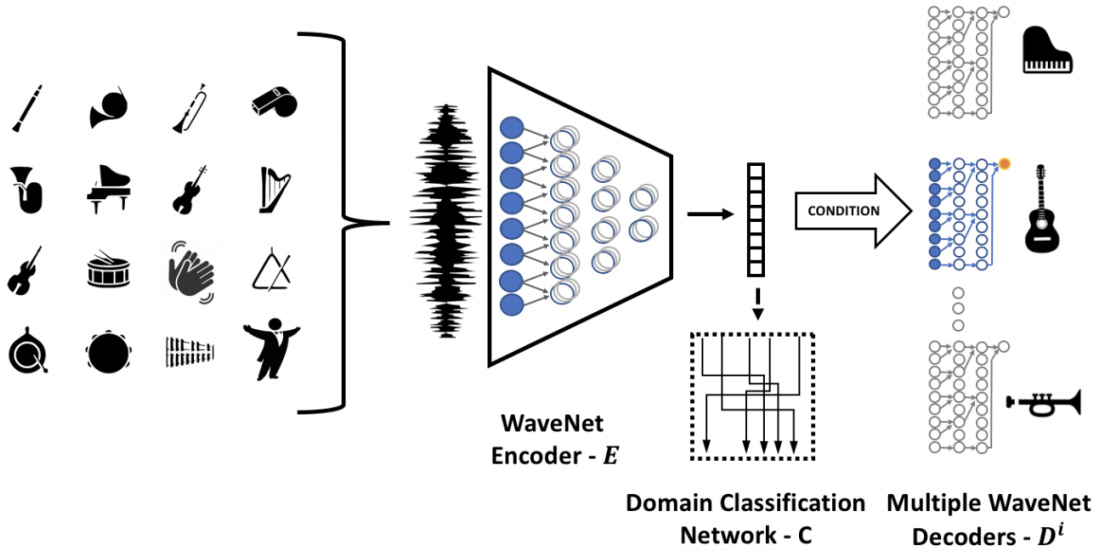


Figure 2.4 – The music translation network of Mor et al. [2019]. The input is encoded using a WaveNet encoder shared between all domains and the output is generated using a domain-specific decoder. The network is trained as an autoencoder (always selecting the appropriate decoder according to the training example) while using the domain classification (confusion) network to encourage the intermediate representation to be domain-independent. Reproduced from Mor et al. [2019]. © 2018 Noam Mor, Lior Wolf, Adam Polyak, Yaniv Taigman.

sible to achieve voice conversion simply by switching the speaker label. They conclude that the model learns 'a high-level abstract space' capable of representing an utterance in a speaker-invariant way – in other words, the autoencoder's latent space becomes disentangled from the speaker embedding space. This inspired our music timbre transfer approach, presented later in Chapter 6; we will give a more detailed explanation of the VQ-VAE there.

Turning finally to music, an approach very similar to Lample et al. [2018] has been applied to **music audio translation** by Mor et al. [2019], using a WaveNet [van den Oord et al., 2016] autoencoder architecture (see Fig. 2.4). As a major difference to Lample et al., the framework accounts for multiple domains $\mathcal{X}_1, \ldots, \mathcal{X}_n$, which in this case correspond to different composers and instrumentations (e.g. $\mathcal{X}_1$ = Mozart's symphonies, $\mathcal{X}_2$ = Haydn's string quartets etc.). The discriminator then becomes a multi-class classifier and is referred to as the *domain confusion network*. The encoder is still shared for all source domains, but a separate decoder is used for each target domain.[1]

For **symbolic music style** (genre) **translation**, Brunner et al. [2018a] propose a *variational autoencoder* (*VAE*) trained on a multi-track representation in two different genres. To promote style-content disentanglement, the encoder is forced to output an explicit one-hot encoding of the style label in the first two dimensions of the latent representation. This is done by attaching a softmax with a cross-entropy loss to these two dimensions, and adding it as an additional term

[1]The authors note that their attempts to train a single decoder conditioned on the target domain were unsuccessful.

to the loss function. The authors argue that this, together with the VAE loss, is enough to ensure disentanglement. Once the model is trained, style conversion is achieved by swapping the dimensions corresponding to the two styles.

In contrast, research on (one-shot) **music style transfer** via representation disentanglement is more recent and concurrent to our work.

Hung et al. [2019] propose an encoder-decoder model for music transcription (i.e. to convert audio to a symbolic representation) with a disentangled latent representation of pitch and timbre (instrumentation). Adversarial training is employed, designed to mutually disentangle the pitch and timbre representations. The trained model then allows to do a form of composition style transfer by changing the instrumentation. However, it is not clear whether it has one-shot capabilities, as it is not evaluated in this setting.

Yang et al. [2019] propose a model based on a VAE for disentangling the pitch and rhythm components of a melody. This is achieved by designating a part of the latent code as the rhythm features z_r (similarly to Brunner et al. [2018a]) and using them to reconstruct the rhythm only, by means of a rhythm decoder. This reconstructed rhythm is then used, together with the rest of the latent code, z_p (i.e. the pitch features), as input to a 'global' decoder to reconstruct the whole melody.

Similarly, Wang et al. [2020b] use a VAE to disentangle the representations of the harmony (chords) and the 'texture' of an accompaniment, each encoded using a dedicated encoder. In this case, however, the input to the chord encoder is already a chord sequence produced by a rule-based chord extraction system. The texture encoder receives a piano-roll representation and uses a convolutional architecture with hyperparameters carefully picked to obtain 'blurry' features with little information about the chords.

Still in a similar spirit, Kawai et al. [2020] and Wu and Yang [2021] train VAEs conditioned on pre-defined musical attributes that can be computed automatically, such as note density and degree of polyphony. Kawai et al. [2020] use an adversarial loss to disentangle the learned representation from these attributes, while Wu and Yang [2021] employ a β-VAE objective [Higgins et al., 2017] with free bits [Kingma et al., 2016]. By considering these attributes to represent style, a form of style transfer can then be achieved simply by transferring these musical attributes from one piece to another (though this is not explicitly studied in these works). Similarly, Nistal et al. [2021] train an audio GAN conditioned on pitch, as well as on attribute probabilities ('soft labels') produced by a pre-trained teacher model.

2.5 Other related work

2.5.1 Self-supervised music completion

A broad family of music transformation tasks is what we could call music *completion*, *inpainting* or *infilling*. Here, we are given a partial piece of music (e.g. a score with missing parts) and wish to generate more musical material to 'fill in the blanks'.

Tasks like this are straightforward to learn from arbitrary music datasets, especially in the case of symbolic music, since we can simply mask out (i.e. treat as unobserved) a part of the input and then train our model to recover it while *conditioned on* the context. Such methods may be classified as *self-supervised learning*, where training inputs and/or targets (labels) are generated from unlabeled data in an automated way.[2]

For example, the Drumify model introduced by Roberts et al. [2019] is trained to reconstruct a drum groove given only the note onset times. Hakimi et al. [2020] consider the problem of generating a jazz solo given a chord sequence; their model is trained on a dataset of jazz solo transcriptions, which contain the solo melody along with the chord sequence.

More relevant to our work are systems that are additionally conditioned on the target style, and hence achieve a form of style transfer. Choi et al. [2020] use a Transformer autoencoder to harmonize a melody in the style of a given piano performance. Lattner and Grachten [2019] train a gated autoencoder (GAE) to generate a kick drum track for a given recording and are able to use it to transfer the kick drum style from one song onto another. Grachten et al. [2020] extend this approach to bass lines.

2.5.2 Music generation with constraints

The methods from the previous subsection directly model the conditional distribution from which we then wish to sample. Another option is to model the *joint* distribution, and impose some constraints on it at runtime in order to incorporate existing musical material.

Such techniques can be viewed as another way to approach the music completion task from the previous subsection. For example, Hadjeres et al. [2016], Hadjeres et al. [2017] and Huang et al. [2017] developed systems for generation of Bach chorales, which are possible to constrain to produce a given melody, and hence perform *harmonization* in the style of Bach. However, more complex constraints are possible. Hadjeres et al. [2016] mention the possibility of imposing chord labels without enforcing specific voicings. Herremans and Chew [2017], on the other hand, constrain a generated piece to have a given 'tension profile' and a particular structure of repeating patterns. This tension profile can be extracted from a template piece or specified by the user. Moreover, it is possible to introduce other soft constraints e.g. to fix the pitches of specific notes.

Inspired by these ideas, Lu and Su [2018] approach music style conversion by iteratively updating a piece of music using pseudo-Gibbs sampling (as in Hadjeres et al. [2017]) with the constraint of keeping the melody fixed. Style conversion is achieved by using a model trained on the desired target style.

2.5.3 Expressive performance rendering

The last task that we will touch on is that of rendering a musical score as an expressive performance, which typically means predicting expressive dynamics and/or timing. This is a rare case where paired data is easy to acquire (unlike in Sections 2.3 and 2.4), since given a performance, the corresponding score is usually

[2]We will say more about self-supervised learning in Section 6.1.2

readily available, at least in Western classical music. Predicting the performance variables from the score then becomes a straightforward machine learning task, which can be viewed as a type of supervised translation (as in Malik and Ek [2017]). For example, Widmer et al. [2009] use a simple Bayesian model which predicts three variables for each note – tempo, loudness and articulation – given some features representing its context. Malik and Ek [2017] use an RNN to predict the sequence of loudness values given the sequence of pitches.

2.6 Conclusion

We have reviewed the literature related to music style conversion. In sum, a wide variety of works are related to ours in that they transform or complete existing music in some way. However, prior work on music style conversion, as we have defined it, is limited. In particular, all existing work on one-shot symbolic music style transfer is concurrent to ours, which we will present in Chapter 4. Similarly, in the audio domain, prior work on style conversion mostly concerns translation as opposed to one-shot transfer. The few existing works on one-shot audio style transfer either do not specifically focus on musical timbre, or are not learning-based, both of which is in contrast to our work presented in Chapter 6.

3. Background: Deep learning for sequence generation

In this work, we make extensive use of deep neural networks for sequence generation, namely neural language models and sequence-to-sequence networks. Such models are applicable whenever we need to learn to generate data represented as sequences of discrete symbols. In recent years, they have become extremely popular, achieving state-of-the-art results on many language processing tasks [Bahdanau et al., 2015, Vaswani et al., 2017, Radford et al., 2018, Devlin et al., 2019, He et al., 2021, *inter alia*], and more recently also leading to advances in music generation and transformation [e.g. Donahue et al., 2019a, Huang et al., 2019a, Huang and Yang, 2020, Choi et al., 2020]. This chapter serves as an introduction to these models, which have several peculiarities relating both to their training and their use for generation.

Note that there exists a variety of other deep learning techniques that have also been applied to music generation. For a survey of these techniques, see Briot et al. [2020].

3.1 Autoregressive neural language models

A language model (**LM**) can be viewed as a way to model probability distributions over sequences of symbols (tokens), i.e. $P(y) = P(y_1, \ldots, y_m)$, where $y_1, \ldots, y_m$ are symbols from a finite vocabulary. These symbols are traditionally words, but may also correspond to characters, subword units or – as in this work – musical events (e.g. notes).

The most common way to build LMs is to factorize the distribution $P(y)$ using the chain rule:

$$P(y_1, \ldots, y_M) = \prod_{m=1}^{M} P(y_m \mid y_1, \ldots, y_{m-1}).$$

This leads to an **autoregressive** language model, where the problem is reduced to modeling the conditional distribution $P(y_m \mid y_1, \ldots, y_{m-1})$ – a classification task also known as *next word prediction*. Such models have many advantages, especially the ease of training and their ability to model complicated distributions and draw samples from them.

Training is typically done via *maximum-likelihood estimation* (MLE), which can be formulated as minimizing a *negative log-likelihood* (NLL) loss:

$$\begin{aligned} \mathcal{L}(y; \theta) &= -\log \prod_{m=1}^{M} P_\theta(y_m \mid y_1, \ldots, y_{m-1}) \\ &= -\sum_{m=1}^{M} \log P_\theta(y_m \mid y_1, \ldots, y_{m-1}). \end{aligned} \tag{3.1}$$

In modern applications, P_θ is usually parameterized using a neural network which receives the tokens $y_1, \ldots, y_{m-1}$ and outputs a softmax probability distribution over y_m, and trained using some form of stochastic gradient descent over a training dataset.

3.1.1 Generation

A trained autoregressive LM can be used to **generate** new sequences – i.e. **sample** from the learned distribution $P_\theta(y_1, \ldots, y_M)$ – in a left-to-right fashion. In the m-th step of this process, we feed the previously generated prefix $\hat{y}_1, \ldots, \hat{y}_{m-1}$ to the model and sample $\hat{y}_m$ randomly according to $P(y_m \mid \hat{y}_1, \ldots, \hat{y}_{m-1})$. Apart from generating novel sequences from scratch, the model also lends itself to **continuation**, where a *prompt* $y_1, \ldots, y_{m-1}$ is given, and generation proceeds from the m-th step.

While this procedure is rather simple, it requires a set of standard tricks to achieve success. Firstly, sampling $\hat{y}_1 \sim P(y_1)$ would require conditioning the model on an empty sequence, which is not normally possible. For this reason, a **beginning-of-sequence** token $y_0 =$ `<s>` is prepended to all training sequences, and generation then starts by sampling $\hat{y}_1 \sim P(y_1 \mid y_0 =$ `<s>`$)$. Similarly, an **end-of-sequence token** `</s>` is appended to all training sequences; during generation, sampling this special token terminates the generation loop.

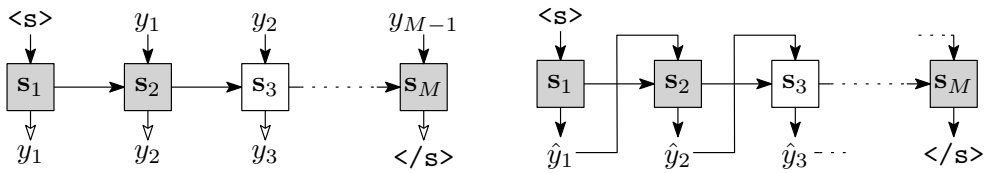
A commonly encountered problem is that sampling naïvely from the learned distribution P can easily lead to incoherent outputs due to overestimating the probabilities of relatively unlikely tokens – a phenomenon called the 'unreliable tail' by Holtzman et al. [2020]. A well-known way to alleviate this problem is **sampling with temperature**, where the logits (pre-softmax activations) obtained from the model are divided by a temperature parameter $\tau > 0$ prior to applying the softmax. With $\tau < 1$, this leads to a lower-entropy distribution, reducing the probability mass assigned to the unreliable tail, but also decreasing the overall diversity of the outputs [Caccia et al., 2018, Hashimoto et al., 2019]. As τ tends to 0, the distribution becomes one-hot, i.e. all the probability mass is assigned to the most likely outcome; this limit behaviour, where the most likely token is chosen deterministically at each step, is known as **greedy decoding**.

Another sampling strategy is **top-k sampling** [Fan et al., 2018, Holtzman et al., 2018], where only the k most likely candidates are considered and the probability of the rest is forced to be equal to 0. **Nucleus sampling** [Holtzman et al., 2020] is an improvement on this strategy, where the number k of considered candidates is chosen dynamically (controlled by a parameter p) based on the shape of the distribution, which is shown to lead to higher-quality outputs.

3.1.2 Model architectures

As mentioned above, a neural language model needs to be able to consume a (variable-length) sequence of input tokens $y_1, \ldots, y_{m-1}$ and output a probability distribution over the next token, $P(y_m \mid y_1, \ldots, y_{m-1})$.

Before applying any neural network architecture to a token sequence, we first need to encode each token as a real-valued vector. This is done using an **embedding layer**, which selects the corresponding column of an *embedding matrix* $\mathbf{W}^{\mathrm{e}}$ (a trainable parameter); in other words, if $\mathbb{1}_{y_m}$ is a one-hot representation of y_m (indicating its position in the vocabulary), then the corresponding embedding can be written as $\mathbf{y}_m = \mathbf{W}^{\mathrm{e}}\mathbb{1}_{y_m}$. We can then feed the sequence $\mathbf{y}_1, \ldots, \mathbf{y}_{m-1}$ to any sequence processing network. This network will output a feature vector $\mathbf{h}_{m-1}$, which is then passed through a linear projection and a softmax operation to obtain a probability distribution over y_m.



(a) Training (b) Generation

Figure 3.1 – The standard training and generation procedure for an RNN language model. Arrows with white heads point to prediction targets.

Recurrent neural networks

One class of neural networks suitable for sequence processing are **recurrent neural networks** (**RNNs**). An RNN works by processing the input sequence from left to right and using each input $\mathbf{y}_m$ to update its internal ('hidden') state:

$$\mathbf{s}_m = f_{\text{RNN}}(\mathbf{y}_{m-1}, \mathbf{s}_{m-1}) \tag{3.2}$$

where f_{RNN} is a non-linear function that depends on the type of the RNN and has trainable parameters. The state is a vector whose dimension (also called the number of *units*) is a hyperparameter of the RNN.

A very popular type of *RNN cell* – i.e. particular form of f_{RNN} from Eq. (3.2) – is **long short-term memory** (**LSTM**, Hochreiter and Schmidhuber [1997]). Many variations on the LSTM cell exist, including the simpler **gated recurrent unit** (**GRU**, Cho et al. [2014]), which has also enjoyed widespread use and which we employ here as well.

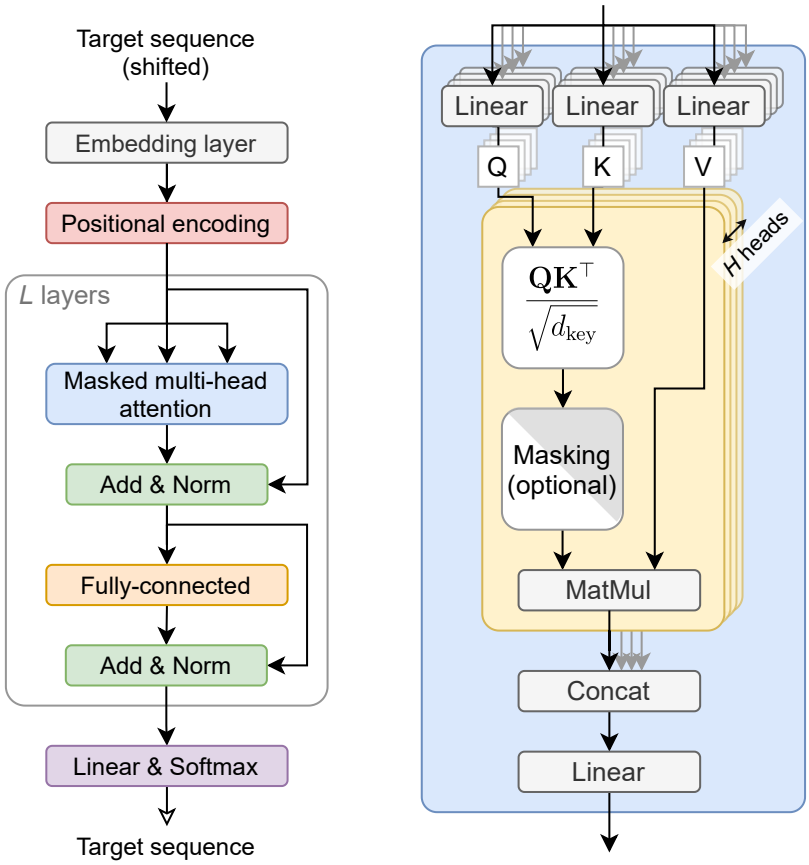
In practice, we do not need to run the RNN for each prefix $\mathbf{y}_1, \ldots, \mathbf{y}_{m-1}$ where $m \in 1, \ldots, M$, since we can easily obtain all $\mathbf{s}_m$, and hence compute the sum in Eq. (3.1), in a single run of the RNN over the entire input sequence $\mathbf{y}_1, \ldots, \mathbf{y}_{M-1}$. This leads to the training procedure displayed in Fig. 3.1a. Generation is also relatively time- and memory-efficient, since as shown in Fig. 3.1b, in each step, we only need the last computed state $\mathbf{s}_{m-1}$ and the embedding $\mathbf{y}_{m-1}$ of the last sampled token to compute the new state $\mathbf{s}_m$ and sample the new token $\hat{y}_m$.

Transformers

Another popular choice of architecture is the Transformer [Vaswani et al., 2017]. Unlike RNNs, a Transformer does not keep track of history using a single state vector, but instead has access to the entire history by means of a **(self-)attention mechanism**. This makes it a more powerful model than RNNs, but slower for generation, since in each step, it needs to consider all previous steps, leading to an overall quadratic time complexity.

The Transformer architecture,[1] displayed in Fig. 3.2, alternates fully-connected layers, which are applied independently to each position, and multi-head self-attention layers, which take care of passing information between different positions. The idea of the self-attention layer is to compute in a set of *attention scores* or *weights*, expressing the importance of each position to the current

[1]The original Transformer proposed by Vaswani et al. is in fact an encoder-decoder model, which we describe in Section 3.2. What is described here is the *decoder-only* variant of the Transformer, commonly used as a basis for LMs, e.g. by Radford et al. [2019].



(a) High-level architecture (b) Multi-head (self-)attention

Figure 3.2 – The architecture of a Transformer language model.

one, and use these to weight the information at those positions. As illustrated in Fig. 3.2b, this is done by considering each position m simultaneously as a *key* $\mathbf{k}_m = \mathbf{W}^{\mathrm{K}}\mathbf{g}_m$, a *query* $\mathbf{q}_m = \mathbf{W}^{\mathrm{Q}}\mathbf{g}_m$ and a *value* $\mathbf{v}_m = \mathbf{W}^{\mathrm{V}}\mathbf{g}_m$, where $\mathbf{g}_m \in \mathbb{R}^{d_{\text{model}}}$ are the input features of the self-attention layer at position m, and $\mathbf{W}^{\mathrm{Q}}, \mathbf{W}^{\mathrm{K}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{key}}}$ and $\mathbf{W}^{\mathrm{V}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{value}}}$ are trainable parameters. A *scaled dot product* between the current query and every key is then computed, and the resulting scores are used to compute a weighted average of the values. Formally, with the queries, keys and values gathered in respective matrices $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{M \times d_{\text{key}}}$ and $\mathbf{V} \in \mathbb{R}^{M \times d_{\text{value}}}$, the attention mechanism can be written as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_{\text{key}}}}\right)\mathbf{V}, \tag{3.3}$$

where the softmax is applied along the key dimension, so that each element in the result is a convex combination of all the values, i.e.:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \left[\frac{\sum_{n=1}^{N} \exp(e_{mn})\mathbf{v}_n}{\sum_{n'=1}^{N} \exp(e_{mn'})}\right]_m, \tag{3.4}$$

$$e_{mn} = \mathbf{q}_m^\top \mathbf{k}_n \Big/ \sqrt{d_{\text{key}}}. \tag{3.5}$$

Moreover, it turns out to be useful to apply the attention mechanism not just once, but several (H) times in parallel (with different parameters), obtaining a different result each time. This is called **multi-head (self-)attention** with H attention heads. Here, the queries for the h-th head are obtained as $\mathbf{q}_m = \mathbf{W}^{\mathrm{Q}}_{(h)}\mathbf{y}_m$, and analogously for keys and values, with $d_{\text{key}} = d_{\text{value}} = D = d_{\text{model}}/H$. The outputs of all attention heads are then concatenated and projected back to d_{model} dimensions using an additional linear layer.

To use a Transformer as a language model, it can be trained analogously to the RNN in Fig. 3.1a. However, note that by simply applying Eq. (3.3) during training, every query would be able to attend to every key, which is undesirable, since during generation, only past positions (i.e. those corresponding to tokens that have already been sampled) are available. For this reason, a **lower triangular** ('**causal**') **mask** is applied to the pre-softmax attention matrix $\mathbf{Q}\mathbf{K}^\top/\sqrt{D}$ during training, forcing all terms above the diagonal to equal $-\infty$. This ensures that $\mathbf{q}_m$ only has access to $\mathbf{k}_1, \ldots, \mathbf{k}_m$ while keeping the attention scores properly normalized.

The last piece of the puzzle is **positional encoding** (**PE**). The fully-connected and self-attention layers of the Transformer are permutation-invariant operations, meaning that shuffling the input positions will not change the output values.[2] For this reason, a positional encoding vector is usually added to each input embedding vector as a form of 'positional bias'. This is sometimes referred to as *absolute positional encoding* (*APE*). Vaswani et al. [2017] generate fixed APE by sampling trigonometric functions at different frequencies. Alternatively, APE can be learned as parameters of the model [Gehring et al., 2017, Devlin et al., 2019].

[2]Note that this is only true for the unmasked version of the Transformer, as the lower triangular mask does provide some sort of position information [Irie et al., 2019]. Nonetheless, positional encoding is used systematically in all Transformer variants.

An alternative method for injecting position information is **relative positional encoding** (**RPE**), which encodes *differences* between pairs of positions (*time lags*), as opposed to absolute positions. This information, encoded as a matrix $\mathbf{R}$, is then used to modify the computation of the attention scores in Eq. (3.3): $\left(\mathbf{Q}\mathbf{K}^\top + \mathbf{Q}\mathbf{R}^\top\right)/\sqrt{D}$.

3.2 Sequence-to-sequence models

Neural language models, described in the previous section, allow generating sequences from the data distribution. However, they become much more useful when we are able to condition them on some input data, and this is precisely what is achieved by **sequence-to-sequence** (**seq2seq**) models. Specifically, seq2seq models are used to model conditional distributions of the form $P(y \mid x) = P(y_1, \ldots, y_M \mid x_1, \ldots, x_N)$, where $x_1, \ldots, x_N$ represents an input sequence and $y_1, \ldots, y_M$ an output sequence. Factorizing this distribution along the output sequence as in Eq. (3.1), the task of the model at each step becomes to predict the distribution $P(y_m \mid y_1, \ldots, y_{m-1}, x_1, \ldots, x_N)$. The language model then becomes the *decoder* and the main challenge lies in conditioning it on the input sequence $x_1, \ldots, x_N$ via an appropriate *encoder*.

A popular way to connect the encoder to the decoder is via an **attention mechanism**. Note that this *encoder-decoder* attention (also called *cross-attention* or *inter-attention*), which will be detailed in the following section, is different from the self-attention from Section 3.1.2: while self-attention serves to exchange information between different positions inside the decoder, the role of encoder-decoder attention is to "route" features from the encoder to different positions in the decoder.

3.2.1 Model architectures

RNN encoder-decoder (without attention)

A simple way to condition an RNN on an arbitrary sequence is to use an encoder capable of summarizing the whole sequence in a single fixed-dimensional vector. The perfect encoder architecture for this task is again an RNN – this time, however, all the intermediate RNN states are ignored, and only the *final state* $\mathbf{h}_N$ (i.e. the state produced after consuming all the inputs) is kept. This final state is then used either as the initial state of the decoder RNN [Sutskever et al., 2014], or as an additional input to the decoder RNN cell in every step [Cho et al., 2014].

RNN encoder-decoder with attention

The above method relies on encoding the entire (variable-length) sequence into a single vector, which may not always be realistic. To address this problem, Bahdanau et al. [2015] enhance the RNN seq2seq model with *encoder-decoder attention*, which makes it possible for each decoder state to obtain a different, dynamically computed view of the input sequence. This is done by considering *all* the encoder states, denoted $\mathbf{h}_1, \ldots, \mathbf{h}_n$, computing a weight α_{mn} for every input position n and the current output position m, and using these weights to calculate

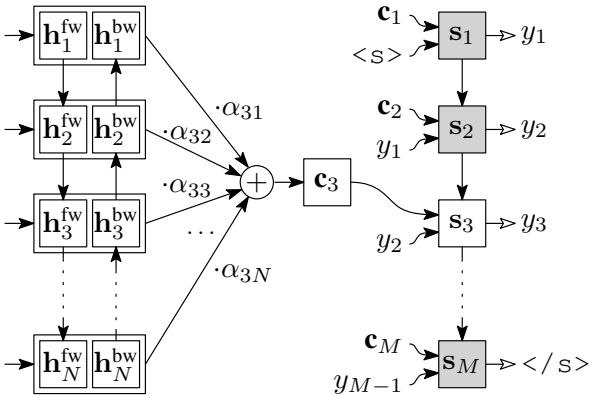


Figure 3.3 – RNN encoder-decoder (seq2seq) model with attention, as proposed by Bahdanau et al. [2015]. For illustration purposes, the diagram focuses on the 3rd decoding step, i.e. $m = 3$. Arrows with white heads point to prediction targets. Modified from ⟨ISMIR2019⟩.

a weighted average $\mathbf{c}_m$ of the encoder states, similarly to Eq. (3.4). Finally, $\mathbf{c}_m$, known as the *context vector*, is used as input to the decoder RNN cell f_{RNN} to obtain the new state $\mathbf{s}_m$:

$$\mathbf{s}_m = f_{\text{RNN}}([\mathbf{y}_{m-1}, \mathbf{c}_m], \mathbf{s}_{m-1}), \tag{3.6}$$

$$\mathbf{c}_m = \frac{\sum_{n=1}^{N} \exp(e_{mn})\mathbf{h}_n}{\sum_{n'=1}^{N} \exp(e_{mn'})}, \tag{3.7}$$

$$e_{mn} = f_{\text{att}}(\mathbf{s}_{m-1}, \mathbf{h}_n). \tag{3.8}$$

In the original "Bahdanau-style" attention, f_{att} is implemented as a fully-connected neural network; more similarly to the later-proposed Transformer, Luong et al. [2015] suggest using a dot product (cf. Eq. (3.5)).

Bahdanau et al. also propose to use a *bidirectional RNN* [Schuster and Paliwal, 1997] in the encoder, i.e. $\mathbf{h}_n = [\mathbf{h}_n^{\text{fw}}, \mathbf{h}_n^{\text{bw}}]$, where $\mathbf{h}_n^{\text{fw}}$ is produced by a forward RNN and $\mathbf{h}_n^{\text{bw}}$ is produced by a backward RNN, which consumes the input from right to left. As a result, $\mathbf{h}_n$ captures the context of the entire input, but is expected to focus around position n.

The model is illustrated in Fig. 3.3.

Transformer encoder-decoder

For completeness, let us also mention the encoder-decoder variant of the Transformer [Vaswani et al., 2017]. Analogously to the model that we just described, it consists of a Transformer encoder and a Transformer decoder, connected using an attention mechanism. Hence, attention is used in three different places in the model: as *encoder* self-attention, *decoder* self-attention, and *encoder-decoder* attention.

As shown in Fig. 3.4, the architecture of the encoder is similar to the one described in Section 3.1.2 (the only difference is that no attention mask is used, so that the whole input is accessible to the self-attention mechanism at any time). As a major addition, the decoder now has the encoder-decoder (cross-)attention inserted in every layer. This encoder-decoder attention is computed according to

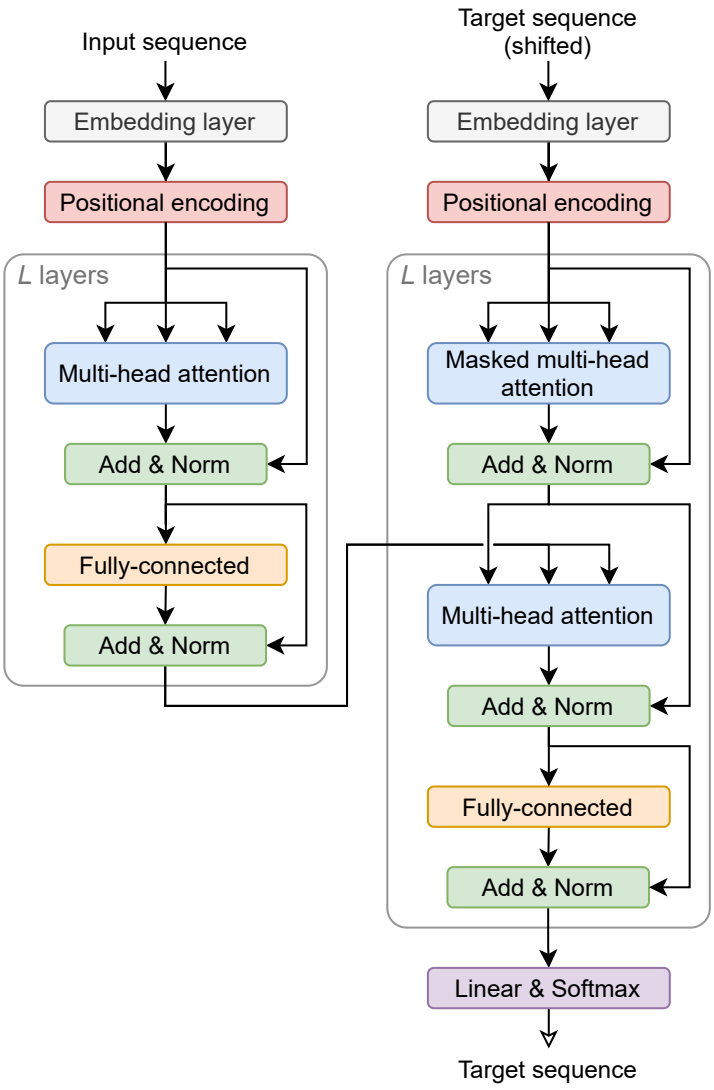


Figure 3.4 – The architecture of a Transformer encoder-decoder model. The encoder (left) encodes the input sequence as a series of feature vectors, which are then used as keys and values in the encoder-decoder attention mechanism inside the decoder (right).

Eq. (3.3); here, however, the keys and values are computed from the outputs of the encoder rather than from decoder features. In other words, we have:

$$\mathbf{k}_n = \mathbf{W}^{\mathrm{K}}\mathbf{h}_n, \quad \mathbf{v}_n = \mathbf{W}^{\mathrm{V}}\mathbf{h}_n, \quad \mathbf{q}_m = \mathbf{W}^{\mathrm{Q}}\mathbf{g}_m, \tag{3.9}$$

where $\mathbf{h}_1, \ldots, \mathbf{h}_N \in \mathbb{R}^{d_{\text{model}}}$ are *encoder* features for the input sequence $x_1, \ldots, x_N$, and $\mathbf{g}_1, \ldots, \mathbf{g}_{M-1} \in \mathbb{R}^{d_{\text{model}}}$ are the *decoder* features for the output sequence $y_1, \ldots, y_{M-1}$.

3.3 Conclusion

In this chapter, we have introduced the principles behind deep neural networks for sequence generation and translation, namely neural language models and sequence-to-sequence (seq2seq) models, as well as the most popular architectures used in this context: RNNs and Transformers. In the next chapter, we will build upon this knowledge while describing our first contribution, consisting in methods for accompaniment style conversion.

4. Supervised symbolic music style conversion

This chapter is based on the papers ⟨**ISMIR2019**⟩ and ⟨**TASLP2020**⟩ by Cífka et al. © 2020 IEEE.

In this chapter, we present two studies focusing on **accompaniment style conversion** 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. 4.2). 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 final goal in this chapter is **one-shot accompaniment style transfer**: given a musical piece (the *content input*), we wish to generate a new accompaniment for it in the style of a different piece (the *style input*). Note that even though we expect the output to follow the same chord chart as the content input, we do not assume this chart to be available.

The first study, presented here in Section 4.3, takes a first step towards this goal by tackling a simpler *accompaniment style translation* task. The only input in this case is the content input, and the target style is simply selected from one of a small set of available options. The method developed here is then extended to the full *one-shot* setting in the second study in Section 4.4.

Since the methods proposed in the two studies share a common basis, relying on *supervised learning* from *synthetically generated accompaniments*, we first give a general overview of this core idea in Section 4.1. Then, in Section 4.2, we set forth automatic evaluation metrics developed to gauge the performance of our systems. Finally, the core of both studies follows in Sections 4.3 and 4.4, respectively.

4.1 Methods overview

Generally speaking, we are interested in learning a transformation $(x, z) \mapsto y$, where x is a song fragment in an arbitrary style S, y is the same fragment in a different style T (the *target style*), and z is an indication of the style T. In the case of style translation (Section 4.3), where the set of possible target styles is known and finite, z will be a discrete label identifying T. In the case of one-shot style transfer (Section 4.4), z will be an example of T, i.e. another fragment in style T.

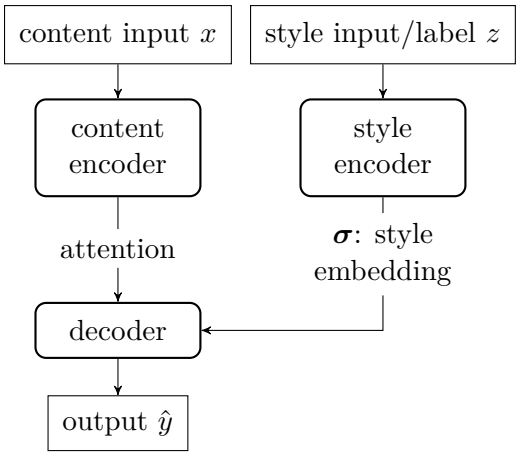


Figure 4.1 – The high-level structure of the proposed style conversion models. The input z to the style encoder is either an accompaniment track (in the one-shot style transfer case) or a style label (in the style translation case – the style encoder then becomes a simple embedding layer). © 2020 IEEE.

As we have seen in Chapter 2, a major difficulty associated with music style conversion tasks is that there are no publicly available 'aligned' or 'parallel' datasets (containing examples of the same music played in different styles) to learn from.[1] As a result, works that predate ours mostly adopt unsupervised learning frameworks and apply them to genre-labeled datasets. In our work, we adopt a different strategy to overcome the lack of aligned data, which is to *synthesize* it. Synthetic training data has proven useful for music information retrieval tasks such as chord recognition [Lee and Slaney, 2008] and fundamental frequency estimation [Mauch and Dixon, 2014, Salamon et al., 2017], and is also popular for tasks like semantic segmentation in computer vision [Ros et al., 2016, Varol et al., 2017]. In our case, synthetic data opens up the possibility for supervised learning techniques known from the machine translation field. Moreover, it allows us to work with fine-grained style labels, as opposed to genre labels, which may be too vague or ambiguous for such purposes.

Our models proposed here are neural networks with an encoder-decoder structure, as depicted in Fig. 4.1. In both cases, we have a *content encoder* that processes the input x (the *content input*) and a *style encoder* processing the input z (the *style input*); the output $\hat{y}$ is then produced by the *decoder* based on the representations computed by the two encoders. In the style translation case, z is simply a label identifying the target style T and the style encoder degenerates to a simple embedding layer. In both cases, the output of the content encoder is a sequence of feature vectors, which are accessed by the decoder via an attention mechanism; on the other hand, the output of the style encoder is a single embedding vector $\boldsymbol{\sigma}$.

An accompaniment typically consists of multiple tracks (e.g. bass, piano, drums). While we could in principle model all the output tracks jointly, we break the problem down by assuming conditional independence of the output tracks given the inputs and by training our models to generate only one track at a time. The complete accompaniment is then obtained by combining the outputs

[1] With the exception of our datasets, presented in this work.

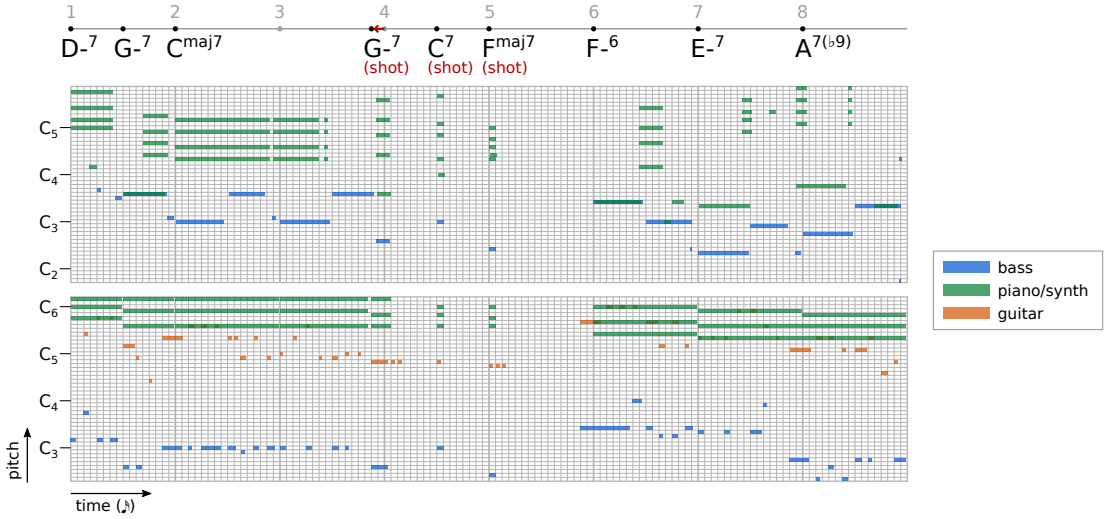


Figure 4.2 – 8-bar excerpts of accompaniments (generated by Band-in-a-Box) in two different styles (top: Jazz Swing, bottom: Progressive Rock) visualized as piano rolls. Both correspond to the chord chart displayed above them. Drums are not included. A 16$^{\text{th}}$ note grid is shown for reference. © 2020 IEEE.

of multiple models or multiple runs of one model.

The architecture of the proposed models is derived from sequence-to-sequence models with attention, described in Section 3.2. The models are simply trained end-to-end to minimize the cross entropy of the ground-truth targets y (represented as token sequences) given the inputs x, z, i.e. $-\mathbb{E}\left[\log P(y \mid x, z)\right]$. This supervised training approach is possible due to our synthetic data generation scheme, outlined below in Section 4.1.1. To test the models, we run them on new pairs (x, z). In style translation, z is inevitably limited to styles from the training set; the one-shot style transfer model, on the other hand, should be tested with target styles not encountered during training.

4.1.1 Synthetic data generation

As we have mentioned, our approach proposed here involves synthetic data generation, allowing to obtain a virtually unlimited number of ground-truth examples (x, z, y), and hence perform supervised learning. In this section, we describe the general synthetic data generation method; details of the datasets generated for the individual studies will be given later in Sections 4.3.1 and 4.4.1.

Specifically, our synthetic dataset is generated using RealBand from the Band-in-a-Box (BIAB) software package.[2] 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 *comping style* or *'groove'* rather than a broad category like genre. For instance, over 150 BIAB styles are categorized as Blues, bearing such names as

[2] `https://www.pgmusic.com/`

'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).

The basic idea of our data generation scheme is to start with a set of chord charts and use BIAB to generate a number of renditions of each chart in different accompaniment styles. Once the dataset is generated, we may simply pick two generated accompaniments corresponding to the same chord chart, as in Fig. 4.2, and use one as the content input x and one as the target y. More precisely, to form the training triplets (x, z, y), we loop over all pairs (x, y) such that x and y correspond to the same chord chart segment, but $x \neq y$. We denote T the style of y (i.e. the target style). Then, in the style translation case, we have $z = T$ (a discrete label); in the style transfer case, we pick z uniformly randomly from all segments in style T.

Notice that only the accompaniment, and not the underlying chord chart, is fed to the model. This allows easily applying the trained model to inputs for which the chord chart is not available, in particular ones that are not BIAB-generated or are in a genre where the chord chart representation is not commonly used (e.g. classical music).

4.2 Evaluation

When evaluating style conversion, we need to consider two complementary criteria: how well the transformed music fits the desired style (*style fit*) and how much content it retains from the original (*content preservation*). Note that it is trivial (but useless) to achieve perfect results on either of these two criteria *alone*, so it is essential to evaluate both of them. In this section, we describe objective – automatically computed – metrics for both criteria.

4.2.1 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. Following Lu and Su [2018], we compute this metric by correlating the chromagram of the generated segment with that of the content input. A chromagram in this context is essentially a 2D histogram with time on the x axis and the 12 pitch classes on the y axis; in other words, it is a matrix that records how many notes from each pitch class are active in each (regularly sampled) time frame.

To measure content preservation, we compute the chromagram of each of the two segments (content input and output) at a rate of 12 frames per beat and smooth each of them using an averaging filter with a window size of 2 beats (24 frames) and a stride of 1 beat (12 frames). Finally, we calculate the average frame-wise cosine similarity between the two sets of chroma features.

4.2.2 Style fit

In some of the recent music style transformation works [Brunner et al., 2018a,b], the quality of a transformation is measured by means of a binary style classifier trained on a pair of styles. However, the merit of such evaluation is limited,

Metric	Observations	Bins
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 4.1 – Objective style fit metric definitions. Each metric is computed as a cosine similarity between *style profiles* – flattened 2D histograms of the observations defined in the middle column (values that fall outside the given ranges are ignored). start$(\cdot)$ and end$(\cdot)$ are the onset and offset time in beats, respectively, of a given note. pitch$(\cdot)$ and velocity$(\cdot)$ denote the respective MIDI values.

since a high classifier score merely demonstrates that the output has some of the distinguishing features of the target style, and not necessarily that it actually fits the style. For this reason, we aim for a more interpretable metric of style fit.

As observed e.g. by McKay [2004], Hadjeres et al. [2016], and Sakellariou et al. [2017], musical style is well captured in musical event statistics, especially pairwise statistics between neighboring events. Drawing inspiration from the features proposed by McKay [2004], we devise a set of so-called *style profiles*, and use cosine similarities between them as style similarity metrics. This allows us to measure the style fit of an output as the style similarity between this output and a reference (i.e. an example of the target style).

The proposed metrics are summarized in Table 4.1. Firstly, to compute the *(pairwise) time-pitch* metric (proposed in ⟨ISMIR2019⟩), we consider all pairs of note onsets less than 4 beats apart and at most 20 semitones apart, and record the time difference and interval for each pair. We then obtain the style profile as a normalized 2D histogram of all these pairs with 6 bins per beat and one bin per semitone, and flatten it to get a 984-dimensional vector.

Clearly, the pairwise time-pitch metric is invariant to time shifts, does not account for note duration or velocity (dynamics) and is not suitable for drums. For this reason, we complement it with 3 additional metrics (proposed later in ⟨TASLP2020⟩), computed on statistics of single notes: *onset-duration*, *onset-velocity* and *onset-drum*. These are defined analogously to the 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).

Note that we compute these metrics on each instrument track separately, since

we presume that the dependencies between notes played by different instruments are often too complex to be captured by our statistics and would only make them more noisy. This choice is also in line with the design of our models, which generate different tracks conditionally independently given the inputs.

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. B.1 in the appendix.

Levels of granularity

Our proposed models operate on 8-bar segments, which may not always be sufficient to compute reliable style profiles, making the metrics noisy. For this reason, we may want to collect the statistics on sets of segments with the same target style, but real-world datasets may not have style labels that are sufficiently fine-grained and accurate for this purpose. In fact, style may in some cases vary significantly even within a single piece. For this reason, we propose different variants of our style similarity metrics depending on the level of granularity:

1. *Nano*: a single style profile is computed for every segment, i.e. each segment is scored separately.

2. *Song-level*: for a given target style, the style profiles are aggregated over all segments in a song. In other words, every (*input song*, *target style*) pair gets a single score.

3. *Macro*: for a given target style, the style profiles are aggregated over all segments in the dataset, i.e. every target style gets scored as a whole.

Metrics validation

In this section, we validate the proposed style similarity metrics on the Bodhidharma dataset [McKay and Fujinaga, 2005], a diverse collection of 950 MIDI recordings annotated with genre labels. To this end, we compute the similarities (*nano* variant) between all pairs of 8-bar segments in Bodhidharma and compare inter- vs. intra-genre similarities. For simplicity, we limit the analysis to pitched instruments, and therefore include only the onset-duration, onset-velocity and time-pitch metrics. 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[3] and average the results.

First, we compare in the left part of Fig. 4.3 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, we would also expect our metrics to capture at least some characteristics of genres; this is demonstrated in the plot on the right, which shows that the average similarity of segments from the

[3]The General MIDI specification [MIDI Manufacturers Association, 1991] 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.

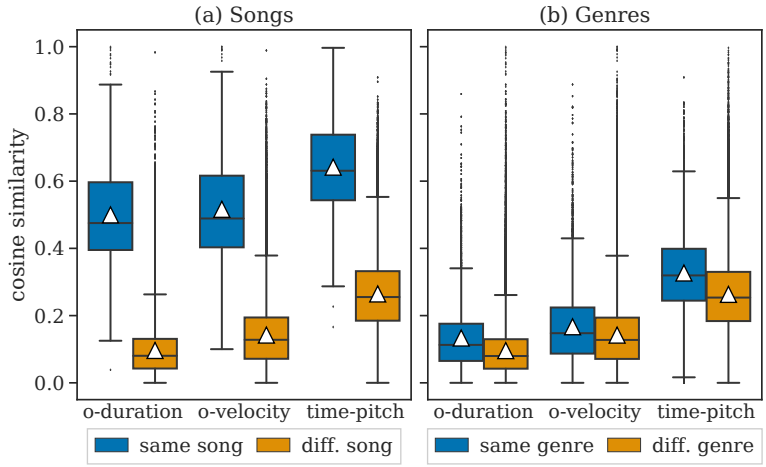


Figure 4.3 – Style similarities between pairs of segments from the Bodhidharma dataset. Plot (a) contrasts similarities within and across songs, while plot (b) contrasts similarities within and across the 38 genres in the Bodhidharma dataset. The values have been averaged so that every data point corresponds to a single pair of songs.

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. 4.4, 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.

4.3 Supervised style translation

This section details our work on supervised accompaniment style translation published in ⟨ISMIR2019⟩. The main contribution of this work is a neural model capable of translating between different accompaniment styles. As this is the first work in this direction, we chose to restrict ourselves to translating between 70 'artificial' styles included in the Band-in-a-Box (BIAB) software, and we only consider bass-and-piano accompaniments for simplicity.

Even in this limited setting, we were able to obtain interesting results. In particular, we show that a relatively simple RNN encoder-decoder architecture is able to achieve very good results on both style fit and content preservation, and that it even generalizes to inputs in styles other than encountered during training. These results provide an excellent stepping stone for the next study on one-shot style transfer (Section 4.4).

Additional materials for this study, notably listening examples[4] and source code,[5] are available online. A complete list of resources is given in Appendix C.

[4] `http://tiny.cc/musicstyle`
[5] `https://github.com/cifkao/ismir2019-music-style-translation`

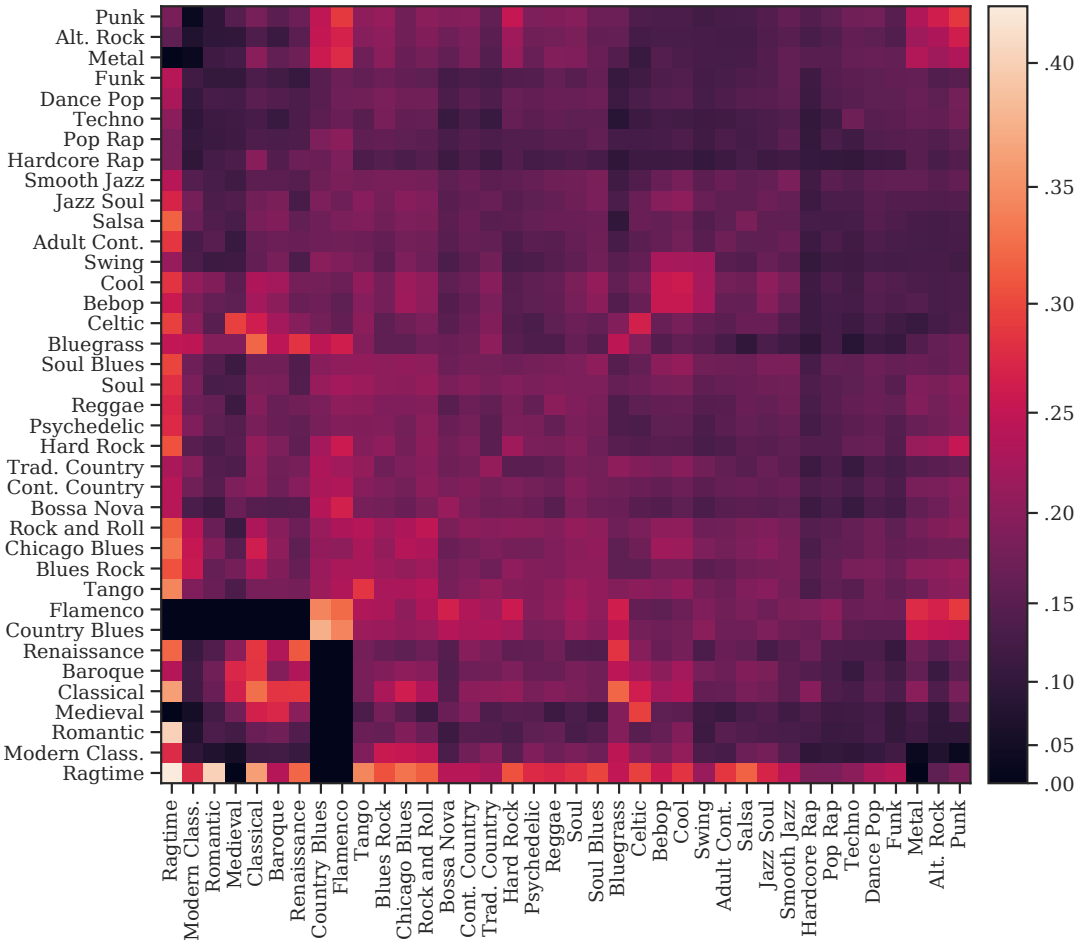


Figure 4.4 – Pairwise similarities between genres from the Bodhidharma dataset. The values are averaged over all 3 metrics and all pairs of segments. Values on the diagonal do not include pairs of segments from the same song. The order of rows and columns has been determined using hierarchical clustering.

4.3.1 Method

We propose an architecture based on RNN encoder-decoder sequence-to-sequence (seq2seq) models with attention, which we have described in Section 3.2. This choice is motivated by the successes of RNNs on symbolic music generation [Eck and Schmidhuber, 2002, Simon and Oore, 2017, Hadjeres et al., 2017, Sturm et al., 2016] and by the ability of the attention mechanism to condition the generation on arbitrary input data without a prior alignment.

Compared to the standard seq2seq model, our architecture has the extra style embedding layer, which is used to condition the decoder on the target style label. This allows a single model to translate to a potentially large number of different styles (similar to multilingual translation, e.g. Johnson et al. [2017]). On the other hand, to simplify the task and facilitate evaluation, we train a dedicated model for each target instrument track (bass, piano).

The rest of this section will cover our choices regarding input/output representation and model architecture and training, as well as details about how the synthetic dataset for this task was created.

Input and output representation

A common choice of representation of symbolic non-monophonic music for neural processing is a piano roll, a matrix that records which notes are active at each point in time. We use a binary-valued piano roll with 128 pitches and 4 columns per beat (quarter note) to encode our input.

For representing the output (and also as an alternative input representation), we opted for a MIDI-like encoding, which – unlike a piano roll – is straightforward to model using an RNN decoder. Specifically, following Simon and Oore [2017], we encode the music as a sequence of 3 types of events, each with one integer argument:

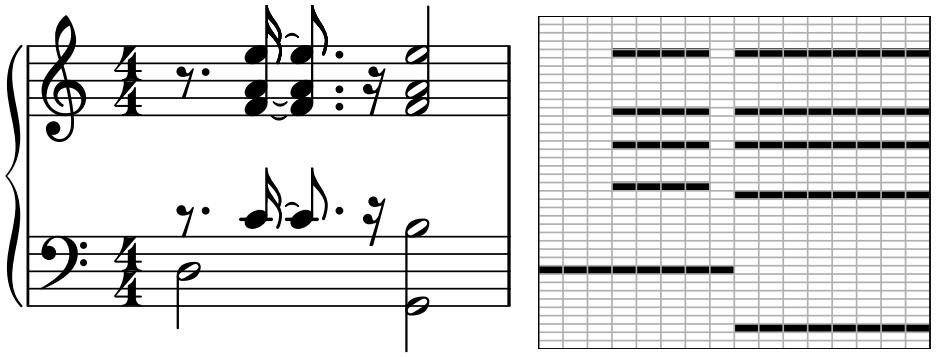
- `NoteOn(pitch)`: start a new note at the given pitch;
- `NoteOff(pitch)`: end the note at the given pitch;
- `TimeShift(delta)`: move forward in time by the specified amount, measured in 12ths of a beat.

`NoteOn` and `NoteOff` take values in the range 0–127, whereas `TimeShift` is within 1–24. When encoding the piano track, which is not monophonic, we compress the sequences by also including a `NoteOff(All)` event that ends all currently active notes. In contrast to Simon and Oore [2017], our representation is tempo-invariant and we do not model dynamics.

Fig. 4.5 illustrates both the piano roll representation and the token-based representation.

Model architecture, training and inference

As mentioned above (and depicted in Fig. 4.1), the proposed model is an encoder-decoder seq2seq model with attention and an additional style embedding layer. The architecture of the encoder (which we call the content encoder) depends on the type of input representation:



```
NoteOn(50) TimeShift(9) NoteOn(60) NoteOn(65) NoteOn(69)
NoteOn(76) TimeShift(12) NoteOff(60) NoteOff(65)
NoteOff(69) NoteOff(76) TimeShift(3) NoteOff(50)
NoteOn(43) NoteOn(59) NoteOn(65) NoteOn(69) NoteOn(76)
TimeShift(24) NoteOff(All)
```

Figure 4.5 – A bar of music in staff notation (top left), as a piano roll (top right) and as a sequence of tokens (bottom).

- If the input is a piano roll, we use a two-layer convolutional network (CNN) with ELU (exponential linear unit; Clevert et al. [2016]) activation and max pooling, followed by a bidirectional RNN with a gated recurrent unit (GRU; Cho et al. [2014]). The CNN serves to compress the input, resulting in a sequence of 1280-dimensional vectors with 2 vectors per bar. The bidirectional GRU (with 256 units in each direction) then adds the ability to incorporate information from a wider context, producing a sequence of 512-dimensional state vectors $\mathbf{h}_1, \ldots, \mathbf{h}_N$.

- If the input is a sequence of tokens, we use an embedding layer, also followed by a bidirectional GRU.

We refer to the two resulting variants of the model as 'roll2seq' and 'seq2seq', respectively.

The decoder is also implemented using a GRU, conditioned on the target style and equipped with a Bahdanau-style attention mechanism (Bahdanau et al. [2015]; see Section 3.2.1) acting on the encoder outputs $\mathbf{h}_1, \ldots, \mathbf{h}_N$ as in a standard seq2seq model. To condition the decoder on the style embedding $\boldsymbol{\sigma}$, we feed it to the decoder RNN cell as an additional input in every step as in Cho et al. [2014]. The m-th decoder state $\mathbf{s}_m$ is then computed as (cf. Eq. (3.6)):

$$\mathbf{s}_m = \mathrm{GRU}([\mathbf{y}_{m-1}, \mathbf{c}_m, \boldsymbol{\sigma}], \mathbf{s}_{m-1}), \tag{4.1}$$

where $[\cdot]$ denotes concatenation, $\mathbf{y}_{m-1}$ is the embedding of the previous output token, $\mathbf{s}_{m-1}$ is the previous state and $\mathbf{c}_m$ is the context vector computed by the attention mechanism.

The models are trained using Adam [Kingma and Ba, 2015] with learning rate decay and with early stopping on the validation set. Our configuration files with complete hyperparameter settings are included with the source code.

Once the model is trained, we perform style translation using greedy decoding, i.e. by taking the most likely output token at every step as explained in

Section 3.1.1. We also explored random sampling with different softmax temperatures, but found that this leads to a higher number of errors (i.e. invalid sequences or incorrect timing) and does not significantly improve the quality of the outputs.

Data generation details

To generate the synthetic training and test data, we followed the procedure from Section 4.1.1. First, we downloaded chord charts of around 3.5 k songs in the BIAB format from a popular online archive.[6] We used BIAB to generate arrangements of these songs in different styles and filtered the resulting MIDI files to keep only those in $\frac{4}{4}$ or $\frac{12}{8}$ time.[7] We then split those files into segments of 8 bars.

We selected a total of 70 styles from the '0 MIDI' and '1 MIDI' style packs included in Band-in-a-Box 2018, representing a wide variety of popular music genres. Each style contains up to 5 accompaniment tracks, from which we select only the bass and piano track.[8] We generated each song in 3 randomly picked styles, providing $2 \times \binom{3}{2} = 6$ training pairs per segment, or around 658 k training examples in total.

In all experiments, we used 2809 songs for training, 46 songs as a validation set and 46 songs for evaluation, each in 3 examples in different styles.

4.3.2 Experimental results

As mentioned above, we limit ourselves to generating the bass and piano tracks, and we train a dedicated model for each of them. We consider two scenarios: generating the track given only the corresponding source track (BASS→BASS, PIANO→PIANO), and using all non-drum accompaniment tracks from the input (ALL→BASS, ALL→PIANO).

For BASS→BASS, we compare the seq2seq and roll2seq architectures defined in Section 4.3.1. For all other pairs, where the input is non-monophonic, we only employ roll2seq, since the sequential representation grows disproportionately in length in these cases and the computational cost of the attention mechanism becomes too heavy.

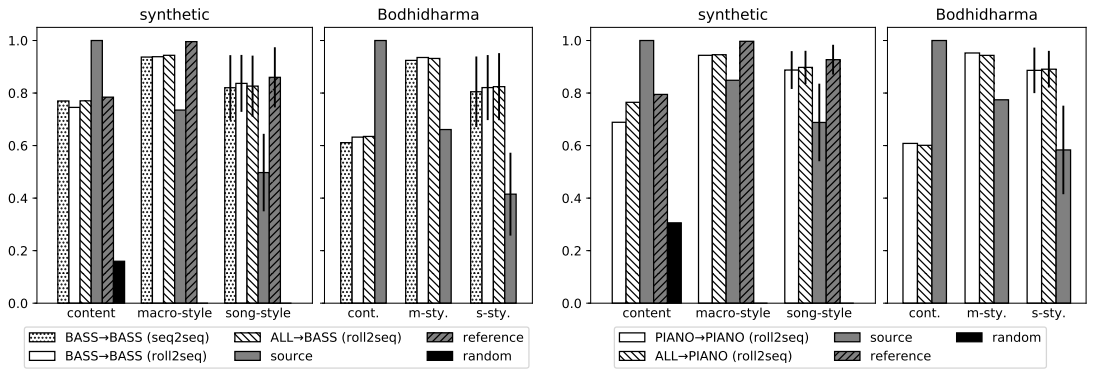
We evaluate our models on our synthetic test set generated by BIAB and on the Bodhidharma MIDI dataset [McKay and Fujinaga, 2005]. The latter dataset was picked on the grounds of being stylistically diverse and balanced, containing an equal number of recordings from 38 different genres. We filtered and pre-processed Bodhidharma in the same way as the synthetic test set and we extracted the bass and piano tracks.

We also made extensive attempts to train the models of Brunner et al. [2018a], Brunner et al. [2018b] and Lu and Su [2018] on our data using the source code published by the authors, but unfortunately without success. This has prevented

[6] Formerly at `https://groups.yahoo.com/group/Band-in-a-Box-Files/`, now moved to `https://band-in-a-box.groups.io/g/main/files`

[7] The time signature depends on the style as well as on the song itself. A song originally in $\frac{4}{4}$ may have a $\frac{12}{8}$ arrangement and vice versa.

[8] The instrument labels are not always accurate; for example, some styles have two guitar tracks, one of which is labeled as piano.



(a) BASS (b) PIANO

Figure 4.6 – Evaluation results on content preservation and style fit. 'Source' is the original track (bass or piano), 'reference' is a track generated by BIAB in the target style and 'random' is a random permutation of the references. For 'song-style', we plot the mean and the standard deviation over all songs and target styles.

us from comparing these models with our proposal. Nonetheless, the provided example outputs can serve as a basis for perceptual comparison.

Evaluation

For a comprehensive evaluation of each model, we translated all inputs to all 70 styles and calculated the content preservation metric and time-pitch style fit metric,[9] the latter in the *macro* and *song-level* variants. The results (averaged) are presented in Fig. 4.6.

We provide two baselines for each track (bass and piano): 'source', which is simply the same track before the translation, and 'reference', which is a track generated by BIAB based on the chord chart (only available for the synthetic test set). As expected, the style fit is low for the source track (measured with respect to the target style) and close to 1 for the reference track. Our models' outputs generally do not fit the target style as perfectly as the reference does, but still score high compared to the source.

As for content preservation, we can notice that the reference value is quite low (0.78 for BASS and 0.79 for PIANO). This should not be too surprising, since we are comparing accompaniments in two different styles, which might have different pitch-class distributions; moreover, there is some random harmonic variation within each style. The results achieved by our models on the synthetic test set are very close to the reference. To illustrate the value range of the metric, we provide the results obtained by a 'randomized' baseline (shown as 'random' in Fig. 4.6), where we randomly permuted the reference segments for each style (obtaining a reference with the correct style, but the wrong content). The resulting value is very low (0.16 for BASS and 0.31 for PIANO) compared both to the true reference and to our models, indicating that the metric is useful and the models are performing well.

[9]We did not use the other metrics as they were not proposed until later in ⟨TASLP2020⟩.

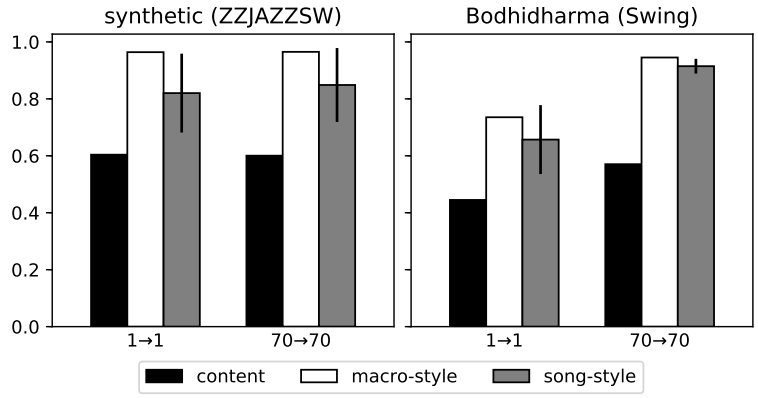


Figure 4.7 – Comparison of a single-style-pair model (1→1) and a full model (70→70) on the ZZJAZZSW→TWIST style pair.

On Bodhidharma, content preservation is generally weaker than on the synthetic test set. One interpretation can be that the encoder simply fails to extract the content information accurately, since it was trained on a different domain. However, we also find that the models often make timing errors on Bodhidharma inputs, leading to misalignment between the input and the output, which may also cause the content preservation metric to drop.

On the other hand, the style fit on Bodhidharma is close to the results on the synthetic test set (and not consistently lower or higher), and the difference to 'source' (i.e. the corresponding input track) is more marked, perhaps reflecting a higher style variability in the Bodhidharma data.

Upon listening, we clearly observe that the outputs are musical and seem to both fit the target style and follow the harmonic structure of the inputs. Besides, even though the piano and the bass tracks are generated independently, they sound surprisingly coherent. However, as mentioned above, we also observe occasional timing errors, which become more prominent when the bass and piano tracks are combined. In the next study, in Section 4.4.1, we will show how this issue can be alleviated by encoding timing in a more robust way. We will also revisit this question later in Section 5.4.

We also note that the single-track models output harmonically incorrect notes more often than the ALL models; this is expected, since their *input* is less harmonically rich. This effect is clearly audible (especially in BASS, where important scale degrees are often missing in the input), but cannot be captured by the content preservation metric, which is computed against the same input.

Comparison with a single-pair model

All models presented so far were trained on music in 70 different styles, as opposed to a single style pair. To investigate the effect of this choice, we picked a pair of fairly dissimilar styles – ZZJAZZSW ('Jazz Swing Variation') and TWIST ('Twist Style', categorized as 'Lite Pop') – and generated a new training, validation and test set with each song rendered in these two styles only. To increase the amount of data, we performed this twice for each song (with different results), obtaining $2 \times 2 = 4$ training pairs per segment.

We used this new dataset to train single-style-pair versions of all models (in

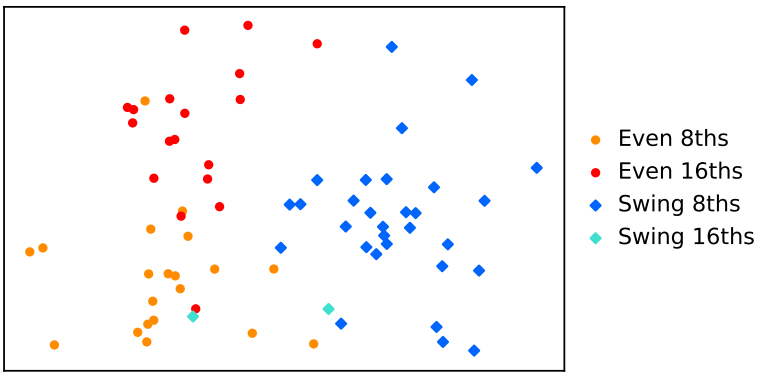


Figure 4.8 – Style embeddings learned by the ALL→PIANO model, labeled with 'feel' annotations provided by BIAB. Dimensionality reduction was performed using linear discriminant analysis (LDA) with the feel labels as targets.

the ZZJAZZSW→TWIST direction only), preserving the original architectures except for the conditioning on the target style. We compare these '1→1' models to the full versions (70→70) on two sets of inputs:

- the synthetic test set in the ZZJAZZSW style;
- the 'Swing' section of Bodhidharma (23 songs).

In Fig. 4.7, we show the results for the two variants of the ALL→BASS model. While the performance on the synthetic data seems to be the same, the scores of the 1→1 model drop considerably on the Bodhidharma data, suggesting that the model is overfitted to the 'synthetic' swing style. On the other hand, the performance of the 70→70 model stays high, showing that training on many different styles helped the model generalize to real swing.

Style embedding analysis

Neural representation spaces are often found to exhibit a meaningful geometry, and we decided to investigate whether this is the case for our learned style embedding space. Fig. 4.8 shows a projection of the embeddings labeled by the 'feel' of each style, with 'even' and 'swing' feel styles being clearly separated. We include more plots in Fig. B.2 in the appendix and also make available an interactive visualization.[10]

4.4 Supervised style transfer (Groove2Groove)

This section details our work on supervised one-shot accompaniment style transfer published in ⟨TASLP2020⟩. This study builds upon the previous one by extending it to the one-shot setting. The result is a system – called *Groove2Groove*[11] (*Grv2Grv* for short) – which is much more practically applicable, since it is not

[10] `https://bit.ly/2G5Jgnq`

[11] 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.

limited to the synthetic styles provided by Band-in-a-Box, but can generalize to novel styles by 'extrapolating' the style of a short example presented at test time.

We created a companion website for this project,[12] gathering all the supplementary materials. These include an interactive demo – allowing to run Groove2Groove on arbitrary MIDI files uploaded by the user –, example videos, source code, and data. See also Appendix C for a complete list of available resources.

4.4.1 Method

The method is largely based on the one proposed in the style translation study in Section 4.3.1. Some of the differences related to dealing with the additional style input were already outlined in Section 4.1 and will be detailed here. We also introduce some improvements based on findings and observations made during the previous study.

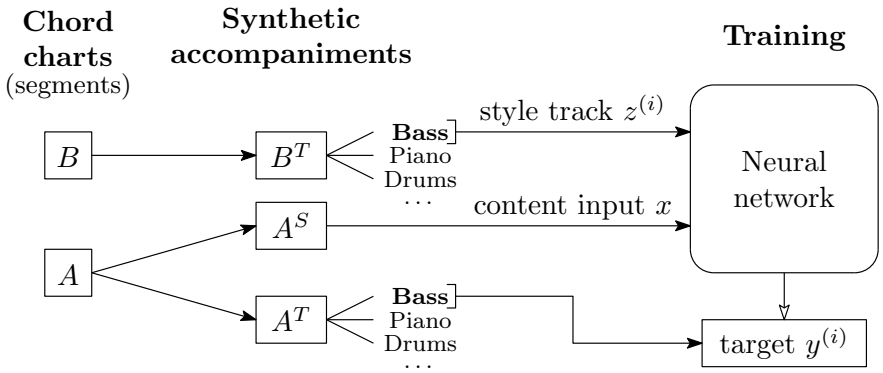
We recall that the main difference between the two methods is that in style translation, the inputs to the models were a content input x and a style label z, while here z becomes a segment of a musical accompaniment, serving as an *example* of the target style – the *style input*. With this comes the need to specify the style encoder architecture and to detail the way the whole model is trained, both of which we will do shortly.

Another major difference, which we have not mentioned yet, lies in how we break down the problem of generating multiple tracks. In the style translation study, we chose to train a separate model for each of the two tracks under consideration (bass, piano). However, in this study, we aim to expand our approach to other instrumentations, so that we may use an arbitrary MIDI file as the style input. In this setting, the said approach is not as suitable, not only considering the number of models that would need to be trained, but also because it may not be clear which track should be fed to which model at test time. For this reason, we propose to use a *single* model for generating all the accompaniment tracks, yet still one at a time.

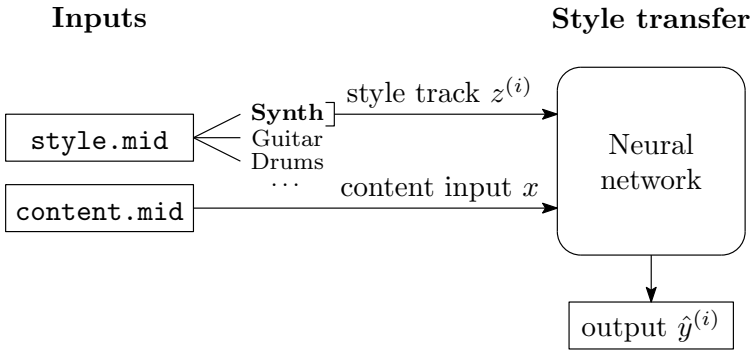
The procedure, illustrated in Fig. 4.9, builds on the idea of the best-performing models from Section 4.3.2 (ALL→BASS and ALL→PIANO). As in these models, we combine all the non-drum tracks from the content input in a single representation x, which is fed to the content encoder regardless of which output track is currently being generated. The style encoder, on the other hand, receives only a single track $z^{(i)}$ of the style input, and the *corresponding* track $y^{(i)}$ of the output is generated by the decoder. Note that neither the index i nor any other information about the track (e.g. the identity of the instrument) is given explicitly. The style encoder therefore does not encode the style of the style input as a whole, but rather the characteristics of the single track at hand.

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. It can also be considered more data-efficient than the model-per-track approach, since a single training segment will result in multiple different parameter updates (one update per target track).

[12] `https://groove2groove.telecom-paris.fr/`



(a) Data generation and training



(b) Inference

Figure 4.9 – An overview of the training and inference procedure for the Groove2Groove system. In both cases, the style input is broken down into individual instrument tracks, which are fed to the model individually as $z^{(i)}$, each resulting in a corresponding output track $y^{(i)}$. On the other hand, the content input x uses a constant representation that contains all tracks (except for drums).

On the other hand, an obvious disadvantage of this approach is that it generates the output tracks conditionally independently given the content input, and therefore cannot model interactions between them. This could be remedied by combining all of the tracks in a single representation as in Donahue et al. [2019a] or as we do later in Section 5.3.1. However, this is computationally more expensive not only due to the sheer length of the input sequences (which scales up with the number of tracks), but also because synchronous generation of multiple tracks is a more difficult task, arguably requiring a more powerful (higher-capacity) model. For this reason, we avoid such approaches in this study.

Input and output representation

Building on the method proposed for style translation, we use a piano roll representation for the content input and a token-based representation for the output. We also use the same token-based representation for the style input.

We keep the piano roll representation from 4.3.1, except that we use the values in the piano roll matrix to encode dynamics. More precisely, we set each value in the matrix to the total velocity of all notes with the given pitch active at the given time, normalized by the maximum velocity (127).

As for the token-based representation, we improve it by proposing a *beat-relative encoding*, which we found to be more robust to timing errors during generation, and the superior performance of which we will demonstrate in the ablation study in Section 4.4.2. Specifically, instead of encoding the time differences between consecutive events, we encode the position of each event within the current beat. To do this, we replace the `TimeShift` tokens with `SetTime` tokens that set the offset within the current beat, plus `SetTimeNext` tokens that allow moving on to the next beat. Moreover, we add `SetVelocity` tokens to be able to represent dynamics, and `DrumOn`/`DrumOff` tokens to encode drum notes.[13] The vocabulary then consists of the following token types:

- `NoteOn(pitch)`, `DrumOn(pitch)`: begin a new note at the given pitch (0–127);
- `NoteOff(pitch)`, `DrumOff(pitch)`: end the note at the given pitch (0–127, or `*` 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).

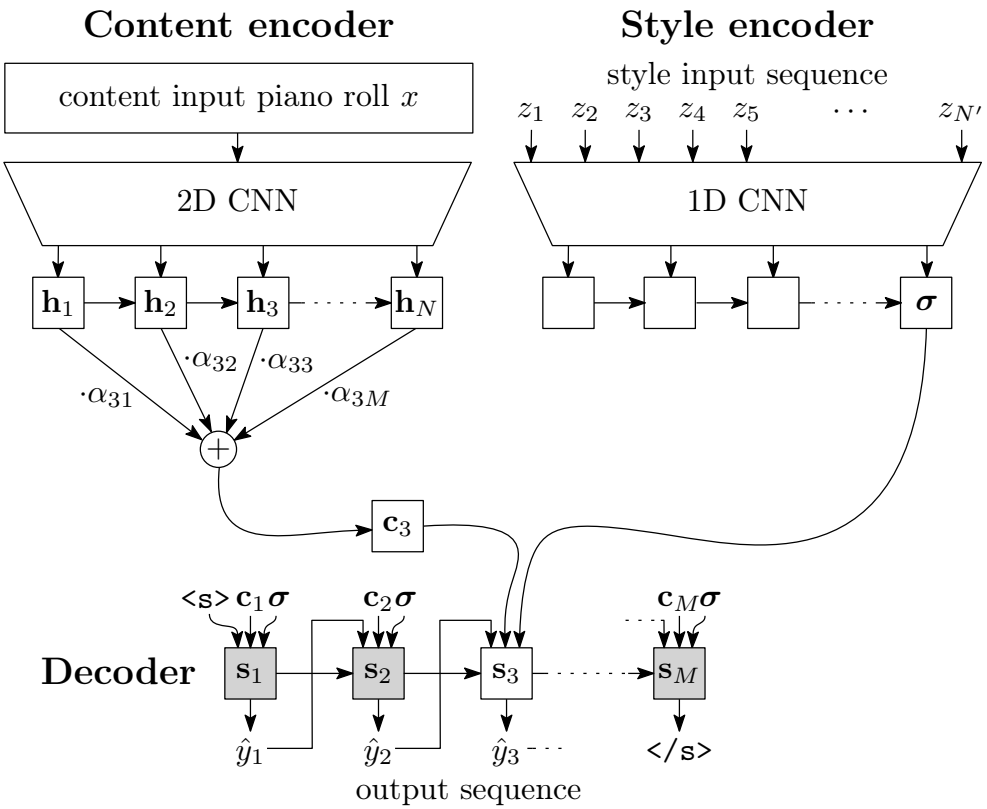


Figure 4.10 – A detailed view of the model architecture with focus on the 3rd decoding step, i.e. $m = 3$ (for illustration purposes). The upper index (i) is omitted everywhere for simplicity. © 2020 IEEE.

Architecture details and training

A detailed view of the architecture is displayed in Fig. 4.10. As in the style translation study, the content encoder consists of a 2D CNN followed by an RNN (GRU); however, we reduce the RNN to a unidirectional (forward) one with a lower number of units in order to limit its memory footprint. The style encoder has a similar architecture, but uses 1D convolutions due to the sequential input representation. More precisely:

- 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 $\mathbf{h}_1, \ldots, \mathbf{h}_N$.

- 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 the style embedding $\boldsymbol{\sigma}^{(i)}$.

Recall that since the style encoder only sees one track $z^{(i)}$ at a time, the style embedding $\boldsymbol{\sigma}^{(i)}$ 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 decoder architecture is identical to the one from the style translation study.

We train the model using Adam [Kingma and Ba, 2015] with a batch size of 64 and with exponential learning rate decay, halving the learning rate every 3 k batches (192 k training triplets). 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.

Data generation details

As in Section 4.3, we need a parallel dataset of accompaniments in different styles, in this case one that allows us to gather *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. While we could extract such triplets from the dataset presented in Section 4.3.1, they would not be suitable for this task for two reasons. First, using only 70 styles would certainly not be enough for generalization to novel *target* styles. Second, for this task, the training, validation and testing sections of the dataset must contain disjoint sets of styles, so that we may monitor and evaluate the performance of the model in the one-shot setting.

[13] In drum tracks, the MIDI note number (which normally represents pitch) identifies the percussion instrument (e.g. snare drum, hi-hat). For this reason, we chose to use a separate set of tokens for drums; this way, we also avoid having to explicitly inform the model that the track is to be played by a drum kit, and not a pitched instrument.

We therefore decided to generate a completely new dataset using the following procedure.

The first step was again to acquire chord charts. Instead of using the same chord charts as in Section 4.3.1, we chose to create a new set of *synthetic* ones. The main, purely practical motivation was 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 enabled us to release it publicly for reproducibility and to foster future research on musical styles.

We obtained the chord charts by sampling from a chord language model (LM) estimated on the iRb corpus [Broze and Shanahan, 2013], which contains chord charts of over a thousand jazz standards. This process resulted in 1200 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. Details of this procedure can be found in Appendix A.1.

Finally, to generate the accompaniments, we picked 1476 MIDI styles from BIAB (their list can be found on the supplementary website) and reserved $20+20$ 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 study, 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.

4.4.2 Experimental results

As in the style translation study, we test the model on our synthetic test set and the Bodhidharma dataset. Additionally, we subjected Bodhidharma to a 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. 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. This is the same as the 'source' baseline in the style translation study.

- CP-STYLE: copies the style input to the output; expected to have perfect performance on style fit, but poor on content preservation (similarly to the 'random' baseline in the style translation study, since the style input is chosen independently of the content input).

- 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 (including our own style translation study) is unfortunately not possible. The main reason is that a style translation 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 and an ablation study, provide some observations about practical use of the proposed system, evaluate the proposed style similarity metrics, and explore the properties of the style embedding space.

Evaluation results

We evaluate our model using the metrics described in Section 4.2, computed in the *macro* and *nano* versions. First, we present in Fig. 4.11 the results on the synthetic test set.

In terms of the content preservation metric, Groove2Groove achieves perfect results (on par with ORACLE), and the gap with respect to CP-STYLE is large. Even though on style fit metrics (*macro* version), Groove2Groove 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 Groove2Groove 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

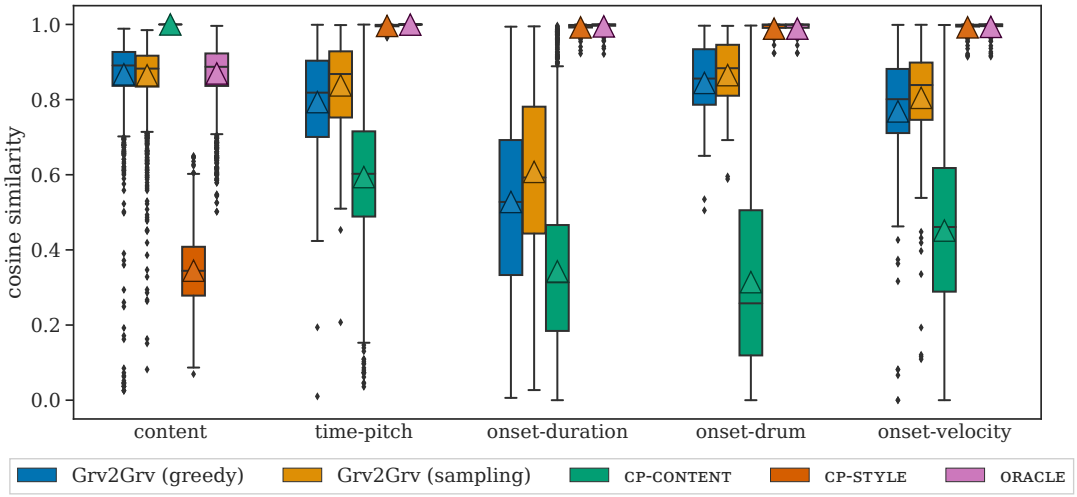


Figure 4.11 – Evaluation results on the synthetic test set. The two leftmost results in each group are those of our main proposed model. The triangles indicate the mean. We use the *macro* variant of the style metrics, i.e. each data point corresponds to one of the target styles. © 2020 IEEE.

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 4.2). To allow for comparison, we compute the *nano* metrics on both datasets (synthetic and Bodhidharma) and display the results alongside in Fig. 4.12. First of all, we can see the style metrics drop and become more 'blurred' with respect to their *macro* versions (Fig. 4.11). For example, on the synthetic dataset, ORACLE decreases from 1.00 to 0.75 on average on time-pitch, and Groove2Groove (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, 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, Groove2Groove – being trained on accompaniments – will instead attempt to generate a pseudo-accompaniment track in the style of the melody. Such tracks are generally

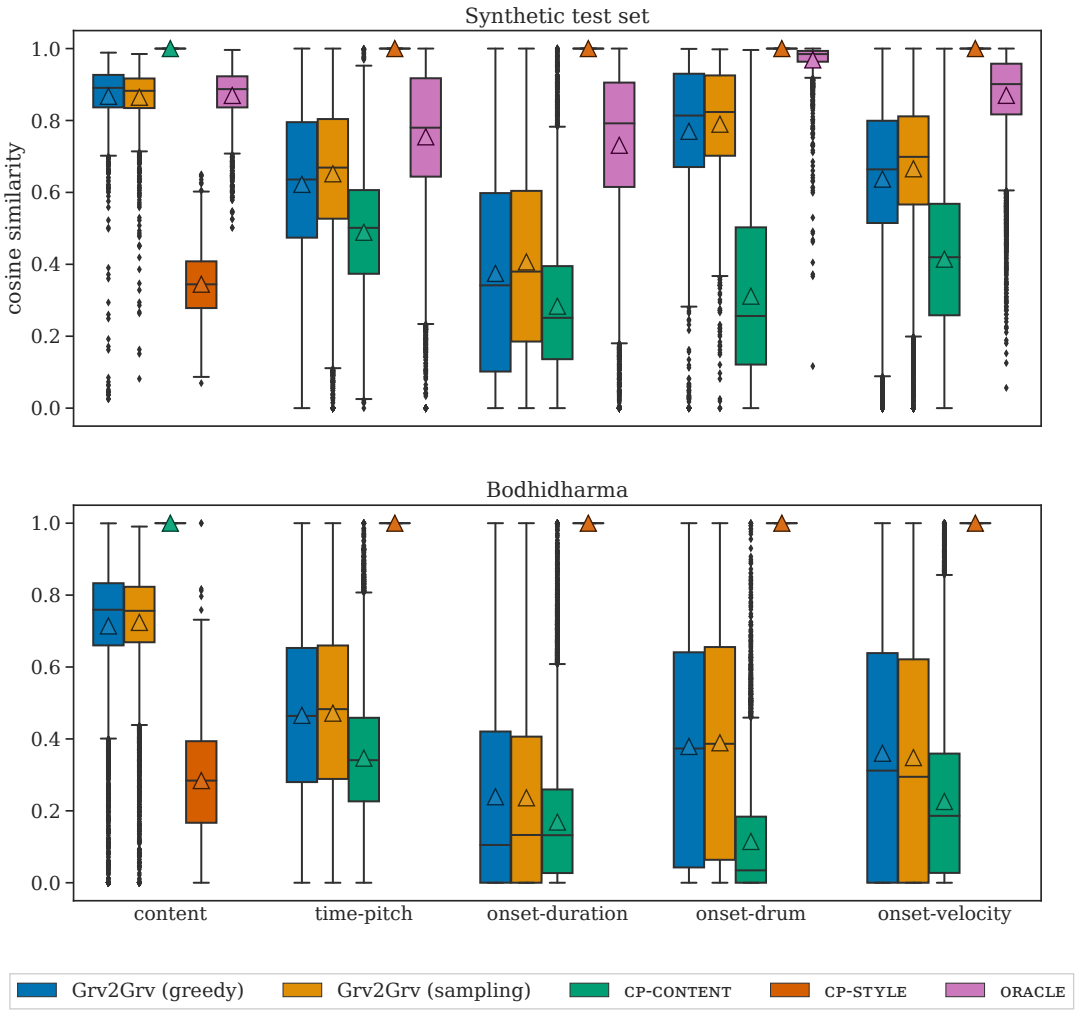


Figure 4.12 – Evaluation results on both test sets. Style metrics are computed in the *nano* variant, i.e. each data point corresponds to a single example. This results in higher variance than in Fig. 4.11, but enables us to evaluate on Bodhidharma.

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.

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. 4.11.

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.

Ablation study

Compared to our style translation work, Groove2Groove 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.

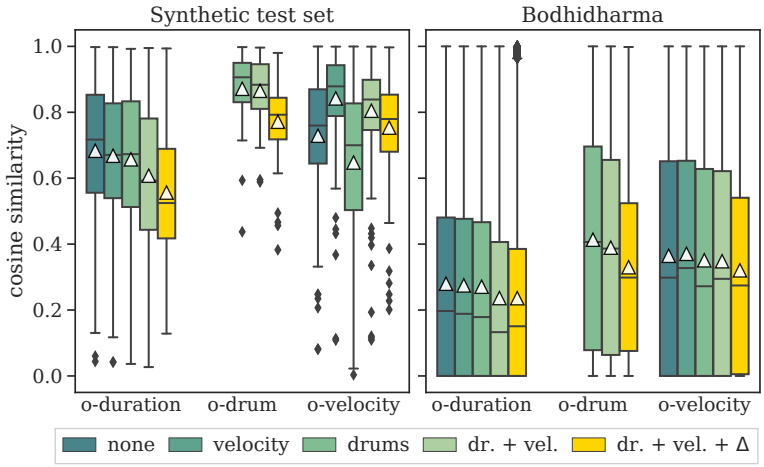


Figure 4.13 – Results of the ablation study. 'Dr. + vel.' ('drums + velocity') is the full Groove2Groove model; 'none' models neither drums nor velocity. Δ stands for the Δ-encoding. All models are evaluated in sampling mode. The synthetic test set uses *macro* metrics as in Fig. 4.11, Bodhidharma uses *nano* metrics as in Fig. 4.12. © 2020 IEEE.

Fig. 4.13 (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).

We are also interested in validating our proposed beat-relative encoding (see Section 4.4.1), designed to overcome the limitations of representing timing as time differences between consecutive events. For this reason, we include in our ablation study a version of the Groove2Groove model which employs this latter strategy, which we will refer to as the Δ-*encoding*. In practice, this means that all `SetTime` and `SetTimeNext` tokens are replaced with `TimeShift` tokens.

The results, displayed in Fig. 4.13 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 already observed in Section 4.3.2). 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.

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 $\boldsymbol{\sigma}_0, \boldsymbol{\sigma}_1$, we conditioned the decoder, in turn, on vectors of the form $(1-\alpha)\cdot\boldsymbol{\sigma}_0 + \alpha\cdot\boldsymbol{\sigma}_1$ 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 $\boldsymbol{\sigma}_0, \boldsymbol{\sigma}_1$ (i.e. for $\alpha = 0, 1$).

Fig. 4.14 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. 4.14 then displays the behavior on average, showing that, consistently with the above observations, the similarity to the initial style decreases with increasing α.

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

Style embedding visualization

To further explore the properties of the style embedding space, we visualize in Fig. 4.15 embeddings of segments from the Bodhidharma dataset, using PCA followed by t-SNE [Maaten and Hinton, 2008] 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. Even though there are no pronounced clusters, we can observe that the individual genres are fairly localized.

4.5 Conclusion

We have proposed an approach to accompaniment style conversion based on supervised learning from synthetic data, focusing first in Section 4.3 on the accompaniment style *translation* task. Although restricted to bass-and-piano accompaniments in 70 selected styles, it has allowed us to demonstrate that with the right kind of training data, supervised machine translation methods can be applied to music with success, even generalizing to inputs in styles unseen during training.

We then extended this approach in Section 4.3 to address our ultimate goal of *one-shot style transfer* while mitigating some of the flaws observed in the style translation experiments.

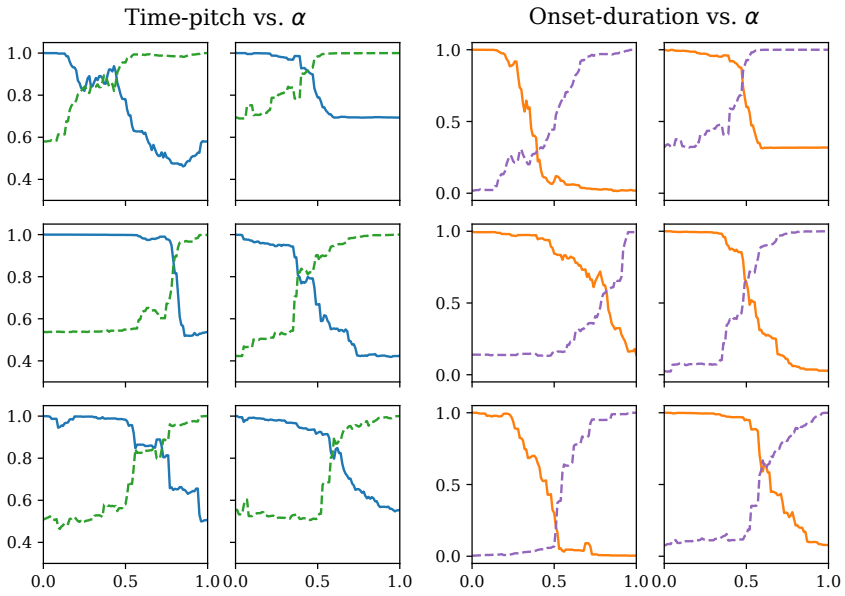
Nevertheless, some limitations remain. Possibly the most important shortcoming of our one-shot style transfer system, Groove2Groove, is the fact that it is limited to accompaniments and does not account for interaction between

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 partial 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. by Roberts et al. [2018], Payne [2019], Thickstun et al. [2019]), albeit more computationally demanding, as already discussed in Section 4.4.1.

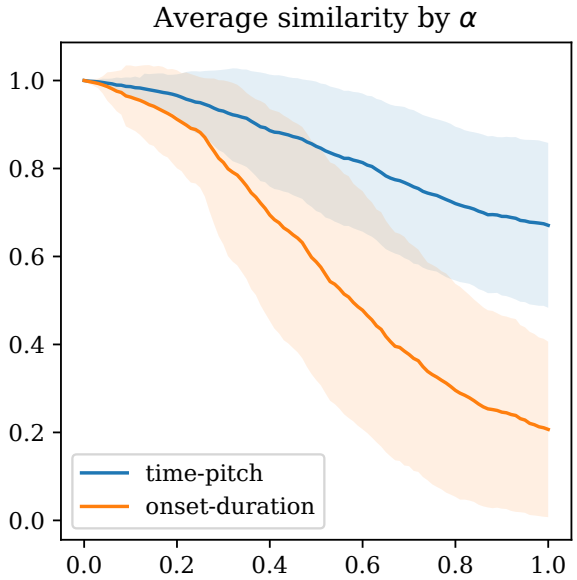
In addition to inter-track ('vertical') dependencies, long-range temporal ('horizontal') dependencies are also of great importance in music generation. The patterns used in a musical piece typically do not stay the same over the duration of the piece, but vary between sections. To model such dependencies, it is necessary to consider the context of the whole piece instead of only 8 measures at a time, which constitutes another scaling problem.

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 suboptimal 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.

Lastly, the applicability of Groove2Groove 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.

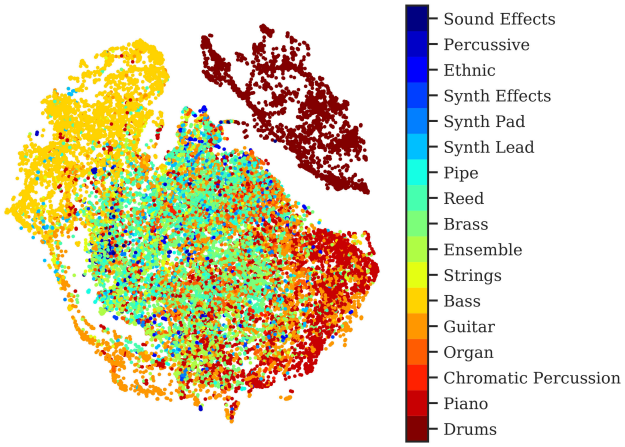


(a) Examples for specific style pairs; the solid line and the dashed line show the similarity to the outputs generated from $\boldsymbol{\sigma}_0$ (i.e. for $\alpha = 0$) and $\boldsymbol{\sigma}_1$ (for $\alpha = 1$), respectively.

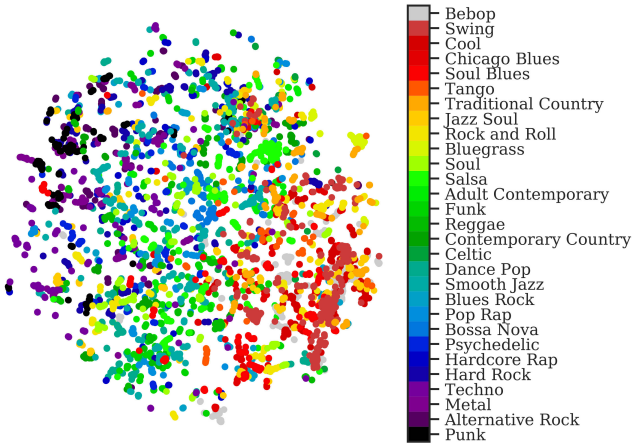


(b) α-wise average and standard deviation of the metrics plotted in (a) (solid line) over 100 style pairs.

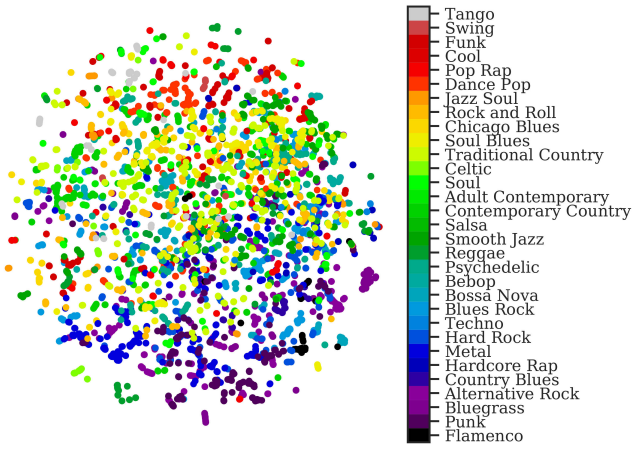
Figure 4.14 – Style similarity to interpolation endpoints as a function of the interpolation coefficient α. © 2020 IEEE.



(a) All embeddings by instrument



(b) Bass embeddings by genre



(c) Guitar embeddings by genre

Figure 4.15 – 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. © 2020 IEEE.

5. Positional encodings for music generation

Sections 5.1.1, 5.1.2, 5.2 and 5.3 in this chapter are based on the paper ⟨**ICML2021**⟩ by Liutkus, Cífka (equal contribution) et al.

In Chapter 4, we identified the need to develop models capable of jointly generating multiple tracks in order to capture inter-track dependencies, as well as handling longer temporal contexts. This involves scaling up the input size *and* model capacity, a difficult challenge and an interesting problem in itself. One of the obstacles is that more powerful models like Transformers, which have a better ability to model complicated dependencies and have recently become the tool of choice for music generation, also come with increased computational complexity, which becomes prohibitive for very long sequences.

For this reason, we make a detour from style transfer in this chapter in order to study **music generation** using the recently proposed **linear complexity Transformers** (see Section 5.1.1 below) with particular focus on **positional encoding** schemes.

After some background (Section 5.1), we present in Section 5.2 our main, theoretical contribution of this chapter: *stochastic positional encoding* (*SPE*), a novel positional encoding (PE) scheme that addresses a fundamental incompatibility between linear complexity Transformers and relative positional encoding (RPE). We experimentally validate the proposed technique on music generation in Section 5.3 (with additional experiments on non-music tasks included in Appendix A.2). In Section 5.4, we follow up with some preliminary results on how PE may be better exploited for music generation by making it encode more musically meaningful information. Section 5.5 concludes the chapter.

5.1 Background

5.1.1 Linear complexity Transformers

The Transformer model [Vaswani et al., 2017] has been described in Section 3.1.2. We recall here Eqs. (3.4) and (3.5), expressing the computation of Transformer (self-)attention from queries $\mathbf{Q}$, keys $\mathbf{K}$ and values $\mathbf{V}$:

$$\mathrm{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \left[\frac{\sum_{n=1}^{N}\exp(e_{mn})\mathbf{v}_n}{\sum_{n'=1}^{N}\exp(e_{mn'})}\right]_m,$$
$$e_{mn} = \mathbf{q}_m^\top \mathbf{k}_n \big/ \sqrt{D}\,.$$

Defining the (unnormalized) *attention matrix* $\mathbf{A} = [a_{mn}]_{mn}$ as

$$\mathbf{A} \equiv [\exp(e_{mn})]_{mn} = \exp\Big(\mathbf{Q}\mathbf{K}^\top \big/ \sqrt{D}\Big), \tag{5.1}$$

we can rewrite Eqs. (3.4) and (3.5) as

$$\mathrm{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathbf{S}^{-1}\mathbf{A}\mathbf{V}, \tag{5.2}$$
$$\mathbf{S} = \mathrm{diag}(\mathbf{A}\mathbf{1}_N), \tag{5.3}$$

where $\mathbf{1}_N$ is an all-ones vector of length N. The multiplication by $\mathbf{S}^{-1}$ is simply a way to express row-wise normalization (division by row sums) of $\mathbf{A}$.

Tsai et al. [2019] and Choromanski et al. [2021] introduce a generalization where $\mathbf{A}$ is expressed in terms of a *kernel function* $\mathcal{K}\colon \mathbb{R}^M \times \mathbb{R}^N \to \mathbb{R}_+$:

$$\mathbf{A} = [\mathcal{K}(\mathbf{q}_m, \mathbf{k}_n)]_{mn}\,. \tag{5.4}$$

In the original Transformer [Vaswani et al., 2017], we have, according to Eq. (5.1):

$$\mathcal{K}(\mathbf{q}, \mathbf{k}) = \exp\Big(\mathbf{q}^\top \mathbf{k} \Big/ \sqrt{D}\Big). \tag{5.5}$$

Other choices of $\mathcal{K}$ are possible, yielding different variants of the Transformer as in Choromanski et al. [2021] and Katharopoulos et al. [2020].

The original Transformer architecture explicitly computes the attention matrix $\mathbf{A}$, leading to a $\mathcal{O}(MND)$ complexity (*quadratic* in the case of self-attention, where $M = N$), which prevents it from scaling to very long sequences. Although this is not necessarily a problem when sequence lengths are barely on the order of a few hundreds, as in some language processing tasks, it is prohibitive for very large signals like high-resolution images or audio.

Focusing on this scalability issue, several approaches have been recently investigated to allow for long sequences:

- *Attention clustering* schemes group items among which dependencies are computed through regular attention. This is either done by using simple proximity rules within the sequences, leading to chunking strategies [Dai et al., 2019], or by clustering the keys and values [Roy et al., 2021]. Inter-cluster dependencies are either ignored or summarized via fixed-length context vectors that are coined in as *memory* [Wu et al., 2020].

- Assuming the attention matrix to be *sparse*. In this case, only a few a_{mn} are nonzero [Child et al., 2019].

- Assuming $\mathbf{A}$ has a particular (low-rank) *structure* and can be decomposed as the product of two smaller matrices. A prototypical example is the Linformer [Wang et al., 2020a], which is limited to fixed-length inputs. Another very recent line of research in this same vein takes:

  $$\mathbf{A} \approx \phi(\mathbf{Q})\phi(\mathbf{K})^\top, \tag{5.6}$$

  where $\phi : \mathbb{R}^D \to \mathbb{R}^R$ is a non-linear *feature map* applied to each key $\mathbf{k}_n$ and query $\mathbf{q}_m$, and $R \ll \min(M, N)$ [Shen et al., 2021, Katharopoulos et al., 2020].

- When $\mathcal{K}$ in Eq. (5.4) is a positive (semi-)definite kernel, the Performer [Choromanski et al., 2021] leverages *reproducing kernel Hilbert spaces* to show that a random ϕ may be used to exploit this convenient decomposition (5.6) *on average*, even when $\mathbf{A}$ is not low rank:

  $$\mathcal{K} \succeq 0 \Leftrightarrow \mathbf{A} = \mathbb{E}_\phi\left[\phi(\mathbf{Q})\phi(\mathbf{K})^\top\right], \tag{5.7}$$

  where ϕ is drawn from a distribution chosen based on $\mathcal{K}$. A simple example is $\phi_{\mathbf{W}}(\mathbf{k}_n) = \max(0, \mathbf{W}\mathbf{k}_n)$, with a random $\mathbf{W} \in \mathbb{R}^{R\times D}$ for some $R \in \mathbb{N}$.

In schemes like (5.6) and (5.7), the improved time and space complexity comes from the fact that the outputs can be obtained efficiently without computing the attention coefficients a_{mn}:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) \approx \mathbf{S}^{-1} \left[\phi(\mathbf{Q}) \left[\phi(\mathbf{K})^\top \mathbf{V}\right]\right], \tag{5.8}$$

$$\mathbf{S} = \text{diag}\Big(\phi(\mathbf{Q}) \left[\phi(\mathbf{K})^\top \mathbf{1}_N\right]\Big). \tag{5.9}$$

Notice that the multiplication $\phi(\mathbf{K})^\top \mathbf{V}$ has time complexity $\mathcal{O}(NRD)$ and the result has dimensions $R \times D$ (independent of the input length). The whole attention mechanism then has time complexity $\mathcal{O}((M+N)\,RD)$ instead of $\mathcal{O}(MND)$ and space complexity $\mathcal{O}(MR + NR + ND + RD)$ instead of $\mathcal{O}(MN + (M+N)\,D)$. Hence, we avoid the MN term and obtain a *linear complexity Transformer*.

5.1.2 Relative positional encoding

Relative positional encoding (*RPE*, Shaw et al. [2018]) has already been mentioned in Section 3.1.2. The corresponding attention matrix can be written as follows:

$$\mathbf{A} = \exp\Big(\Big(\mathbf{Q}\mathbf{K}^\top + \mathbf{\Omega}\Big)\Big/\sqrt{D}\Big), \tag{5.10}$$

$$\mathbf{\Omega} = \left[\sum_{d=1}^{D} q_{md} \mathcal{P}_d(m-n)\right]_{mn}, \tag{5.11}$$

where $\mathcal{P}_d \colon \mathbb{Z} \to \mathbb{R}$ is the d-th component – activated by query dimension d – of the (learnable) relative positional encoding.

Although RPE is beneficial for performance [Shaw et al., 2018, Dai et al., 2019, Tsai et al., 2019], we are only aware of implementations that either require the computation of $\mathbf{A}$, and clustered attention schemes, which *in fine* decompose $\mathbf{A}$ into smaller attention matrices, and *compute them*. This is in sharp contrast to Eqs. (5.8) and (5.9), which never compute the attention matrix.

5.1.3 Music generation with Transformers

Recent years have seen increasing amounts of work on applying Transformers to music generation [Huang et al., 2019a, Payne, 2019, Donahue et al., 2019a, Jiang et al., 2020, Huang and Yang, 2020, Wu and Yang, 2020, Muhamed et al., 2021], largely motivated by the well-known ability of the self-attention mechanism to model long-range dependencies. To make scaling to longer sequences computationally feasible, some works [Donahue et al., 2019a, Huang and Yang, 2020, Wu and Yang, 2020] have adopted *Transformer-XL* [Dai et al., 2019], a variant of the Transformer enhanced with a recurrence mechanism. This model allows training on arbitrarily long sequences, but only a fixed-length past context is available through the self-attention mechanism. Due to this property, it also needs to be used together with relative attention (RPE).

MIDI-like token-based representations, such as the one used in Section 4.3, seem to be the most popular output format [Huang et al., 2019a, Payne, 2019, Donahue et al., 2019a]. Huang and Yang [2020] observe the same issues with regular rhythms as we did in Section 4.3.2, and propose *REMI* (*revamped MIDI-derived format*). In a similar spirit to our *beat-relative encoding* from Section 4.4.1,

REMI has `Bar` events that mark a downbeat (the start of a new bar) and `Position` events that indicate the offset from the last downbeat.

Regarding PE, the vast majority of works use absolute (APE) or relative (RPE) positional encoding based on token indices as usual. Huang et al. [2019a] employ a generalized, *relation-aware* variant of RPE [Shaw et al., 2018], encoding the time and pitch difference between two notes in addition to the difference between token indices. This is found to improve performance, but comes with an increased computational cost; this is likely why the authors only use this technique in the first layer of the network. To our knowledge, this is the only existing music generation work where PE is based on other kinds of information than solely token indices.

5.2 Stochastic positional encoding – theory

We now introduce our *stochastic positional encoding* (*SPE*), a novel positional encoding scheme designed to generalize RPE (Section 5.1.2 above) while being compatible with linear complexity Transformers (Section 5.1.1 above). A key idea for SPE is to draw a connection between positional encoding and *cross-covariance* structures of *correlated random processes*, as we will explain in the following. This will allow us to design a way to augment keys and queries in the attention mechanism so that a given relative positional pattern arises when they are multiplied together.

Index set. For generality, let us consider input/output sequences indexed by $n \in \mathcal{N}$ and $m \in \mathcal{M}$, respectively, where $\mathcal{N}, \mathcal{M} \subseteq \mathbb{T}$ and $\mathbb{T}$ is called the *index set*. As before, the sequence lengths are denoted $N = |\mathcal{N}|$ and $M = |\mathcal{M}|$.

Note that in all our experiments, we will work with regularly sampled sequences, i.e. $\mathbb{T} = \mathbb{N}$, but more settings are possible, such as irregularly sampled time series ($\mathbb{T} = \mathbb{R}$) or images ($\mathbb{T} = \mathbb{N}^2$). Also, we will only operate in the *self-attention* setting (as opposed to *encoder-decoder attention*), where the 'input' and 'output' sequence are the same ($\mathcal{N} = \mathcal{M} = \{1, \dots, N\}$) and the indices n, m simply pertain to keys $\{\mathbf{k}_n\}_{n\in\mathcal{N}}$ and queries $\{\mathbf{q}_m\}_{m\in\mathcal{N}}$, respectively, computed based on this sequence.

Assumptions. We seek an attention matrix $\mathbf{A}$ given by:

$$\mathbf{A} = \exp\left(\left[\sum_{d=1}^{D} q_{md}\mathcal{P}_d(m,n)k_{nd}\right]_{mn} \Big/ \sqrt{D}\right), \tag{5.12}$$

where $\{\mathcal{P}_d\}_{d=1}^{D}$ are *position kernels*. Defining $\mathbf{P}_d \equiv [\mathcal{P}_d(m,n)]_{mn}$, this can be written in matrix form as:

$$\mathbf{A} = \exp\left(\sum_{d=1}^{D} \operatorname{diag}(\mathbf{q}_{*d})\mathbf{P}_d \operatorname{diag}(\mathbf{k}_{*d}) \Big/ \sqrt{D}\right), \tag{5.13}$$

which is understood as having D positional attention templates $\mathbf{P}_d$ jointly activated by the queries $\mathbf{q}_{*d}$ and keys $\mathbf{k}_{*d}$. Note that while original RPE in Eqs. (5.10) and (5.11) is different from this formulation, it can be seen as a special case where some entries are kept constant.

Positional attention as covariance. The key idea for SPE is to see the position kernel $\mathcal{P}_d(m,n)$ as a *covariance*:

$$(\forall \mathcal{M}, \mathcal{N})\,(\forall m,n)\;\mathcal{P}_d(m,n) = \mathbb{E}\left[\bar{Q}_d(m)\bar{K}_d(n)\right], \tag{5.14}$$

where $\bar{Q}_d(m)$ and $\bar{K}_d(n)$ are two real-valued random variables, which will be chosen with the single condition that their covariance function matches $\mathcal{P}_d$. Semantically, they should be understood as (randomly) encoding position m for queries and position n for keys, respectively. When multiplied together as in dot-product attention, they yield the desired attention template $\mathcal{P}_d(m,n)$ on average. The central intuition is that the actual positional encodings do not matter as much as their dot product.

In what follows, we will impose specific structures on the cross-covariance $\mathcal{P}_d(m,n)$, which will allow us to design *random processes* $\bar{Q}_d = \{\bar{Q}_d(m)\}_{m\in\mathcal{M}}$ and $\bar{K}_d = \{\bar{K}_d(n)\}_{n\in\mathcal{N}}$ such that (5.14) holds. The core advantage of this construction is to allow for $\mathbf{P}_d$ to be factorized. Let us for now assume that we construct $\bar{Q}_d, \bar{K}_d$ in such a way that we can sample from them (we will see how in Section 5.2.1) and consider R independent realizations of them for given $\mathcal{M}$ and $\mathcal{N}$, gathered in the $M \times R$ and $N \times R$ matrices $\bar{\mathbf{Q}}_d$ and $\bar{\mathbf{K}}_d$, respectively:

$$\bar{\mathbf{Q}}_d \sim [\bar{Q}_d(m)]_{mr},\; \bar{\mathbf{K}}_d \sim [\bar{K}_d(n)]_{nr}. \tag{5.15}$$

For large R, by the law of large numbers, we obtain:

$$\mathbf{P}_d \approx \bar{\mathbf{Q}}_d \bar{\mathbf{K}}_d^\top \big/ R. \tag{5.16}$$

This leads to $\mathbf{A}$ in Eq. (5.13) being given by:

$$\mathbf{A} \approx \exp\left(\sum_{d=1}^{D} \operatorname{diag}(\mathbf{q}_{*d}) \frac{\bar{\mathbf{Q}}_d \bar{\mathbf{K}}_d^\top}{R} \operatorname{diag}(\mathbf{k}_{*d}) / \sqrt{D}\right) \tag{5.17}$$

$$\approx \exp \frac{\left(\sum_{d=1}^{D} \operatorname{diag}(\mathbf{q}_{*d}) \bar{\mathbf{Q}}_d\right)\left(\sum_{d=1}^{D} \operatorname{diag}(\mathbf{k}_{*d}) \bar{\mathbf{K}}_d\right)^\top}{R\sqrt{D}}. \tag{5.18}$$

Here, a crucial observation is that for large R, the cross-terms $\bar{\mathbf{Q}}_d \bar{\mathbf{K}}_{d'\neq d}^\top$ are negligible due to independence, *provided that the processes are zero-mean.* Finally, picking queries and keys as:

$$\hat{\mathbf{Q}} \leftarrow \sum_{d=1}^{D} \operatorname{diag}(\mathbf{q}_{*d}) \bar{\mathbf{Q}}_d \big/ \sqrt[4]{DR}, \tag{5.19}$$

$$\hat{\mathbf{K}} \leftarrow \sum_{d=1}^{D} \operatorname{diag}(\mathbf{k}_{*d}) \bar{\mathbf{K}}_d \big/ \sqrt[4]{DR}, \tag{5.20}$$

we see from Eqs. (5.18) to (5.20) that we get back to the usual multiplicative scheme (5.4) with $\mathbf{A} = \exp(\hat{\mathbf{Q}}\hat{\mathbf{K}}^\top/\sqrt{R})$, where the queries/keys now have dimension R and can be used in Eq. (5.8) to directly obtain the outputs without computing $\mathbf{A}$.

The procedure is summarized in Algorithm 1: we provide a way (5.19–5.20) to achieve PE in the *keys domain*, such that the desired model (5.12) is enforced in the *attention domain*, parameterized by the attention kernels $\mathcal{P}_d$. Interestingly, this is done without ever computing attention matrices, complying with linear complexity Transformers. The remaining challenge, which we discuss next, is to generate $\bar{\mathbf{Q}}_d$ and $\bar{\mathbf{K}}_d$ enforcing Eq. (5.16).

Algorithm 1 Stochastic Positional Encoding.

Input:

- position kernel $\mathcal{P}(m, n)$, number of replicas R.
- initial $M \times D$ and $N \times D$ queries $\mathbf{Q}$ and keys $\mathbf{K}$.

Positional encoding:

- Draw the D independent tuples $\{(\bar{\mathbf{Q}}_d, \bar{\mathbf{K}}_d)\}_d$ of $M \times R$ and $N \times R$ matrices as in Section 5.2.1.
- Set $\hat{\mathbf{Q}}$ and $\hat{\mathbf{K}}$ as in Eqs. (5.19) and (5.20).

Inference: Compute outputs $\mathbf{Y}$ with a linear complexity Transformer as in Eqs. (5.8) and (5.9), i.e.:

$$\mathbf{S} \leftarrow \mathrm{diag}\Big(\phi(\mathbf{Q})\left[\phi(\mathbf{K})^\top \mathbf{1}_N\right]\Big), \quad \mathbf{Y} \leftarrow \mathbf{S}^{-1}\left[\phi(\mathbf{Q})\left[\phi(\mathbf{K})^\top \mathbf{V}\right]\right].$$

5.2.1 Drawing stochastic positional encodings

Inspecting (5.14), we notice that our objective is to draw samples from D pairs of centered random processes $\{\bar{Q}_d, \bar{K}_d\}_d$ with a prescribed cross-covariance structure $\mathcal{P}_d$. It is reasonable to use Gaussian processes for this purpose [Williams and Rasmussen, 2006], which have the maximum entropy for known mean and covariance. Such distributions are frequently encountered in geophysics in the *co-kriging* literature [Matheron, 1963, Genton and Kleiber, 2015], where scientists routinely handle correlated random fields. The particular twists of our setup are: we have a *generative* problem, e.g. as in Vořechovský [2008]; however, as opposed to their setting, we are not directly interested in the marginal covariance function of each output, provided that the desired cross-covariance structure holds.

The most straightforward application of SPE arises when we pick $\mathcal{P}_d(m, n) = \mathcal{P}_d(m - n)$, i.e. a stationary position kernel, which was coined in as choosing *relative* attention in Shaw et al. [2018] and boils down to enforcing a *Toeplitz* structure for the cross-covariance matrix $\mathbf{P}_d \equiv [\mathcal{P}_d(m - n)]_{m,n}$ between $\bar{Q}_d$ and $\bar{K}_d$.

We propose two variants of SPE to handle this important special case, illustrated in Fig. 5.1. The first variant yields *periodic* covariance functions. It can be beneficial whenever attention should not vanish with large lags, e.g. as in traffic prediction [Xue and Salim, 2020]. The second variant generates *vanishing* covariance functions; a concept which has recently been shown useful [Wang et al., 2021].

Variant I. Periodic attention (`sineSPE`). In our first approach, we consider the case where $\mathcal{P}_d$ is periodic, which gets a convenient treatment. We assume:

$$\mathcal{P}_d(m,n) = \sum_{k=1}^{K} \lambda_{kd}^2 \cos(2\pi f_{kd}\,(m-n) + \theta_{kd})\,, \tag{5.21}$$

where $K \in \mathbb{N}$ is the number of *sinusoidal* components and $\mathbf{f}_d \in [0\ 1]^K$, $\boldsymbol{\theta}_d \in [-\pi\ \pi]^K$ and $\boldsymbol{\lambda}_d \in \mathbb{R}^K$ gather their K frequencies, phases, and weights, respectively. By using the matrix notation, we can rewrite (5.21) as:

$$\mathbf{P}_d = \boldsymbol{\Omega}(\mathcal{M}, \mathbf{f}_d, \boldsymbol{\theta}_d)\,\mathrm{diag}\big(\ddot{\boldsymbol{\lambda}}_d\big)^2 \boldsymbol{\Omega}(\mathcal{N}, \mathbf{f}_d, \mathbf{0})^\top, \tag{5.22}$$

where $\ddot{\mathbf{v}} \equiv \Big[v_{\lfloor p/2 \rfloor}\Big]_p \in \mathbb{R}^{2K}$ denotes a twice upsampled version of a vector $\mathbf{v} \in \mathbb{R}^K$, $\lfloor \cdot \rfloor$ denotes the floor operation, and for an index set $\mathcal{I}$, $\boldsymbol{\Omega}(\mathcal{I}, \mathbf{a}, \mathbf{b})$ is a matrix of size $|\mathcal{I}| \times 2K$, with entries (0-based indexing):

$$\left[\boldsymbol{\Omega}\left(\mathcal{I}, \mathbf{a}, \mathbf{b}\right)\right]_{nl} = \begin{cases} \cos(2\pi a_k n + b_k) & \text{if } l = 2k \\ \sin(2\pi a_k n + b_k) & \text{if } l = 2k+1 \end{cases}$$

It can be shown that if $\boldsymbol{\theta}_d = \mathbf{0}$ and $\mathcal{M} = \mathcal{N}$, we get back to the (unique) Vandermonde decomposition for positive definite Toeplitz matrices[1] [Yang et al., 2016], which boils down in our context to assuming that $(\forall \tau)\ \mathcal{P}_d(0) \geq \mathcal{P}_d(\tau)$. Since this is not always desirable, we keep the more general (5.22).

At this point, we can easily build $\bar{\mathbf{Q}}_d$ and $\bar{\mathbf{K}}_d$. We draw a $2K \times R$ matrix $\mathbf{Z}_d$ with independent and identically distributed (i.i.d.) Gaussian entries of unit variance, and define:

$$\bar{\mathbf{Q}}_d \leftarrow \boldsymbol{\Omega}(\mathcal{M}, \mathbf{f}_d, \boldsymbol{\theta}_d)\,\mathrm{diag}\big(\ddot{\boldsymbol{\lambda}}_d\big)\mathbf{Z}_d \big/ \sqrt{2K}\,, \tag{5.23}$$

$$\bar{\mathbf{K}}_d \leftarrow \boldsymbol{\Omega}(\mathcal{N}, \mathbf{f}_d, \mathbf{0})\,\mathrm{diag}\big(\ddot{\boldsymbol{\lambda}}_d\big)\mathbf{Z}_d \big/ \sqrt{2K}\,. \tag{5.24}$$

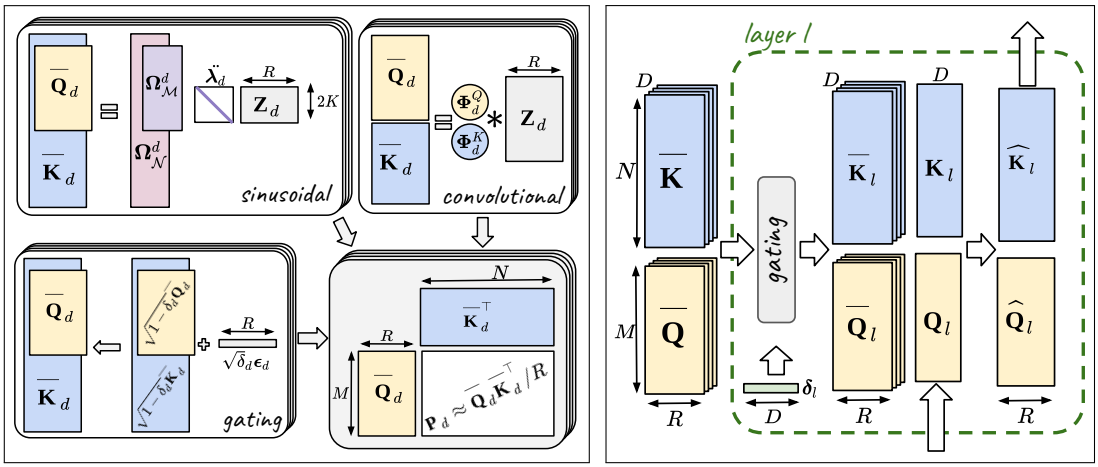
It is easy to check that such a construction leads to (5.16). Its parameters are $\{\mathbf{f}_d, \boldsymbol{\theta}_d, \boldsymbol{\lambda}_d\}_d$, which can be trained through stochastic gradient descent (SGD) as usual.

Variant II. Vanishing attention with regular sampling (`convSPE`). Due to their periodic structure, the covariance functions generated by Variant I are *non-vanishing*. Yet, our framework is flexible enough to allow for vanishing covariance structures, which may be more desirable depending on the application [Wang et al., 2021].

As opposed to Variant I, where we imposed a specific structure on $\mathcal{P}_d$, we will now follow an indirect approach, where $\mathcal{P}_d$ will be *implicitly* defined based on our algorithmic construction. In this case, we assume that the signals are regularly sampled (typical in e.g. text, images, audio), and we will exploit the structure of Gaussian random matrices and basic properties of the convolution operation.

For ease of notation, we assume self attention, i.e. $\mathcal{M} = \mathcal{N}$. Let $\{\boldsymbol{\Phi}_{\boldsymbol{d}}^{\boldsymbol{Q}}, \boldsymbol{\Phi}_d^K\}_d$ denote a collection of *filters*, which will ultimately be learned from training data. The size and the dimension of these filters can be chosen according to the input data (i.e. can be vectors, matrices, tensors). We then propose the following procedure, which leads to a Toeplitz $\mathbf{P}_d$ by means of *convolutions*:

[1] If $\mathbf{P}_d \succeq 0$ and $K \geq N$, Eq. (5.22) still holds but is not unique.



(a) The two SPE variants and gating (b) Layer-wise SPE sharing

Figure 5.1 – (a) Generation of $\bar{\mathbf{Q}}$ and $\bar{\mathbf{K}}$, which approximate the templates $\mathbf{P}_d$ when multiplied together. (b) $\bar{\mathbf{Q}}$ and $\bar{\mathbf{K}}$ can be shared across layers. At each layer l, different gating is (optionally) used before applying Eqs. (5.19) and (5.20) to generate new queries $\hat{\mathbf{Q}}$ and keys $\hat{\mathbf{K}}$.

- We first draw an $M \times R$ random matrix $\mathbf{Z}_d$ with i.i.d. standard Gaussian entries. For multidimensional signals, $\mathbf{Z}_d$ gathers R random vectors, matrices, cubes, etc.

- The desired $\bar{\mathbf{Q}}_d$ and $\bar{\mathbf{K}}_d$ are obtained by convolving $\mathbf{Z}_d$ with respective filters $\boldsymbol{\Phi}_d^Q$ and $\boldsymbol{\Phi}_d^K$:

$$\bar{\mathbf{Q}}_d = \mathbf{Z}_d * \boldsymbol{\Phi}_d^Q \,, \;\; \bar{\mathbf{K}}_d = \mathbf{Z}_d * \boldsymbol{\Phi}_d^K \,, \tag{5.25}$$

  where $*$ denotes convolution with appropriate dimension (e.g. 1D, 2D or 3D). Using convolutions with finite filters ensures vanishing covariance (see the appendix of ⟨ICML2021⟩ for a proof).

Due to the independence of the entries of $\mathbf{Z}_d$, for large R, the product $\mathbf{Z}_d\mathbf{Z}_d^\top/R$ will tend to the identity matrix. Given the fact the convolution operations in (5.25) can be equivalently expressed as a multiplication by triangular Toeplitz matrices constructed from the respective filters, it can be shown that, as $R \to \infty$, $\bar{\mathbf{Q}}_d\bar{\mathbf{K}}_d^\top/R$ tends to the product of two triangular Toeplitz matrices. Hence, by using the properties of triangular Toeplitz matrices (cf. Kucerovsky et al. [2016, Theorem 2.4]), we conclude that, as $R \to \infty$, our construction yields a Toeplitz matrix $\mathbf{P}_d$ as desired. This approach is parameterized by the filters $\{\boldsymbol{\Phi}_d^Q, \boldsymbol{\Phi}_d^K\}_d$, which will be learned from training data through SGD.

The variety of attention patterns $\mathcal{P}(m-n)$ that can be obtained directly depends on the kernel sizes, which is a classical result from signal processing [Vetterli et al., 2014]. Cascading several convolutions as in the VGGNet [Simonyan and Zisserman, 2015] may be a convenient way to augment the expressive power of this convolutional SPE variant.

From a more general perspective, the two operations in (5.25) can be understood as producing PE through filtering white noise. Other classical signal

processing techniques may be employed, such as using *infinite impulse response* filters. Such considerations are close to the ideas proposed by Engel et al. [2020].

To summarize, the core difference between the two proposed constructions (5.23–5.24) and (5.25) lies in the behaviour of RPE beyond a maximum lag, implicitly defined through the frequencies $\mathbf{f}_d$ for (5.23–5.24) and through the sizes of the filters for (5.25). While the sinusoidal construction leads to a periodic RPE, the filtering construction leads to a vanishing RPE. Both may be the desired option depending on the application.

5.2.2 Gating and sharing

Although RPE and the generalization (5.13) we propose are novel and efficient strategies to handle position information, it may be beneficial to also allow for attention coefficients that are computed without positional considerations, simply through $\mathbf{q}_m^\top \mathbf{k}_n$. As a general *gating* mechanism, we propose to weight between positional and non-positional attention through a *gate parameter* $\delta_d \in [0\ 1]$:

$$\mathbf{P}_d \equiv [\delta_d + (1 - \delta_d)\mathcal{P}_d(m, n)]_{m,n}. \tag{5.26}$$

This gating scheme can be implemented simply by augmenting $\bar{\mathbf{Q}}_d$ and $\bar{\mathbf{K}}_d$ generated as above through:

$$\bar{\mathbf{q}}_{d,m} \leftarrow \sqrt{1 - \delta_d}\bar{\mathbf{q}}_{d,m} + \sqrt{\delta_d}\boldsymbol{\epsilon}_d, \tag{5.27}$$

$$\bar{\mathbf{k}}_{d,m} \leftarrow \sqrt{1 - \delta_d}\bar{\mathbf{k}}_{d,m} + \sqrt{\delta_d}\boldsymbol{\epsilon}_d, \tag{5.28}$$

where $\boldsymbol{\epsilon}_d \in \mathbb{R}^R$ in Eqs. (5.27) and (5.28) is the same and has i.i.d. standard Gaussian entries.

In practice, we can share some SPE parameters across the network, notably across layers, to strongly reduce computing time and memory usage. In our implementation, *sharing* means generating a single instance of $\bar{\mathbf{Q}}$ and $\bar{\mathbf{K}}$ for each head, on which a layer-wise gating is applied before Eqs. (5.19) and (5.20). This is illustrated in Fig. 5.1.

5.3 Stochastic positional encoding – experimental results

In this section, we present experiments studying the performance of the proposed SPE on music generation.

Besides these experiments, we also include in Appendix A.2 results on the *Long-Range Arena* (*LRA*) benchmark [Tay et al., 2021], consisting of text and image classification tasks. On this benchmark, SPE tends to perform comparably to absolute positional encoding, with the sinusoidal variant bringing a considerable performance boost on one task.

The companion website[2] for SPE contains listening examples, as well as a link to the source code. See also Appendix C for a complete list of available resources.

[2] `https://cifkao.github.io/spe/`

5.3.1 Accompaniment continuation

This section presents an updated and extended version of the accompaniment continuation results from ⟨**ICML2021**⟩.

In this experiment, we evaluate Performers [Choromanski et al., 2021] with different positional encodings on an *accompaniment continuation* task. Similar to the accompaniment style transfer task from Section 4.4, the model is given a short example of an accompaniment (a 'style input') and is expected to continue generating in the same style; however, there is no content input to guide the generation, and the model simply generates freely. This is achieved simply by training the model for unconditional generation, then priming it with a short *prompt* (2-bar musical fragment) and letting it generate a continuation. We then observe whether the generated continuation matches the style of the prompt.

This experiment is specifically designed to investigate training on relatively short sequences (512 tokens) and generalization beyond the training length. This also enables us to train the original Transformer architecture (computing the full attention matrix) in the same setting, allowing to perform a direct comparison of SPE and RPE.

Experimental setup

The models have 24 layers, 8 attention heads and $d_{\text{model}} = 512$ and are trained on sequences of length $N = 512$, corresponding to 2–10 bars. At test time, the model is prompted with 2 bars in a style not seen during training and new tokens are sampled to complete the sequence to a length of 1024, i.e. twice the training length.

As a baseline, we include 'vanilla' Transformers/Performers, which add classical sinusoidal absolute positional encoding to the input (`APE`). The SPE-based models (`sineSPE`, `convSPE`) use layer-wise sharing and layer-specific gating. As an ablation experiment to investigate the importance of positional encodings in music generation, we include a Performer with no positional encoding (`noPE`).

On the Transformer side, we additionally include two variants of RPE, both of which are implemented exactly (without approximation) by modifying the computation of the attention matrix. First, the original `RPE` from Eqs. (5.10) and (5.11). Second, `sineRPE`, a sinusoidal version of RPE, where $\mathcal{P}_d$ in Eq. (5.11) is obtained via our (learnable) sinusoidal scheme (5.21).

Training data

We use the synthetic dataset from Section 4.4.1. We only use the training section of the dataset and perform a custom training/validation/test split such that each section contains a unique set of BIAB styles (2761 for training and 50 each for validation and testing).

We convert each accompaniment to a trio consisting of bass, drums and another randomly selected accompaniment track (e.g. piano, guitar). We then perform random data augmentation by skipping measures at the beginning, dropping some of the instruments, and transposition.

Data representation

We use a representation similar to the beat-relative encoding from Section 4.4.1, but adapted to a multi-track setting, taking inspiration from Donahue et al. [2019a]. Specifically, we encode a piece of music as a sequence of the following types of event tokens, each with two integer arguments:

- `NoteOn(track, pitch)`: Begins a new note at the given pitch (0–127).
- `NoteOff(track, pitch)`: Ends the note at the given pitch (0–127).
- `TimeShift(beats, offset)`: Advances current time by a given number of beats and then sets the offset within the beat, given as the number of ticks from its beginning (0–11). The maximum possible shift is `(2, 0)`.

Results and discussion

We use the *time-pitch*, *onset-duration* and *onset-drum* metrics from Section 4.2 to quantify the similarity of the generated continuation to the style of the prompt. When listening to the generated music, we notice a drift in quality along time. For this reason, we divide each generated sample into four successive chunks based on token position and evaluate them separately.

The results are displayed in Fig. 5.2. In the Transformer experiments, both `RPE` and `sineRPE` stay close to `sineSPE`, which is encouraging and confirms the soundness of our proposal.

For positions up to 512, neither of the positional encodings seems to have a substantial advantage over the others. This is an interesting finding, especially in the case of `noPE`, suggesting that positional encoding is not particularly important here. While seemingly surprising, this is in agreement with the literature, e.g. Irie et al. [2019] and Tsai et al. [2019], who found that removing positional encoding altogether in the decoder part of Transformer-based architectures only leads to a small drop in performance, and in some cases even a slight performance boost. This is explained by the fact that the causal masking, where the set of available keys strictly increases with query position, already provides a strong positional signal.

On the other hand, for positions exceeding the training length 512, the performance of `APE` drops sharply, while the other positional encodings are more stable (`convSPE` more so than `sineSPE`). This includes `noPE`, which achieves comparable or slightly better results than SPE.

To conclude, the experiment suggests that our proposed SPE behaves similarly to RPE on music generation, and in particular, it seems to inherit one of the celebrated advantages of RPE, its ability to generalize beyond training sequence lengths [Huang et al., 2019a]. On the other hand, we are able to obtain similar benefits by removing PE altogether. This raises the important question whether PE in general is beneficial at all – or even detrimental – in music and sequence generation. A more thorough investigation is needed to answer this question; we present follow-up results in this direction in Section 5.4.

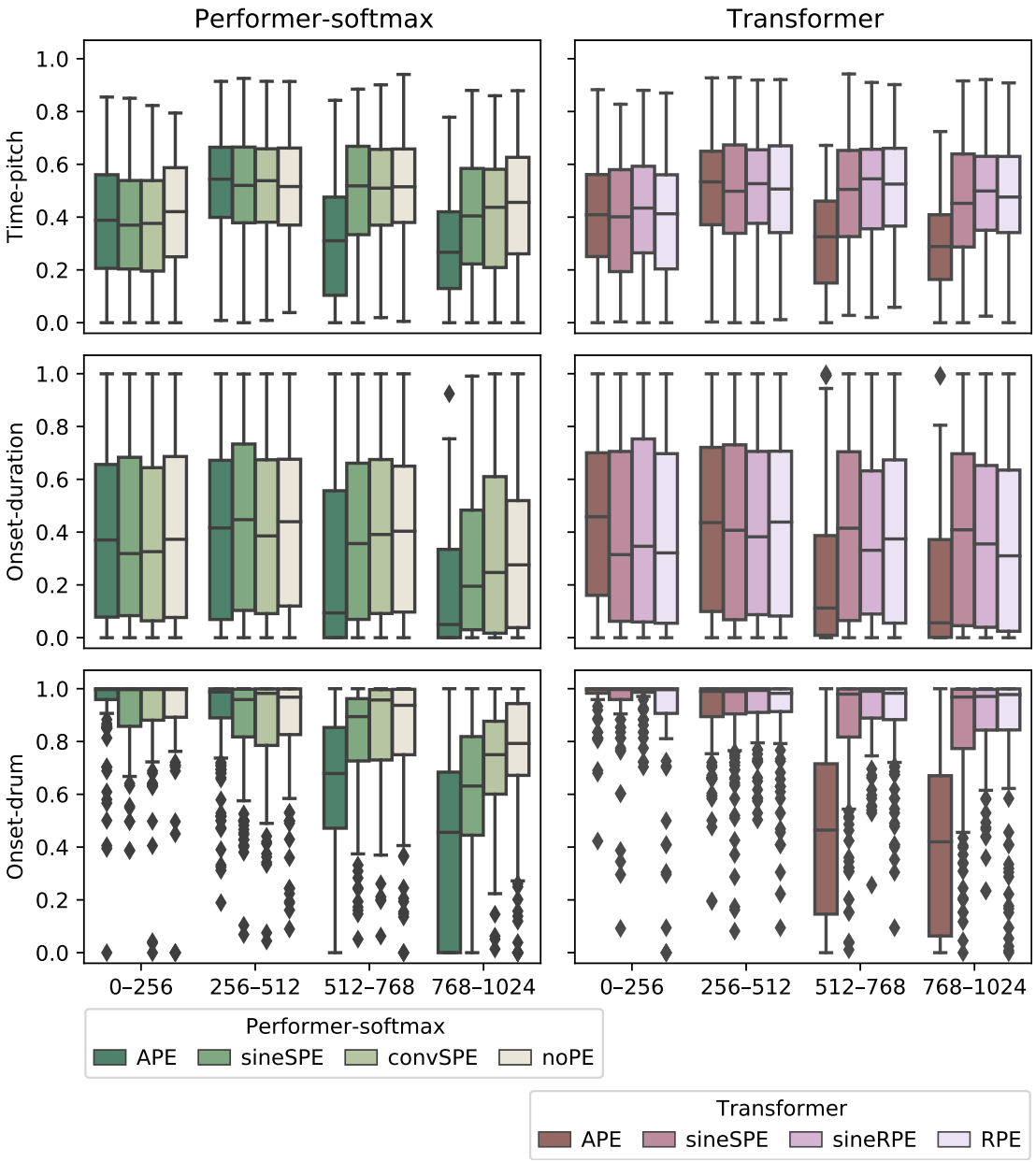
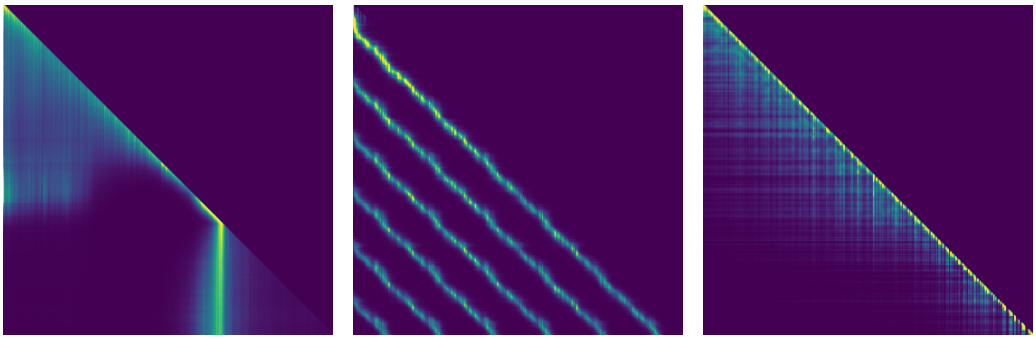


Figure 5.2 – Musical style similarity between output and initial prompt (higher is better). The results are sorted along the x axis by token position; positions ≥ 512 were not encountered during training. Each data point corresponds to a single BIAB style.



(a) `APE` (b) `sineSPE` (c) `convSPE`

Figure 5.3 – Examples of attention patterns observed at inference time in the Performers trained for pop piano music generation. The plots are for sequence length $M = N = 3072$ while training sequences have length 2048.

5.3.2 Pop piano music generation

> This section presents the contribution of S.-L. Wu and Y.-H. Yang. The content is reproduced from ⟨**ICML2021**⟩ and edited with their permission. See the paper for more details.

In this section, we present another music generation experiment, this time scaling up to longer sequences (2048 tokens) and focusing on *unconditional* generation of solo piano music.

Experimental setup

In this experiment, Performers, again with 24 layers and 8 heads per layer, were trained on a dataset composed of 1747 pop piano tracks, encoded using *REMI* [Huang and Yang, 2020]. The models were trained with sequence length $N = 2048$, corresponding to ∼1 minute of music. 5 models are included, differing only in the PE strategy, namely baseline `APE`, as well as `sineSPE` and `convSPE`, with or without gating.

Results and discussion

For qualitative assessment, we first display in Fig. 5.3 one attention pattern for each PE model: `APE` and `sineSPE`/`convSPE` (gated), obtained as an average over 20 from-scratch generations for a chosen layer-head pair. More plots can be found in Fig. B.3 in the appendix. Interestingly, we notice that for early layers, `APE` attention does not go much beyond training sequence length; this would explain the poor extrapolation performance of `APE` observed in Section 5.3.1, as well as below. This behaviour is not found in SPE variants, which consistently attend to all positions. Another remarkable feature of the proposed model (only displayed in the appendix) is that gating visually disables PE altogether for some layers/heads, in which case attention is global.

Since the literature suggests that RPE improves generalization performance [Shaw et al., 2018, Zhou et al., 2019, Rosendahl et al., 2019], we display in Fig. 5.4 validation cross entropy as a function of the target token position. The values

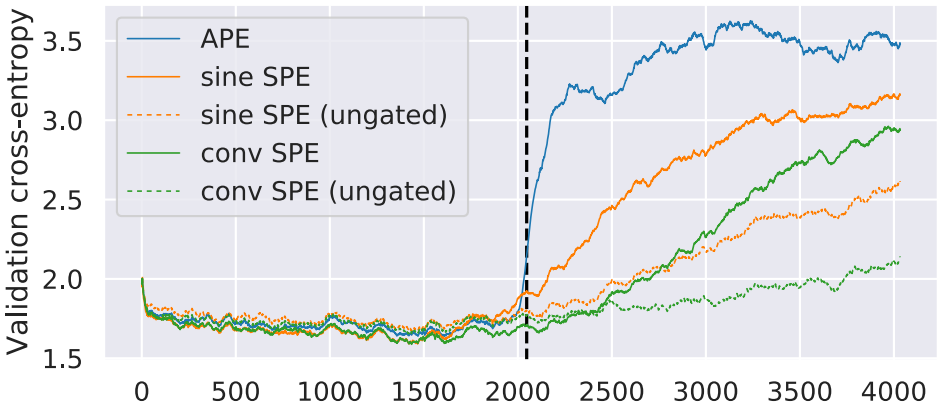


Figure 5.4 – Validation cross entropy (lower is better) vs. token position on the pop piano music generation task. The black vertical line indicates the maximum position to which the models are trained.

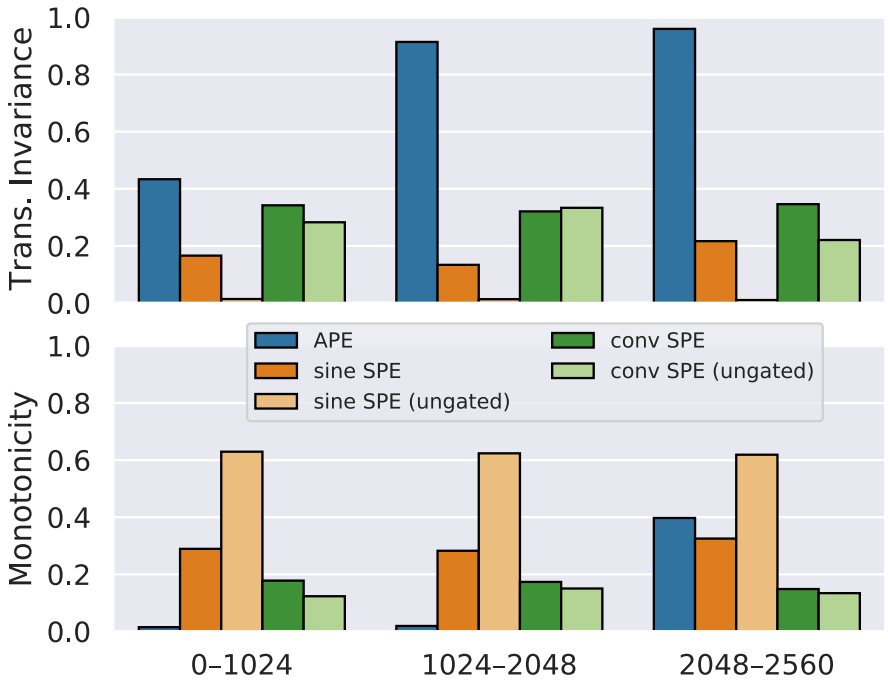


Figure 5.5 – PE evaluation metrics [Wang et al., 2021] for the pop piano music generation task in the 1st layer (lower is better), w.r.t. query positions. Training sequence length is 2048. Only query-key offsets <128 are considered here.

indicate how well the models predict the token at a certain position given the preceding tokens, for tracks in the validation set. As in Section 5.3.1, we notice that all SPE variants, especially *ungated* `convSPE`, behave much better than `APE` for token positions ≥2048. There is less difference between the different PEs for positions <2048, but *gated* `convSPE` seems to perform the best here, achieving slightly but consistently lower cross entropy values than `APE`.

Recently, Wang et al. [2021] defined metrics for the evaluation of PE, suggesting that *translation invariance* and *monotonicity* are desirable properties. Translation invariance states that the distance of two PEs should only depend on their relative position (i.e. be translation-invariant), while monotonicity states that neighboring positions should be assigned PEs that are close to each other, compared to faraway positions. Following their *identical word probing* methodology, these metrics are reported in Fig. 5.5. As expected, SPE variants greatly outperform `APE` in terms of translation invariance. This is not the case for monotonicity; still, SPE scores are remarkably stable across positions, contrarily to `APE`, which again rapidly degrades beyond the training length.

5.4 Metrical positional encoding

The experiments in the previous section seemed to suggest that PEs of either kind are not particularly beneficial for symbolic music generation using Transformers. However, one aspect that we omitted is the fact that the usual way PE is used in music generation is not *musically meaningful*. This is because in token-based representations, token indices do not have any obvious semantics besides determining order. In particular, the indices of two tokens generally do not convey any information about the delay between the corresponding musical events, and do not even allow to determine whether the events occur simultaneously or not. Hence, the question arises whether PE could be better leveraged by making it encode more musically relevant position information, e.g. metrical timing of events as in Huang et al. [2019a]. In this section, we propose an experiment that revisits this question.

Experimental setup

We train Linear Transformers[3] [Katharopoulos et al., 2020] for music generation. The networks are relatively large with 28 layers, 8 attention heads, $d_{\text{model}} = 1024$ and a maximum sequence length of $N = 1024$. We compare the following:

(a) classical (sinusoidal) APE, encoding token indices (`APE`);

(b) *metrical* positional encoding: APE encoding the metrical time in ticks (12ths of a beat) from the beginning of the piece (`MPE`);

(c) no positional encoding (`noPE`).

[3]Linear Transformers are simpler compared to Performers in that they use a deterministic feature map, and they achieved better results in the LRA benchmark in Appendix A.2. Here, following Katharopoulos et al. [2020], we use the feature map $\phi(x) = \text{ELU}(x) + 1$, where ELU is the *exponential linear unit* activation [Clevert et al., 2016].

Even though SPE (especially the sinusoidal variant) is in principle capable of naturally handling irregularly sampled sequences as in (b), we unfortunately found it computationally infeasible in this context. This is mainly because each example in a training batch has a distinct index set $\mathcal{N}$, which prevents sharing the same $\bar{\mathbf{Q}}$ and $\bar{\mathbf{K}}$ within the batch, leading to high memory consumption. For this reason, we limit ourselves to APE (vanilla or timing-flavored) in this experiment.

We adopt the multi-track token-based representation from Section 5.3.1, but besides the *beat-relative* encoding of timing (with `TimeShift(beats, offset)` tokens), we alternatively use a Δ-*encoding* (as in Chapter 4) with tokens of the form `TimeShift(delta)`, simply encoding the time differences between consecutive events. The basic time unit for both representations is a 12th of a beat.

Note that our proposal (b) is different from that of Huang et al. [2019a] (outlined in Section 5.1.3) in two key aspects. First, Huang et al. use RPE, which is not feasible with our large sequence length and model size;[4] moreover, absolute timing carries additional information such as the offset within the current beat, which may help anchor the generated music to the metrical grid. Second, we do not encode token indices (unlike Huang et al., who encode both indices and timing information), as we have argued in Section 5.3.1 that this is not necessary.

Also note that neither `MPE` nor the beat-relative encoding provide any additional information about the metrical grid (since beats happen at regular intervals and the first beat is assumed to occur at time 0) compared to the other options we consider; what they do provide is simply a more explicit representation of this information.

We evaluate the trained models on a music continuation task as in Section 5.3.1. Specifically, we prime the model with a 16-beat prompt (4 bars assuming that the time signature is $\frac{4}{4}$) and let it generate a continuation not exceeding the training length, employing nucleus sampling [Holtzman et al., 2020] with $p = 0.8$ and softmax temperature $\tau = 0.9$. We then measure the style similarity between the continuation and the song from which the prompt was extracted.

Training data and procedure

We use the 'full' version (`LMD-full`) of the Lakh MIDI Dataset[5] [Raffel, 2016], containing 178 k MIDI files. As in Section 5.3.1, we obtain a dataset of trios by randomly choosing three tracks – drums, bass, other – from each file (here, however, the third track is not necessarily an *accompaniment* track). We train for 10 epochs ($\sim$44 h on a Tesla V100 GPU) while applying similar data augmentation techniques as in Section 5.3.1. We also employ a form of *curriculum learning* [Bengio et al., 2009] where we gradually increase the length of the training sequences from 32 to 1024 for the first $\sim$3.4 epochs.

Results and discussion

As before, we use the metrics from Section 4.2 to quantify the style similarity of the generated continuation to the prompt. Because LMD includes files with mixed time signatures and files where downbeat (measure boundary) locations

[4]The authors themselves note that their relation-aware RPE does not scale beyond Bach chorales.

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

cannot be reliably identified, we modified the *onset-duration* and *onset-drum* metrics so that the onset time is now represented relative to the current beat and not the last downbeat. Our (preliminary) results on the validation set are shown in Fig. 5.6.

The Δ-encoding turns out to be very sensitive to the choice of PE: with `APE`, it scores the lowest of all setups on all three metrics, but with `MPE`, it scores the highest. The beat-relative encoding maintains relatively good performance for all three PEs, but is still outperformed by the Δ-encoding with `MPE` on 2 of the 3 metrics.

We notice two surprising results. Firstly, `noPE` tends to achieve better results than `APE` (always comparing results on the same metric and with the same token-based representation). This effect is especially strong with the Δ-encoding. This suggests that injecting token positions via APE is actually harmful in this case, perhaps acting mostly as noise that the network can easily get distracted by and/or overfit to.

Secondly, the beat-relative encoding, which appeared superior to the Δ-encoding in Section 4.4.2, does not yield the best results even when combined with `MPE`. We do not have a convincing explanation for this, but we speculate that, provided that metrical timing information is present in the input, it may be easier for the network to predict a simple relative time shift than to compute the next beat-relative position.

We also display in Table 5.1 the validation losses achieved by the different models. Interestingly, the results do not correlate with those in Fig. 5.6: all the values for `APE` and `MPE` are close to 0.613, with `APE` *slightly* lower – i.e. better – than `MPE`; on the other hand, the value for `noPE` is considerably higher in both cases (0.6368, 0.6256). Hence, we can see that choices which clearly benefit generation, possibly by making 'errors' less likely to occur, do not necessarily make the learning task easier.

We would also like to remark that our objective metrics in Fig. 5.6 should not be taken as an absolute measure of output quality and that the ideal evaluation method is a subjective listening test. This is especially true here (as opposed to when working with synthetic BIAB data as in Chapter 4 and Section 5.3.1), since a good continuation might diverge significantly from the style of the prompt.

Listening to the generated outputs,[6] we notice that all models are capable of imitating the style of the prompt almost perfectly, often copying the prompt only with small but meaningful variations. On the other hand, all of the models also occasionally enter an 'arhythmic state' and fail to produce any musically meaningful material, or quickly drop some of the instruments of the trio. (Note that the latter issue cannot be captured by our metrics, which evaluate each instrument separately.) These issues seem to be more common with Δ-encoding + `APE`/`noPE` than the other models.

5.5 Conclusion

In this chapter, we studied positional encodings (PEs) and their role in music generation using efficient Transformers. In Sections 5.2 and 5.3, we presented

[6] `https://cifkao.github.io/metrical-pe/`

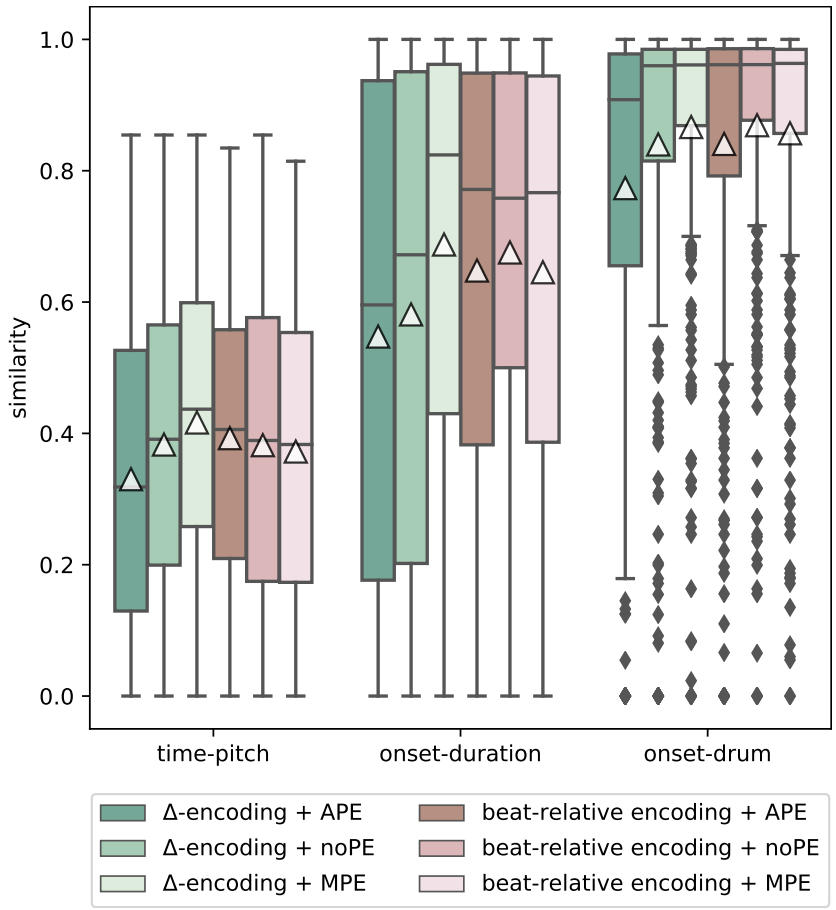


Figure 5.6 – Musical style similarity between output and initial prompt (higher is better). The triangles indicate the mean. Each data point corresponds to a single file from the validation set.

Δ-encoding			beat-relative encoding		
APE	noPE	MPE	APE	noPE	MPE
0.6135	0.6368	0.6140	0.6122	0.6256	0.6125

Table 5.1 – Validation cross entropies (lower is better) achieved by the models. Note that the results for Δ-encoding and beat-relative encoding are not necessarily comparable.

SPE, a novel PE that generalizes RPE, allowing to induce a given relative attention pattern while avoiding any explicit computation in the attention domain. This makes it compatible with linear complexity Transformers, allowing for the first time to test RPE-like attention at scale. Our experimental results confirm that SPE behaves like RPE on music generation, in particular inheriting its generalization capabilities on sequence lengths unseen during training. At the same time, we observe similar behavior when PE is ablated; this raises a question about the real benefits of RPE – and PE in general – in sequence generation tasks.

We partly answered this question in Section 5.4, conducting an experiment which suggests that (i) conventional absolute PE, encoding token indices, can be *detrimental* for music generation compared to *no PE*, and (ii) absolute PE which encodes *metrical time* instead of indices (called here *metrical positional encoding*) is *beneficial* to at least the same extent as token-based representations that explicitly encode the metrical grid. It will be interesting to see whether these results generalize to architectures like Transformer-XL [Dai et al., 2019], which rely on RPE, and how metrical PE behaves together with – and compares to – more sophisticated token-based representations like REMI [Huang and Yang, 2020].

Coming back to music style transfer, the main topic of this thesis, we are excited about the observed 'style imitation' capabilities of efficient Transformers, especially when combined with timing-based PE. We are hoping to leverage these capabilities to improve upon the one-shot style transfer performance of our Groove2Groove system (Section 4.4).

6. Self-supervised audio timbre transfer

This chapter is based on the paper ⟨**ICASSP2021**⟩ by Cífka et al. © 2021 IEEE.

Let us now turn from symbolic music to audio, and specifically to one-shot audio timbre transfer, a challenging problem that has received little attention in prior work. We view this task as a variant of music style transfer, but with a considerably different definition of style and content than in previous chapters. Namely, we consider *single-instrument*[1] (but not necessarily monophonic) audio recordings, and define style as *timbre* (understood roughly as instrument identity, including any audio effects) and content as *pitch*. Our goal is then to transfer the timbre of the style input onto the content input while preserving the pitch content of the latter.

Clearly, pitch is a *local* (fast-changing) feature, while timbre, in our definition, is a *global* feature (staying largely the same over the course of a recording). Note that there are other features that characterize a recording, notably loudness. In this work, we do not prescribe (or evaluate) whether or to what extent such features are transferred. However, we intuitively expect that local variations in these features end up being part of content, while global characteristics (e.g. average loudness) should belong to style.

To address the task, 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. Engel et al. [2020], Wang et al. [2020b], Nercessian [2020], as well as our own work in Chapter 4), 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 [van den Oord et al., 2017]), self-supervised learning, and data augmentation.

The contributions presented in this chapter can be summarized as follows:

- We propose 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 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.

[1] Considering multi-instrument recordings would lead to a more challenging and open-ended task closer to *instrumentation transfer*, which includes assigning an instrument (or set of instruments) to each note.

- 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.

The companion website[2] for this project contains audio examples, as well as links to the source code and a demo notebook. See also Appendix C.

6.1 Background

6.1.1 Vector-quantized variational autoencoder

The *vector-quantized variational autoencoder* (*VQ-VAE*; van den Oord et al. [2017]), is an autoencoder with a *discrete* latent space. Its latent representation is a sequence of discrete symbols from a (learned) finite dictionary, which puts an explicit limit on its capacity. While the authors' goal is not representation disentanglement, they show experimentally that after training this model on speech with conditioning on the speaker identity, it is possible to achieve voice conversion simply by switching the speaker label. They conclude that the model learns 'a high-level abstract space' capable of representing an utterance in a speaker-invariant way – in other words, the autoencoder's latent space becomes disentangled from the speaker embedding space.

A VQ-VAE consists of an encoder, which maps the input x to a sequence $\mathbf{Z} \coloneqq (\mathbf{z}_1, \ldots, \mathbf{z}_L)$ of discrete code vectors from a codebook, and a decoder, which tries to map $\mathbf{Z}$ back to x. Formally, given the input x, the encoder neural network E first outputs a sequence $E(x) = \mathbf{E} \in \mathbb{R}^{L \times D}$ 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) $\mathbf{Q} \in \mathbb{R}^{K \times D}$:

$$\mathbf{z}_i = Q(\mathbf{e}_i) = \underset{\mathbf{q}_j, 1 \leq j \leq K}{\arg\min} \left\| \mathbf{e}_i - \mathbf{q}_j \right\|. \tag{6.1}$$

The model is trained to minimize a reconstruction error $\mathcal{L}_{\text{ae}}$ between the input x and the output of the decoder $D(Q(E(x))) = D(\mathbf{Z})$. 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 $\mathbf{Q}$ and the encoder outputs $E(x)$, two other terms appear in the VQ-VAE objective – the codebook loss and the commitment loss:

$$\mathcal{L}_{\text{cbk}} = \frac{1}{L} \sum_{i=1}^{L} \left\| \text{sg}\left[\mathbf{z}_i\right] - \mathbf{e}_i \right\|^2, \tag{6.2}$$

$$\mathcal{L}_{\text{cmt}} = \frac{1}{L} \sum_{i=1}^{L} \left\| \mathbf{z}_i - \text{sg}\left[\mathbf{e}_i\right] \right\|^2. \tag{6.3}$$

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

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

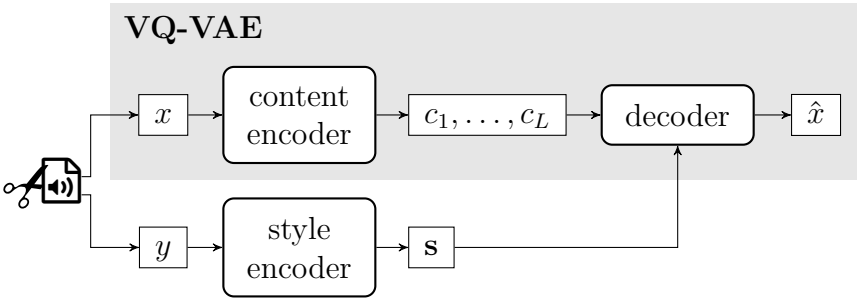


Figure 6.1 – A high-level depiction of the proposed method. We extract pairs of segments from audio files and use them for self-supervised learning of a VQ-VAE with an additional style encoder. The content representation $c_1, \ldots, c_L$ is discrete, the style representation $\mathbf{s}$ is continuous. © 2021 IEEE.

derivatives w.r.t.) the codebook $\mathbf{Q}$ (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:

$$\mathcal{L} = \mathcal{L}_{\text{ae}} + \mathcal{L}_{\text{cbk}} + \beta \mathcal{L}_{\text{cmt}} \tag{6.4}$$

6.1.2 Self-supervised learning

Self-supervised learning is a family of techniques 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 [Mikolov et al., 2013] or a missing patch in an image [Pathak et al., 2016]), the original orientation of a rotated image [Gidaris et al., 2018] or the 'arrow of time' in a (possibly reversed) video [Wei et al., 2018]. The representation arising from (pre-)training on the pretext task is then typically used as features for a *downstream task*.

In the present work, following this principle, we extract pairs of excerpts from audio files and rely on them to learn a style representation as detailed in the following section. Nevertheless, the method we propose is not a *typical* self-supervised one, since we do not have a pre-training stage followed by fine-tuning on a downstream task. Instead, we simply train our model on the pretext task (a reconstruction task) and then directly use it to perform style transfer.

6.2 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. 6.1, 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 6.1.1.

(*ii*) The output of our style encoder is a single *continuous-valued* embedding vector $\mathbf{s}$. 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 6.2.1 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 $\mathbf{s}$ to only encode style. Also note that the discretization bottleneck in (*i*) is key for learning a useful style representation $\mathbf{s}$: 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.

6.2.1 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 (`LMD-full`) of the Lakh MIDI Dataset[3] [Raffel, 2016], 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 (*Fluid R3 GM*, *TimGM6mb*, and *Arachno SoundFont*).[4]

2. **RT.** A set of audio tracks from PG Music;[5] 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

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

[4] See the MuseScore SoundFont list: `https://musescore.org/en/handbook/soundfonts-and-sfz-files`

[5] `https://www.pgmusic.com`

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[6] and 90 k from RT).

6.2.2 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 algorithm of Griffin and Lim [1983].

The model architecture is depicted in Fig. 6.2. 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; Cho et al. [2014]) layer, whose final state $\mathbf{s}$ (a 1024-dimensional vector) is used as the style representation. This vector $\mathbf{s}$ 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 for better temporal modeling, particularly to combine the content and style representations in a context-aware fashion.

We train the model using Adam [Kingma and Ba, 2015] to minimize the VQ-VAE loss from Eq. (6.4), 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.

[6]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.

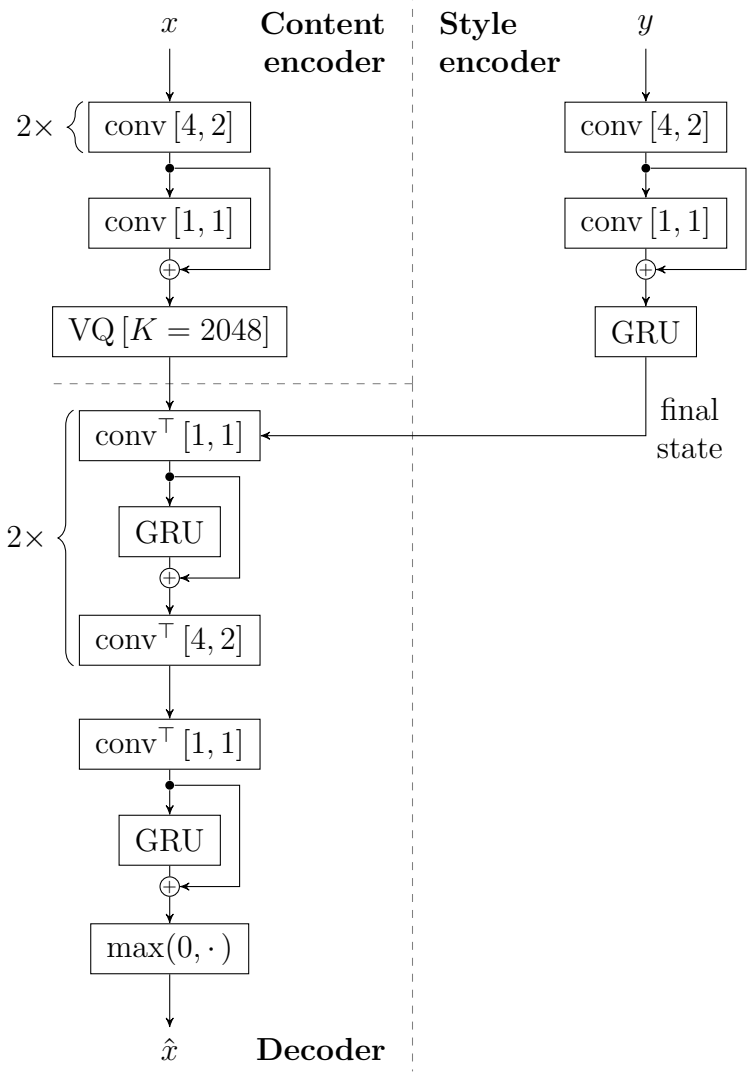


Figure 6.2 – The proposed timbre transfer model architecture. All convolutions are 1D, with the kernel size and stride shown. All layers have 1024 channels, except for the last two (conv$^\top$ & GRU), which have 1025 (the number of frequency bins). All layers except for the input layers and the VQ are preceded by batch normalization and a Leaky ReLU activation [Maas et al., 2013]. conv$^\top$ stands for transposed convolution. © 2021 IEEE.

6.3 Evaluation

As in Chapter 4, 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 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 [Salamon and Gómez, 2012] implemented in the Essentia library [Bogdanov et al., 2013]. 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 [Richard et al., 2013]. Since they are computed on a per-frame basis, instead of comparing them directly, we apply *metric learning* [Hoffer and Ailon, 2015] on top of them to aggregate them over time and get a single dissimilarity score.

More precisely, we use a *triplet network* with a convolutional architecture, which receives a sequence of MFCC vectors (only coefficients 2–13) and outputs an embedding. The metric is then computed as the cosine distance $D_{\cos}$ in this embedding space. The network is trained to minimize the *triplet hinge loss*:

$$\mathcal{L}\Big(x, x^+, x^-\Big) = \max\Big\{0, D_{\cos}\Big(f(x), f(x^+)\Big) - D_{\cos}\Big(f(x), f(x^-)\Big) + 0.1\Big\},$$

where f is the function implemented by the network, x is an *anchor*, x^+ is a *positive example* and x^- is a *negative example*. The training dataset consists of 7381 such triplets extracted from the Mixing Secrets data so that x and x^+ are from the same file and x^- is from a different file. The aim is to make the metric good at discriminating between different instruments, but largely pitch-independent.

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 [2016] (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 (λ_c, λ_s) of the two losses on a small synthetic validation set to minimize the log-spectral distance (LSD) to the ground

truth (see Section 6.3.1 below). However, since the ratio found by this procedure ($\lambda_s = 10^{-2.1}\lambda_c$) is too small for the style input to have any noticeable effect on the output, we also include results with a larger style weight ($\lambda_s = 10\lambda_c$) chosen manually so that the effect of both inputs is clearly audible. (In both cases, the weights are normalized: $\lambda_s + \lambda_c = 1$.)

- *Musaicing*: A freely available implementation[7] of the musaicing algorithm of Driedger et al. [2015]. Note that this implementation involves a pre-processing step which automatically transposes (pitch-shifts) the style input to all 12 keys and concatenates the results.

6.3.1 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 generated the data based on the Lakh MIDI Dataset (LMD) similarly as in Section 6.2.1, but using a set of files held out from the training set, and with no data augmentation. We rendered the music using the *Timbres Of Heaven* SoundFont,[4↑] not used in the training set.

We randomly drew 721 content-style input pairs and generated a corresponding ground-truth target for each pair by synthesizing the content input using the instrument (MIDI program) of the style input. To avoid pairs of extremely different inputs (e.g. bass line + piccolo duet) for which the task would make little sense, we sorted all instrument parts into 4 bins using two median splits: on the average pitch and on the average number of voices (simultaneous notes); we then formed each pair by drawing two examples from the same bin. To obtain a balanced distribution of instruments, we limited the total number of examples per MIDI program to 4.

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 *log-spectral distance* or **LSD**.

6.3.2 'Real data' benchmark

We created a more realistic test set based on the 'Mixing Secrets' audio library,[8] containing over 400 multi-track recordings from various (mostly popular music) genres. After filtering out multi-instrument, vocal and unpitched percussion tracks, we extracted 690 content-style input pairs similarly as in Section 6.3.1. (In this case, we used multi-pitch MELODIA to estimate the average pitch and number of voices of each track, then used this information to perform the median splits.)

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.

[7] `https://github.com/ctralie/LetItBee/`

[8] `https://www.cambridge-mt.com/ms/mtk/`

	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, $\lambda_c/\lambda_s = 10^{-2.1}$ (tuned)	14.50	0.3483	**0.5441**	0.4792	**0.1315**
U+L, $\lambda_c/\lambda_s = 10$	23.08	0.3645	0.6508	0.4579	0.6043
Musaicing	14.51	0.2933	0.6445	0.2319	0.6297
This work	**12.16**	**0.2063**	0.5500	**0.2278**	0.6197

Table 6.1 – 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. U+L = Ulyanov and Lebedev [2016]; Musaicing = Driedger et al. [2015]. © 2021 IEEE.

6.4 Experimental results

The results of both benchmarks are shown in Table 6.1. 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 say that this transformation changes the input in the correct 'direction' 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.

6.5 Discussion

We will now attempt to explain the results from the previous section and complement them with our subjective observations. We encourage the reader to listen to the provided audio examples, which should make the following more obvious.

6.5.1 Our system

Our subjective observations upon examining the outputs 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 [Pons et al., 2021]. 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 BPM, only 25.4 bits of information can be encoded per beat. This 'codebook collapse' [Dieleman et al., 2018] 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 [van den Oord et al., 2016].

Finally, choosing a more suitable loss function, such as a perceptual loss [Manocha et al., 2020] or an adversarial loss [Donahue et al., 2019b] could also help improve the output quality.

6.5.2 Baselines

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.

As for U+L, we have already mentioned that tuning it automatically resulted in a style weight which is too small (about 100 times lower than the content weight). It is therefore unsurprising that the tuned variant performs close to the CP-CONTENT baseline on all metrics (i.e. achieving excellent results on pitch, but poor on timbre). In other words, it seems that for U+L, the best strategy to minimize LSD – which captures both timbre and pitch – is to ignore timbre altogether.

By increasing the style weight manually to achieve subjectively better timbre transfer, we were able to improve the timbre metric on the real test set (though it is still far behind our system) but all other metrics deteriorated. When listening to the outputs, 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. This is a sign that the style representation is heavily pitch-dependent, and therefore not suitable for this task.

6.6 Conclusion

In this chapter, 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 seem to be artifacts and in general subjectively poor output audio quality. We believe that a more expressive decoder such as a WaveNet, RNN or Transformer 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. Alternatively, it may be possible to improve the output quality by a more careful choice of loss function.

7. Conclusion

In this work, we studied *music style transfer*, a family of tasks where the goal is to transfer the style of one musical piece or fragment onto another. Our main contribution lies in proposing novel approaches and evaluation metrics for two tasks which have received little attention in prior work: *accompaniment style transfer* and *audio timbre transfer*. We have also contributed to the field of *symbolic music generation* and to research on *positional encodings* in Transformer models.

In the rest of this final chapter, we give a more detailed summary of our contributions, followed by directions for future research.

7.1 Summary of contributions

Task definitions. We hope to have helped clear the terminological vagueness around music style transfer by proposing the terms *style conversion*, *style translation* and *one-shot style transfer*. We also echo the call of Xia and Dai [2018] for more precise definitions of different variants of these tasks and we have followed it to the best of our ability in defining our accompaniment style conversion and timbre transfer tasks.

Music style conversion methods. We have proposed two novel methods for music style conversion: (1) an approach to *accompaniment style transfer* (and as an intermediate step, *accompaniment style translation*), based on supervised learning from a synthetic dataset; (2) an approach to *audio timbre transfer*, based on an extension of the *vector-quantized variational autoencoder* (*VQ-VAE*), along with a simple *self-supervised* learning strategy. Our results demonstrate the ability of our methods to transfer the style from the style input while preserving the content of the content input. We also provide additional experiments and analyses aimed at a better understanding of the proposed methods.

Music style conversion evaluation. For both (1) and (2) above, we have designed objective metrics measuring *content preservation* and *style fit*, two essential criteria for evaluating style conversion. To our knowledge, these metrics constitute the first automatic evaluation protocol for music style transfer that considers both of these criteria.

Symbolic music generation. As a general contribution to symbolic music generation, we have proposed two novel, beat-aware encodings of timing for event-based representations: *beat-relative encoding* and *metrical positional encoding*. Our results suggest that both lead to improved generation quality.

Positional encodings for Transformers. We have proposed *stochastic positional encoding* (*SPE*), a novel positional encoding strategy for Transformer models capturing *relative positions* while being compatible with a recently proposed

family of efficient Transformers. In music generation experiments, we demonstrated that SPE allows for better extrapolation beyond the training sequence length than the commonly used *absolute positional encoding* (*APE*). We also evaluated SPE on a non-music benchmark and found that it performs at least as well as APE (bringing a performance boost on one task).

On the other hand, our ablation experiments suggest that in music generation, positional encodings in general (including SPE) bring little to no benefit in the form that is commonly used, compared to not using any positional encoding whatsoever. To further investigate the importance of positional encodings in music generation, we have proposed the aforementioned *metrical positional encoding* as a way to use (absolute) positional encodings to more robustly encode metrical timing in music. Unlike the classical index-based APE, this novel timing-based APE appears to be strongly beneficial for music generation.

Reproducibility. We have made publicly available the data pre-processing, training and evaluation code, as well as complete hyperparameter settings for all the published papers. We have also released the *Groove2Groove MIDI Dataset*, generated for training our accompaniment style transfer system, making it to our knowledge the first publicly available *parallel* (pairwise aligned) symbolic music dataset. Another remarkable feature of this data are the exceptionally fine-grained style labels.

7.2 Future directions

Symbolic music style transfer. While we have shown our supervised approach to one-shot accompaniment style transfer to be effective, we have also identified some important limitations, namely:

- It is restricted to accompaniments, i.e. it is not designed to handle melodies or music without a clear melody-accompaniment distinction.
- It does not account for interactions between different instruments.
- There is room for improvement in the one-shot generalization to novel styles.

An approach capable of overcoming these issues will need to be able to model multi-track music without strong independence assumptions, perhaps using techniques similar to those that we have explored in our work on music generation using efficient Transformers. It will also need to be trained on more open-domain data, calling for methods capable of learning from non-parallel (unaligned) datasets. One such method is our 'self-supervised VQ-VAE', which was developed for audio timbre transfer, but may be equally applicable to accompaniment or arrangement style transfer. An interesting future direction is then to tackle these tasks by combining a Transformer decoder with a vector-quantized encoder and a variant of our self-supervised learning strategy.

Another possible improvement is to modify the approach to make it capable of taking audio (as opposed to MIDI files) as input, rendering the system more accessible to end users.

Positional encodings for Transformers. Our research on positional encodings raises interesting questions that merit further investigation, namely:

- How important are strong positional encoding schemes for different tasks? In which tasks are they critical?
- Can positional encoding do harm in sequence generation tasks?
- How do our results on timing-based positional encodings generalize to other Transformer architectures used for music generation, more sophisticated token-based representations or even tasks other than music generation?

Audio timbre transfer. The most important shortcomings of our timbre transfer method seem to be artifacts and in general subjectively poor output audio quality. This could be attributed to the use of a deterministic decoder, the chosen loss function or the Griffin-Lim reconstruction algorithm. We believe that a perceptual loss function or a more expressive model such as a WaveNet, recurrent network (RNN) or generative adversarial network (GAN) should allow improving the performance.

In practical applications, real-time processing will also be desirable, as it will enable timbre transfer to be used during live performances.

Style transfer in other modalities. Due to the data-driven nature of our 'self-supervised VQ-VAE', we believe that it may not only be applicable to different kinds of music style transfer, but also serve as a general style transfer framework for other modalities such as speech, images and text.

Cross-modal conditioning. Whereas we have focused on transferring style from a given music example, follow-up work should explore music transformations controlled by non-musical inputs, as we have described in the introduction. Examples include transforming music to convey a given emotion or mood or to fit the narrative of a video, a gameplay situation, a real-life occasion or the musical taste of a specific user or demographic group. The methods that we have developed in this work could serve as a basis for implementing such transformations.

Bibliography

H. F. Aarabi and G. Peeters. Music retiler: Using NMF2D source separation for audio mosaicing. In *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion*. ACM, 2018. doi: 10.1145/3243274.3243299. URL https://doi.org/10.1145/3243274.3243299.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations (ICLR 2015)*, 2015. URL http://arxiv.org/abs/1409.0473.

Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48. Association for Computing Machinery, 2009. doi: 10.1145/1553374.1553380. URL https://doi.org/10.1145/1553374.1553380.

A. Bitton, P. Esling, and T. Harada. Vector-quantized timbre representation. In *International Computer Music Conference*, July 2021. URL https://hal.archives-ouvertes.fr/hal-03208036.

D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, and X. Serra. Essentia: An audio analysis library for music information retrieval. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*. ISMIR, 2013. doi: 10.5281/zenodo.1415016. URL https://doi.org/10.5281/zenodo.1415016.

J.-P. Briot, G. Hadjeres, and F.-D. Pachet. *Deep Learning Techniques for Music Generation*. Springer International Publishing, 2020. doi: 10.1007/978-3-319-70163-9. URL https://doi.org/10.1007/978-3-319-70163-9.

P. F. Brown, J. Cocke, S. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16:79–85, 1990.

Y. Broze and D. Shanahan. Diachronic changes in jazz harmony. *Music Perception: An Interdisciplinary Journal*, 31(1):32–45, 2013. doi: 10.1525/mp.2013.31.1.32.

G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer. MIDI-VAE: Modeling dynamics and instrumentation of music with applications to style transfer. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*. ISMIR, 2018a. doi: 10.5281/zenodo.1492525. URL https://doi.org/10.5281/zenodo.1492525.

G. Brunner, Y. Wang, R. Wattenhofer, and S. Zhao. Symbolic music genre transfer with CycleGAN. In *IEEE 30th International Conference on Tools with Artificial Intelligence*. IEEE, 2018b. doi: 10.1109/ICTAI.2018.00123. URL https://doi.org/10.1109/ICTAI.2018.00123.

M. Caccia, L. Caccia, W. Fedus, H. Larochelle, J. Pineau, and L. Charlin. Language GANs falling short. In *Critiquing and Correcting Trends in Machine Learning: NeurIPS 2018 Workshop*, 2018.

X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016.

R. Child, S. Gray, A. Radford, and I. Sutskever. Generating long sequences with sparse Transformers. *CoRR*, abs/1904.10509, 2019. URL http://arxiv.org/abs/1904.10509.

K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014. doi: 10.3115/v1/D14-1179. URL https://www.aclweb.org/anthology/D14-1179.

K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel. Encoding musical style with transformer autoencoders. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020. URL https://proceedings.mlr.press/v119/choi20b.html.

K. M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlós, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, D. B. Belanger, L. J. Colwell, and A. Weller. Rethinking attention with Performers. In *9th International Conference on Learning Representations (ICLR 2021)*. OpenReview.net, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

D. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *4th International Conference on Learning Representations (ICLR 2016)*, 2016. URL http://arxiv.org/abs/1511.07289.

K. Collins. An introduction to procedural music in video games. *Contemporary Music Review*, 28(1):5–15, 2009. doi: 10.1080/07494460802663983. URL https://doi.org/10.1080/07494460802663983.

Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL 2019)*. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1285. URL https://doi.org/10.18653/v1/p19-1285.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional Transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

S. Dieleman, A. van den Oord, and K. Simonyan. The challenge of realistic music generation: modelling raw audio at scale. In *Advances in Neural Information Processing Systems*, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/3e441eec3456b703a4fe741005f3981f-Abstract.html.

S. Dixon, W. Goebl, and G. Widmer. The "air worm": an interface for real-time manipulation of expressive music performance. In *Proceedings of the 2005 International Computer Music Conference (ICMC 2005)*. Michigan Publishing, 2005. URL http://hdl.handle.net/2027/spo.bbp2372.2005.048.

C. Donahue, H. H. Mao, Y. E. Li, G. Cottrell, and J. McAuley. LakhNES: Improving multi-instrumental music generation with cross-domain pre-training. In *Proceedings of the 20th International Society for Music Information Retrieval Conference*, pages 685–692. ISMIR, 2019a. doi: 10.5281/zenodo.3527902. URL https://doi.org/10.5281/zenodo.3527902.

C. Donahue, J. J. McAuley, and M. S. Puckette. Adversarial audio synthesis. In *7th International Conference on Learning Representations (ICLR 2019)*. OpenReview.net, 2019b. URL https://openreview.net/forum?id=ByMVTsR5KQ.

J. Driedger, T. Prätzlich, and M. Müller. Let it Bee – towards NMF-inspired audio mosaicing. In *Proceedings of the 16th International Society for Music Information Retrieval Conference*. ISMIR, Oct. 2015. doi: 10.5281/zenodo.1415698. URL https://doi.org/10.5281/zenodo.1415698.

K. Ebcioğlu. An expert system for harmonizing chorales in the style of J. S. Bach. *The Journal of Logic Programming*, 8(1):145–185, 1990. ISSN 0743-1066. doi: https://doi.org/10.1016/0743-1066(90)90055-A. URL https://www.sciencedirect.com/science/article/pii/074310669090055A.

D. Eck and J. Schmidhuber. Finding temporal structure in music: blues improvisation with LSTM recurrent networks. In *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing (NNSP 2002)*. IEEE, 2002. doi: 10.1109/NNSP.2002.1030094. URL https://doi.org/10.1109/NNSP.2002.1030094.

A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2001)*. ACM, 2001. URL https://dl.acm.org/citation.cfm?id=383296.

J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan. Neural audio synthesis of musical notes with WaveNet autoencoders. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017. URL https://proceedings.mlr.press/v70/engel17a.html.

J. H. Engel, L. Hantrakul, C. Gu, and A. Roberts. DDSP: Differentiable digital signal processing. In *8th International Conference on Learning Representations, (ICLR 2020)*. OpenReview.net, 2020. URL https://openreview.net/forum?id=B1x1ma4tDr.

A. Fan, M. Lewis, and Y. Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-1082. URL https://www.aclweb.org/anthology/P18-1082.

E. Frid, C. Gomes, and Z. Jin. Music creation by example. In *CHI '20: CHI Conference on Human Factors in Computing Systems*. ACM, 2020. doi: 10.1145/3313831.3376514. URL https://doi.org/10.1145/3313831.3376514.

L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.265. URL https://doi.org/10.1109/CVPR.2016.265.

J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017. URL http://proceedings.mlr.press/v70/gehring17a.html.

M. G. Genton and W. Kleiber. Cross-covariance functions for multivariate geostatistics. *Statistical Science*, pages 147–163, 2015.

S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *6th International Conference on Learning Representations (ICLR 2018)*. OpenReview.net, 2018. URL https://openreview.net/forum?id=S1v4N2l0-.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014. URL https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html.

M. Grachten, S. Lattner, and E. Deruty. BassNet: A variational gated autoencoder for conditional generation of bass guitar tracks with learned interactive control. *Applied Sciences*, 10(18), 2020. ISSN 2076-3417. doi: 10.3390/app10186627. URL https://www.mdpi.com/2076-3417/10/18/6627.

D. W. Griffin and J. S. Lim. Signal estimation from modified short-time Fourier transform. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP '83)*. IEEE, 1983. doi: 10.1109/ICASSP.1983.1172092. URL https://doi.org/10.1109/ICASSP.1983.1172092.

E. Grinstein, N. Q. K. Duong, A. Ozerov, and P. Pérez. Audio style transfer. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2018)*. IEEE, 2018. doi: 10.1109/ICASSP.2018.8461711. URL https://doi.org/10.1109/ICASSP.2018.8461711.

G. Hadjeres, J. Sakellariou, and F. Pachet. Style imitation and chord invention in polyphonic music with exponential families. *CoRR*, abs/1609.05152, 2016. URL http://arxiv.org/abs/1609.05152.

G. Hadjeres, F. Pachet, and F. Nielsen. DeepBach: A steerable model for Bach chorales generation. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017. URL https://proceedings.mlr.press/v70/hadjeres17a.html.

S. H. Hakimi, N. Bhonker, and R. El-Yaniv. BebopNet: Deep neural models for personalized jazz improvisations. In *Proceedings of the 21st International Society for Music Information Retrieval Conference*. ISMIR, 2020. doi: 10.5281/zenodo.4245562. URL https://doi.org/10.5281/zenodo.4245562.

T. B. Hashimoto, H. Zhang, and P. Liang. Unifying human and statistical evaluation for natural language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2019)*, 2019.

P. He, X. Liu, J. Gao, and W. Chen. DeBERTa: Decoding-enhanced BERT with disentangled attention. In *9th International Conference on Learning Representations (ICLR 2021)*. OpenReview.net, 2021. URL https://openreview.net/forum?id=XPZIaotutsD.

D. Herremans and E. Chew. MorpheuS: Generating structured music with constrained patterns and tension. *IEEE Transactions on Affective Computing*, 10(4):510–523, 2017.

I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations (ICLR 2017)*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Sy2fzU9gl.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735–1780, 1997.

E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*. Springer, 2015.

A. Holtzman, J. Buys, M. Forbes, A. Bosselut, D. Golub, and Y. Choi. Learning to write with cooperative discriminators. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-1152. URL https://www.aclweb.org/anthology/P18-1152.

A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi. The curious case of neural text degeneration. In *8th International Conference on Learning Representations (ICLR 2020)*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

C. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck. Music transformer: Generating music with long-term structure. In *7th International Conference on Learning Representations (ICLR 2019)*. OpenReview.net, 2019a. URL https://openreview.net/forum?id=rJe4ShAcF7.

C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. C. Courville, and D. Eck. Counterpoint by convolution. In *Proceedings of the 18th International Society for Music Information Retrieval Conference*. ISMIR, 2017. doi: 10.5281/zenodo.1416370. URL https://doi.org/10.5281/zenodo.1416370.

C.-Z. A. Huang, H. V. Koops, E. Newton-Rex, M. Dinculescu, and C. Cai. AI song contest: Human-AI co-creation in songwriting. In *Proceedings of the 21st International Society for Music Information Retrieval Conference*. ISMIR, 2020. doi: 10.5281/zenodo.4245530. URL https://doi.org/10.5281/zenodo.4245530.

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 *7th International Conference on Learning Representations (ICLR 2019)*. OpenReview.net, 2019b. URL https://openreview.net/forum?id=S1lvm305YQ.

Y.-S. Huang and Y.-H. Yang. Pop music Transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proceedings of the 28th ACM International Conference on Multimedia*. Association for Computing Machinery, 2020. doi: 10.1145/3394171.3413671. URL https://doi.org/10.1145/3394171.3413671.

Y. Hung, I. Chiang, Y. Chen, and Y. Yang. Musical composition style transfer via disentangled timbre representations. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI 2019)*. ijcai.org, 2019. doi: 10.24963/ijcai.2019/652. URL https://doi.org/10.24963/ijcai.2019/652.

K. Irie, A. Zeyer, R. Schlüter, and H. Ney. Language modeling with deep Transformers. In *20th Annual Conference of the International Speech Communication Association (INTERSPEECH 2019)*. ISCA, 2019. doi: 10.21437/Interspeech.2019-2225. URL https://doi.org/10.21437/Interspeech.2019-2225.

P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, (CVPR 2017)*. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.632. URL https://doi.org/10.1109/CVPR.2017.632.

J. Jiang, G. G. Xia, D. B. Carlton, C. N. Anderson, and R. H. Miyakawa. Transformer VAE: A hierarchical model for structure-aware and interpretable music representation learning. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020)*, 2020. doi: 10.1109/ICASSP40776.2020.9054554.

Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song. Neural style transfer: A review. *IEEE Transactions on Visualization and Computer Graphics*, 2020. doi: 10.1109/TVCG.2019.2921336. URL https://doi.org/10.1109/TVCG.2019.2921336.

M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, 2017.

T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2019)*. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00453. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Karras_A_Style-Based_Generator_Architecture_for_Generative_Adversarial_Networks_CVPR_2019_paper.html.

A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

L. Kawai, P. Esling, and T. Harada. Attributes-aware deep music transformation. In *Proceedings of the 21st International Society for Music Information Retrieval Conference*. ISMIR, 2020. doi: 10.5281/zenodo.4245520. URL https://doi.org/10.5281/zenodo.4245520.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR 2015)*, 2015. URL http://arxiv.org/abs/1412.6980.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations (ICLR 2014)*, 2014. URL http://arxiv.org/abs/1312.6114.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2016.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

D. Kucerovsky, K. Mousavand, and A. Sarraf. On some properties of Toeplitz matrices. *Cogent Mathematics*, 3(1), 2016. doi: 10.1080/23311835.2016.1154705. URL http://doi.org/10.1080/23311835.2016.1154705.

G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, and M. Ranzato. Fader networks: Manipulating images by sliding attributes. In *Advances in Neural Information Processing Systems*, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3fd60983292458bf7dee75f12d5e9e05-Abstract.html.

G. Lample, A. Conneau, L. Denoyer, and M. Ranzato. Unsupervised machine translation using monolingual corpora only. In *6th International Conference on Learning Representations (ICLR 2018)*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rkYTTf-AZ.

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.

K. Lee and M. Slaney. Acoustic chord transcription and key extraction from audio using key-dependent HMMs trained on synthesized audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 16:291–301, 2008.

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

W. T. Lu and L. Su. Transferring the style of homophonic music using recurrent neural networks and autoregressive models. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*. ISMIR, 2018. doi: 10.5281/zenodo.1492523. URL https://doi.org/10.5281/zenodo.1492523.

T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2015. doi: 10.18653/v1/D15-1166. URL https://www.aclweb.org/anthology/D15-1166.

A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference*. The Association for Computer Linguistics, 2011. URL https://aclanthology.org/P11-1015/.

A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models, 2013. URL https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf.

L. v. d. Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

I. Malik and C. H. Ek. Neural translation of musical style. *CoRR*, abs/1708.03535, 2017. URL http://arxiv.org/abs/1708.03535.

P. Manocha, A. Finkelstein, R. Zhang, N. J. Bryan, G. J. Mysore, and Z. Jin. A differentiable perceptual audio metric learned from just noticeable differences. In *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2020.

G. Matheron. Principles of geostatistics. *Economic geology*, 58(8):1246–1266, 1963.

M. Mauch and S. Dixon. PYIN: a fundamental frequency estimator using probabilistic threshold distributions. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2014)*. IEEE, 2014. doi: 10.1109/ICASSP.2014.6853678. URL https://doi.org/10.1109/ICASSP.2014.6853678.

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

C. McKay and I. Fujinaga. The Bodhidharma system and the results of the MIREX 2005 symbolic genre classification contest, 2005. URL http://jmir.sourceforge.net/publications/ISMIR_2005_MIREX_Symbolic.pdf.

MIDI Manufacturers Association. General MIDI system level 1, 1991. URL https://www.midi.org/specifications-old/item/general-midi.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations (ICLR 2013)*, 2013. URL http://arxiv.org/abs/1301.3781.

N. Mor, L. Wolf, A. Polyak, and Y. Taigman. A universal music translation network. In *7th International Conference on Learning Representations (ICLR 2019)*, 2019. URL https://openreview.net/forum?id=HJGkisCcKm.

A. Muhamed, L. Li, X. Shi, S. Yaddanapudi, W. Chi, D. Jackson, R. Suresh, Z. C. Lipton, and A. J. Smola. Symbolic music generation with Transformer-GANs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/16117.

N. Nangia and S. R. Bowman. ListOps: A diagnostic dataset for latent tree learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2018)*. Association for Computational Linguistics, 2018. doi: 10.18653/v1/n18-4013. URL https://doi.org/10.18653/v1/n18-4013.

S. Nercessian. Zero-shot singing voice conversion. In *Proceedings of the 21st International Society for Music Information Retrieval Conference*. ISMIR, 2020. doi: 10.5281/zenodo.4245370. URL https://doi.org/10.5281/zenodo.4245370.

J. Nistal, S. Lattner, and G. Richard. DarkGAN: Exploiting knowledge distillation for comprehensible audio synthesis with gans. *CoRR*, abs/2108.01216, 2021. URL https://arxiv.org/abs/2108.01216.

D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.278. URL https://doi.org/10.1109/CVPR.2016.278.

C. Payne. MuseNet. OpenAI, 2019. URL https://openai.com/blog/musenet/.

G. Peeters and G. Richard. Deep Learning for Audio and Music. In *Multi-faceted Deep Learning: Models and Data*. Springer, 2021.

J. Pons, S. Pascual, G. Cengarle, and J. Serrà. Upsampling artifacts in neural audio synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2021)*. IEEE, 2021. doi: 10.1109/ICASSP39728.2021.9414913. URL https://doi.org/10.1109/ICASSP39728.2021.9414913.

L. Prétet, G. Richard, and G. Peeters. Cross-modal music-video recommendation: A study of design choices. In *Special Session of the International Joint Conference on Neural Networks (IJCNN 2021)*, 2021. URL https://hal.telecom-paris.fr/hal-03208323.

D. R. Radev, P. Muthukrishnan, V. Qazvinian, and A. Abu-Jbara. The ACL anthology network corpus. *Language Resources and Evaluation*, 47(4):919–944, Jan. 2013. doi: 10.1007/s10579-012-9211-2. URL https://doi.org/10.1007/s10579-012-9211-2.

A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training, 2018. URL https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners, 2019. URL https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

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

D. Rezende, S. Mohamed, I. Danihelka, K. Gregor, and D. Wierstra. One-shot generalization in deep generative models. In *Proceedings of The 33rd International Conference on Machine Learning*. PMLR, 2016. URL https://proceedings.mlr.press/v48/rezende16.html.

G. Richard, S. Sundaram, and S. Narayanan. An overview on perceptually motivated audio indexing and classification. *Proceedings of the IEEE*, 101(9):1939–1954, 2013. doi: 10.1109/JPROC.2013.2251591.

A. Roberts. YACHT's new album is powered by ML + artists. Magenta Blog, 2019. URL https://magenta.tensorflow.org/chain-tripping.

A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck. A hierarchical latent vector model for learning long-term structure in music. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018. URL https://proceedings.mlr.press/v80/roberts18a.html.

A. Roberts, J. Engel, Y. Mann, J. Gillick, C. Kayacik, S. Nørly, M. Dinculescu, C. Radebaugh, C. Hawthorne, and D. Eck. Magenta Studio: Augmenting creativity with deep learning in Ableton Live. In *Proceedings of the 6th International Workshop on Musical Metacreation*. MUME, 2019. doi: 10.5281/zenodo.4285266. URL https://doi.org/10.5281/zenodo.4285266.

G. Ros, L. Sellart, J. Materzynska, D. Vázquez, and A. M. López. The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.352. URL https://doi.org/10.1109/CVPR.2016.352.

J. Rosendahl, V. A. K. Tran, W. Wang, and H. Ney. Analysis of positional encodings for neural machine translation. In *16th International Workshop on Spoken Language Translation (IWSLT)*, 2019. doi: 10.5281/zenodo.3525024. URL https://doi.org/10.5281/zenodo.3525024.

A. Roy, M. Saffar, A. Vaswani, and D. Grangier. Efficient content-based sparse attention with routing Transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021. doi: 10.1162/tacl_a_00353. URL https://aclanthology.org/2021.tacl-1.4.

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

J. Salamon and E. Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Speech and Audio Processing*, 20(6):1759–1770, 2012. doi: 10.1109/TASL.2012.2188515. URL https://doi.org/10.1109/TASL.2012.2188515.

J. Salamon, R. M. Bittner, J. Bonada, J. J. Bosch, E. Gómez, and J. P. Bello. An analysis/synthesis framework for automatic F0 annotation of multitrack datasets. In *Proceedings of the 18th International Society for Music Information Retrieval Conference*. ISMIR, 2017. doi: 10.5281/zenodo.1415588. URL https://doi.org/10.5281/zenodo.1415588.

M. N. Schmidt and M. Mørup. Nonnegative matrix factor 2-D deconvolution for blind single channel source separation. In *Independent Component Analysis and Blind Signal Separation, 6th International Conference (ICA)*. Springer, 2006. doi: 10.1007/11679363_87. URL https://doi.org/10.1007/11679363_87.

M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

P. Shaw, J. Uszkoreit, and A. Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. Association for Computational Linguistics, 2018. doi: 10.18653/v1/n18-2074. URL https://doi.org/10.18653/v1/n18-2074.

Z. Shen, M. Zhang, H. Zhao, S. Yi, and H. Li. Efficient attention: Attention with linear complexities. In *IEEE Winter Conference on Applications of Computer Vision (WACV 2021)*. IEEE, 2021. doi: 10.1109/WACV48630.2021.00357. URL https://doi.org/10.1109/WACV48630.2021.00357.

I. Simon and S. Oore. Performance RNN: Generating music with expressive timing and dynamics. Magenta Blog, 2017. URL https://magenta.tensorflow.org/performance-rnn.

I. Simon, D. Morris, and S. Basu. MySong: automatic accompaniment generation for vocal melodies. In *Proceedings of the 2008 Conference on Human Factors in Computing Systems (CHI 2008)*. ACM, 2008. doi: 10.1145/1357054.1357169. URL https://doi.org/10.1145/1357054.1357169.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations (ICLR 2015)*, 2015. URL http://arxiv.org/abs/1409.1556.

B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova. Music transcription modelling and composition using deep learning. *CoRR*, abs/1604.08723, 2016. URL http://arxiv.org/abs/1604.08723.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 2014. URL https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html.

Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler. Long Range Arena: A benchmark for efficient Transformers. In *9th International Conference on Learning Representations (ICLR 2021)*. OpenReview.net, 2021. URL https://openreview.net/forum?id=qVyeW-grC2k.

J. Thickstun, Z. Harchaoui, D. Foster, and S. Kakade. Coupled recurrent models for polyphonic music composition. In *Proceedings of the 20th International Society for Music Information Retrieval Conference*. ISMIR, 2019. doi: 10.5281/zenodo.3527806. URL https://doi.org/10.5281/zenodo.3527806.

C. J. Tralie. Cover song synthesis by analogy. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*. ISMIR, 2018. doi: 10.5281/zenodo.1492381. URL https://doi.org/10.5281/zenodo.1492381.

Y. H. Tsai, S. Bai, M. Yamada, L. Morency, and R. Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP 2019)*. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1443. URL https://doi.org/10.18653/v1/D19-1443.

D. Ulyanov and V. Lebedev. Audio texture synthesis and style transfer, 2016. URL https://dmitryulyanov.github.io/audio-texture-synthesis-and-style-transfer/.

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 *The 9th ISCA Speech Synthesis Workshop (SSW)*. ISCA, 2016. URL http://www.isca-speech.org/archive/SSW_2016/abstracts/ssw9_DS-4_van_den_Oord.html.

A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/7a98af17e63a0ac09ce2e96d03992fbc-Abstract.html.

G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid. Learning from synthetic humans. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.492. URL https://doi.org/10.1109/CVPR.2017.492.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

M. Vetterli, J. Kovačević, and V. K. Goyal. *Foundations of signal processing*. Cambridge University Press, 2014.

M. Vořechovský. Simulation of simply cross correlated random fields by series expansion methods. *Structural safety*, 30(4):337–363, 2008.

B. Wang, L. Shang, C. Lioma, X. Jiang, H. Yang, Q. Liu, and J. G. Simonsen. On position embeddings in BERT. In *9th International Conference on Learning Representations (ICLR 2021)*. OpenReview.net, 2021. URL https://openreview.net/forum?id=onxoVA9FxMw.

S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768, 2020a. URL https://arxiv.org/abs/2006.04768.

Z. Wang, D. Wang, Y. Zhang, and G. Xia. Learning interpretable representation for controllable polyphonic music generation. In *Proceedings of the 21st International Society for Music Information Retrieval Conference*. ISMIR, 2020b. doi: 10.5281/zenodo.4245518. URL https://doi.org/10.5281/zenodo.4245518.

D. Wei, J. J. Lim, A. Zisserman, and W. T. Freeman. Learning and using the arrow of time. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2018)*. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00840. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Wei_Learning_and_Using_CVPR_2018_paper.html.

G. Widmer, S. Flossmann, and M. Grachten. YQX plays Chopin. *AI Magazine*, 30: 35–48, 2009.

C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Q. Wu, Z. Lan, J. Gu, and Z. Yu. Memformer: The memory-augmented Transformer. *CoRR*, abs/2010.06891, 2020. URL https://arxiv.org/abs/2010.06891.

S. Wu and Y. Yang. MuseMorphose: Full-song and fine-grained music style transfer with just one Transformer VAE. *CoRR*, abs/2105.04090, 2021. URL https://arxiv.org/abs/2105.04090.

S.-L. Wu and Y.-H. Yang. The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures. In *Proceedings of the 21st International Society for Music Information Retrieval Conference*. ISMIR, 2020. doi: 10.5281/zenodo.4245390. URL https://doi.org/10.5281/zenodo.4245390.

G. G. Xia and S. Dai. Music style transfer: A position paper. In *Proceedings of the 6th International Workshop on Musical Metacreation (MUME 2018)*, 2018.

X. Xie, F. Tian, and H. S. Seah. Feature guided texture synthesis (FGTS) for artistic style transfer. In *Proceedings of the Second International Conference on Digital Interactive Media in Entertainment and Arts (DIMEA 2007)*. ACM, 2007. doi: 10.1145/1306813.1306830. URL https://doi.org/10.1145/1306813.1306830.

H. Xue and F. D. Salim. TRAILER: Transformer-based time-wise long term relation modeling for citywide traffic flow prediction. *CoRR*, abs/2011.05554, 2020. URL https://arxiv.org/abs/2011.05554.

R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia. Deep music analogy via latent representation disentanglement. In *Proceedings of the 20th International Society for Music Information Retrieval Conference*. ISMIR, 2019. doi: 10.5281/zenodo.3527880. URL https://doi.org/10.5281/zenodo.3527880.

Z. Yang, L. Xie, and P. Stoica. Vandermonde decomposition of multilevel Toeplitz matrices with application to multidimensional super-resolution. *IEEE Transactions on Information Theory*, 62(6):3685–3701, 2016.

P. Zhou, R. Fan, W. Chen, and J. Jia. Improving generalization of Transformer for speech recognition with parallel schedule sampling and relative positional embedding. *CoRR*, abs/1911.00203, 2019. URL http://arxiv.org/abs/1911.00203.

J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV 2017)*. IEEE Computer Society, 2017. doi: 10.1109/ICCV.2017.244. URL https://doi.org/10.1109/ICCV.2017.244.

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

A. Miscellaneous details

A.1 Chord chart generation

This section is adopted and modified from the paper ⟨**TASLP2020**⟩ by Cífka et al.

To obtain the chord charts mentioned in Section 4.4.1, we sample from a chord language model (LM) estimated on the iRb corpus [Broze and Shanahan, 2013], 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 extract the necessary features using the `jazzparser.sh` script provided by Broze and Shanahan. We separate songs in major keys from songs in minor keys, and train a smoothed bigram LM (using Lidstone or add-ε smoothing with $\varepsilon = 0.01$) 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 assumed that it would be sufficiently diverse to cover most styles thanks to the harmonic variability of jazz, and the LM smoothing. Nevertheless, we acknowledge that expanding the dataset to chord charts from other genres could provide some benefits.

We sample sequences from the LM repeatedly and concatenate them until we reach the maximum number of measures. We then choose the key of the chord chart at random (following the distribution of keys in iRb), and convert each token to a chord symbol according to the chosen key.

We optionally add some of the following modifiers (defined by BIAB) to each chord: (a) *push*: creates an 8^{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. (See e.g. bar 4 in Fig. 4.2, featuring a G minor 7^{th} chord with shot and push modifiers.) These modifiers are turned on and off at random, with probabilities chosen so that they are scattered sparsely throughout the chord charts. This kind of data enhancement is necessary in order for the trained system to be able to handle such rhythmic variations; notably, we observed that the model from Section 4.3, which was trained without using this technique, always produces continuous accompaniments even when the inputs contain prominent breaks.

A.2 Long-Range Arena results

This section is based on the paper ⟨**ICML2021**⟩ by Liutkus, Cífka (equal contribution) et al.

We evaluate the proposed stochastic positional encoding (SPE) in the Long-Range Arena (LRA, [Tay et al., 2021]), a benchmark for efficient Transformers, consisting of sequence classification tasks with a focus on long-range dependencies. We use the following tasks from this benchmark:

- *ListOps*: parsing and evaluation of hierarchical expressions. a longer variant of Nangia and Bowman [2018];
- *Text*: movie review sentiment analysis on the IMDB corpus [Maas et al., 2011];
- *Retrieval*: article similarity classification on the ACL Anthology Network (AAN) corpus [Radev et al., 2013];
- *Image*: object recognition on the CIFAR10 dataset [Krizhevsky, 2009] represented as pixel sequences.

The tasks are challenging due to the large sequence lengths, deliberately increased by choosing a character-/pixel-level representation. More details on the tasks are given in Table A.1. We do not include *Pathfinder* (a synthetic image classification task) as we were unable to reproduce the results of Tay et al. on this task, even through correspondence with the authors.

We evaluate SPE (the gated variant) on two efficient Transformer models: the (softmax) Performer [Choromanski et al., 2021], and a Linear Transformer [Katharopoulos et al., 2020] with a ReLU feature map, i.e. choosing $\phi(\cdot) = \max(0, \cdot)$ element-wise in Eq. (5.6).[1] It should be noted that the ReLU feature map does not approximate the softmax kernel, which SPE is designed for (see Eq. (5.12)). Nevertheless, it is possible to use SPE with any feature map in practice, allowing us to include Linear Transformer-ReLU as an interesting test of generalization to alternative kernels.

We adopt the configuration of Tay et al., only changing the PE and the batch sizes/learning rates to allow training on limited hardware with similar results. All other hyperparameters are kept identical to the original LRA. It is worth noting that the *Image* models are different from the rest in that they employ a single-layer network and only use the first position for prediction, dramatically limiting their ability to benefit from relative positional information.

Since we observe some variation between different runs, we train and evaluate each model 3 times (except for Performer with convolutional SPE, which is computationally more costly) and report the mean and standard deviation of the results.

The results of the benchmark are given in Table A.2. The accuracies achieved by the baseline Linear Transformer-ReLU (`APE`) are similar to or surpass those reported by Tay et al., which is a clear validation of our experimental setup.

[1] A model named 'Performer' is reported by Tay et al., but communication with the authors revealed it to be in fact equivalent to our Linear Transformer-ReLU, as it does not use random features. To avoid confusion, we refer to this model as such herein.

Name	Dataset	Input	Length	Goal	# classes
ListOps	ListOps	expression with operations on lists of numbers 0–9	2 k	evaluate expression	10
Text	IMDB	movie review as byte string	8 k	classify sentiment	2
Retrieval	AAN	pair of articles as byte strings	2×4 k	detect citation link	2
Image	CIFAR10	8-bit gray-scale 32×32 image as byte string	1 k	recognize object	10

Table A.1 – Long-Range Arena classification tasks used in this work.

	ListOps	Text	Retrieval	Image
Best [Tay et al.]	37.27 (Reformer)	65.90 (Linear Trans.)	59.59 (Sparse Trans.)	44.24 (Sparse Trans.)
Lin. Trans.-ReLU [Tay et al.]	18.01	65.40	53.82	42.77
Performer-softmax (`APE`)	**17.80** $\pm$ 0.00	62.58 $\pm$ 0.22	59.84 $\pm$ 1.46	41.81 $\pm$ 1.16
Performer-softmax + `sineSPE`	17.43 $\pm$ 0.32	62.60 $\pm$ 0.50	60.00 $\pm$ 1.20	41.12 $\pm$ 1.70
Performer-softmax + `convSPE`	**17.80**	60.94	57.22	40.06
Lin. Trans.-ReLU (`APE`)	17.58 $\pm$ 1.01	63.98 $\pm$ 0.05	58.78 $\pm$ 0.93	**42.25** $\pm$ 0.01
Lin. Trans.-ReLU + `sineSPE`	**17.80** $\pm$ 0.00	**64.09** $\pm$ 0.62	**62.39** $\pm$ 0.59	41.21 $\pm$ 1.18
Lin. Trans.-ReLU + `convSPE`	9.50 $\pm$ 1.17	63.23 $\pm$ 1.31	61.00 $\pm$ 1.34	39.96 $\pm$ 1.31

Table A.2 – Long-Range Arena results (higher scores are better). Mean and standard deviation of accuracy over three runs is reported, except for Performer with convolutional SPE, where only a single run was completed. For comparison, the best result reported by Tay et al. [2021], along with the name of the best-performing model (in parentheses), is included.

Discussion. Results on ListOps are poor overall, with accuracies around 17 %. This complies with Tay et al. [2021], who reason that 'kernel-based models [e.g. Performer, Linear Transformers] are possibly not as effective on hierarchically structured data,' leaving room for improvement. We also hypothesize this is largely due to some issues with the training data for this task, which unfortunately were unknown to us and had not been fixed at the time of our experiments.[2]

Regarding performance of SPE, we first notice that the `sineSPE` variant yields the best results on three tasks, which is a strong achievement and validates our approach, especially considering the difficulty of this evaluation benchmark. While it is only marginally better than APE for *ListOps* and *Text*, it is worth mentioning that `sineSPE` combined with the Linear Transformer-ReLU yields an accuracy improvement of $\sim$3 % on *Retrieval* compared to the best result obtained by Tay et al. [2021].

Regarding `convSPE`, its performance in the LRA is not as remarkable. This mitigated result appears somewhat in contradiction with the discussion found in Wang et al. [2021], which presents vanishing attention as a desirable property of PE. On the contrary, we empirically observe that our non-vanishing sinusoidal version `sineSPE` behaves better in these particular tasks.

Finally, the superior results of `APE` on *Image* are not unexpected, given the limited ability of these models to exploit relative positions. On the contrary, the relatively good performance of SPE on this task is in fact remarkable, especially

[2]The official data loader for ListOps contained a bug that caused it to inadvertently strip some characters from the input sequences.

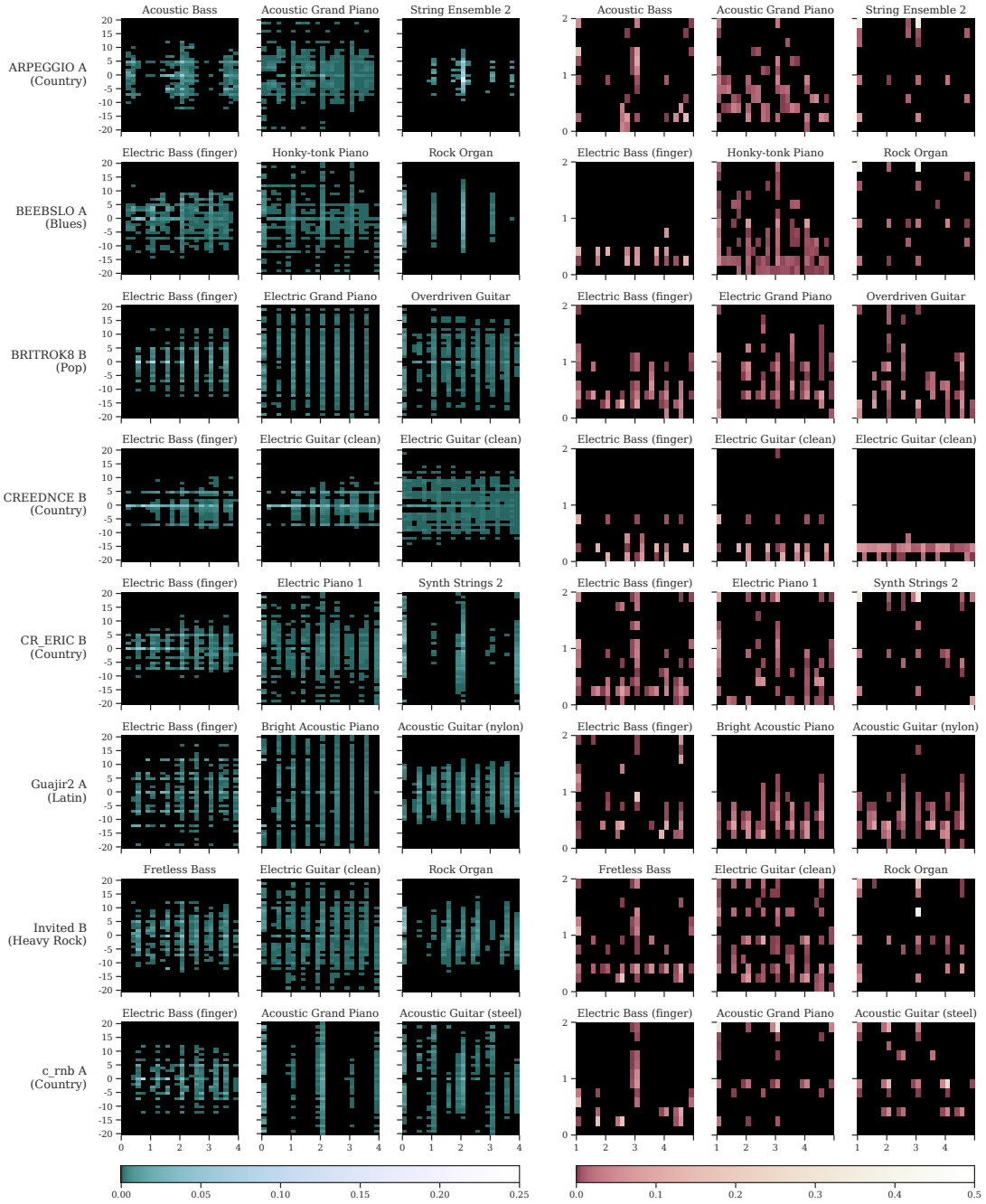
considering that the baseline systems for this task use *learnable* APE.

In sum, the performance of SPE is very much comparable to `APE` (which is already encouraging) and in some cases considerably better, but this boost in performance is not systematic. This raises interesting considerations:

(i) *The variance of the Monte Carlo estimator might be problematic.* We are enthusiastic about the elegant formulation of stochastic feature maps as in the Performer, which was a strong inspiration. Still, we must acknowledge that their computation relies on a Monte Carlo estimator (5.18). We suspect that the variance of the estimator might play a role in the final performance in large dimensions, which opens up the direction of exploring variance-reduced estimation methods, rather than plain Monte Carlo.

(ii) *LRA tasks might not benefit from strong (R)PE schemes.* The LRA was designed to compare Transformer *architectures*, filling a gap in this domain and standing as the *de facto* standard, justifying our choice. Still, although PE is known to be important in many cases, it is not known whether it is so in the LRA tasks. We feel that there is room for such a specialized comparison, which is scheduled in our future work, possibly leading to new long-range tasks where PE is critical.

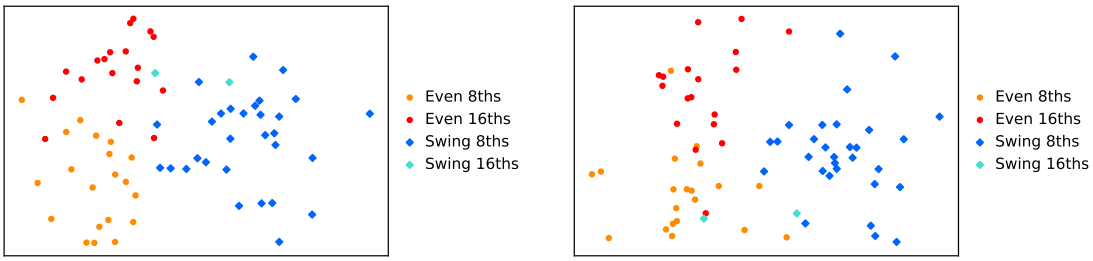
B. Additional figures



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

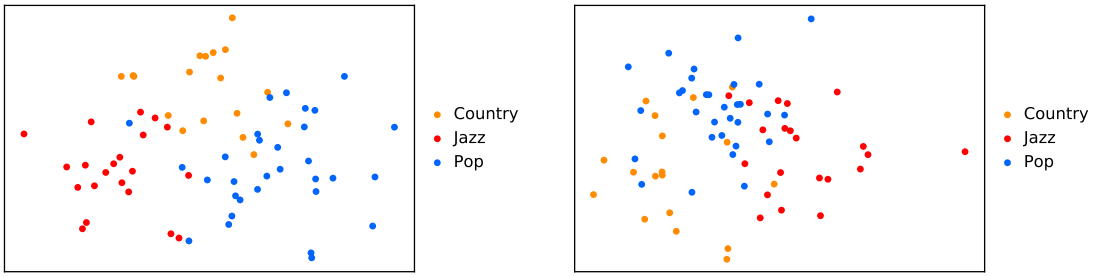
(b) Onset-duration (x: onset time, y: duration)

Figure B.1 – Examples of style profiles (see Section 4.2.2) computed on the synthetic test set from Section 4.4.1. 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. Reproduced from ⟨TASLP2020⟩. © 2020 IEEE.



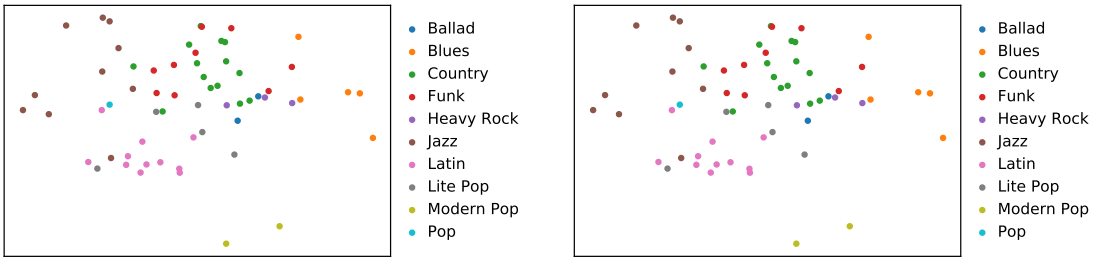
(a) ALL→BASS, feel

(b) ALL→PIANO, feel (same as Fig. 4.8)



(c) ALL→BASS, style group

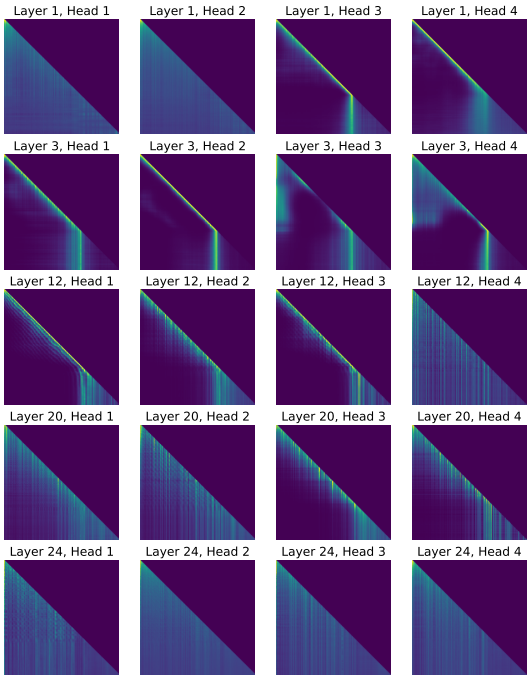
(d) ALL→PIANO, style group



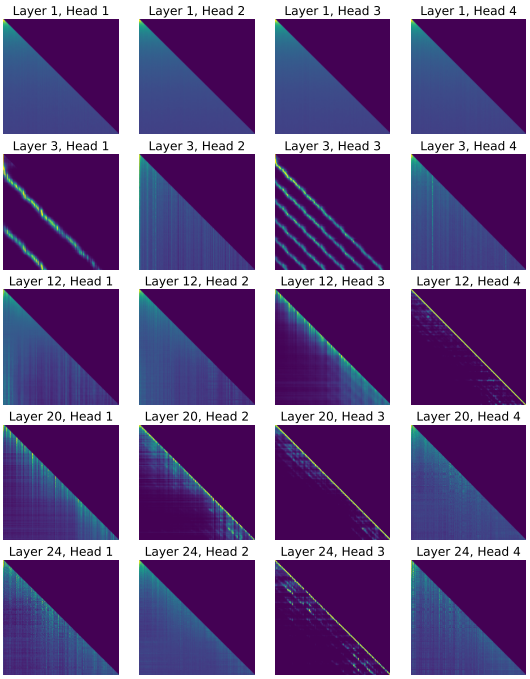
(e) ALL→BASS, genre

(f) ALL→PIANO, genre

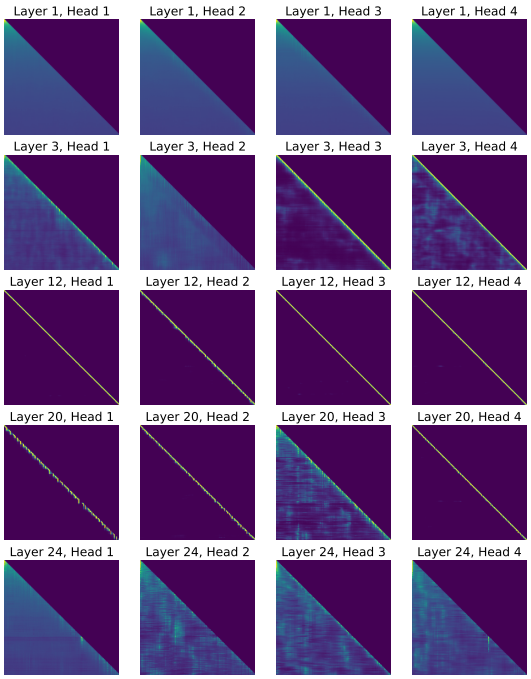
Figure B.2 – Style embeddings for two different models from Section 4.3 (ALL→BASS and ALL→PIANO) labeled with 'feel', 'style group' and genre annotations provided by BIAB. Dimensionality reduction was performed using linear discriminant analysis (LDA) with the annotations as targets. Reproduced from the supplementary material of ⟨ISMIR2019⟩.



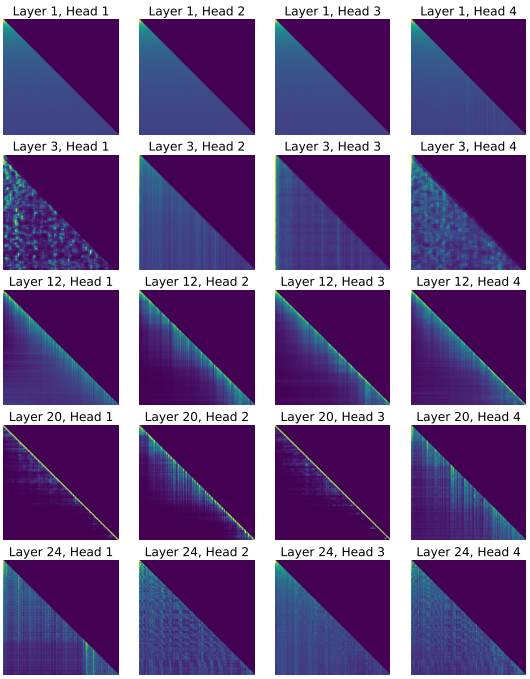
(a) `APE`



(b) `sineSPE` (gated)



(c) `convSPE` (ungated)



(d) `convSPE` (gated)

Figure B.3 – Attention matrices (first 4 heads only) from the pop piano generation experiment (Section 5.3.2). Max position = 2048. Reproduced and modified from the supplementary material of ⟨ICML2021⟩.

C. Supplementary materials

This appendix gathers references to all the additional resources that accompany the published articles and this thesis.

⟨**ISMIR2019**⟩ Supervised symbolic music style translation using synthetic data.

`https://ondrej.cifka.com/posts/neural-music-style-translation/`
– blog post with listening examples

`https://github.com/cifkao/ismir2019-music-style-translation`
– source code

`https://collegerama.tudelft.nl/Mediasite/Showcase/ismir2019/Presentation/f4dd611fbfa64c6da166efadce72a2611d`
– ISMIR 2019 oral presentation

`http://tiny.cc/musicstyle`
– listening examples

`http://doi.org/10.5281/zenodo.3250606`
– supplementary files

`https://bit.ly/2G5Jgnq`
– interactive style embedding visualization

`http://doi.org/10.5281/zenodo.3245374`
– trained model parameters

⟨**TASLP2020**⟩ Groove2Groove: One-shot music style transfer with supervision from synthetic data.

`https://groove2groove.telecom-paris.fr/`
– companion website with an interactive demo

`https://github.com/cifkao/groove2groove`
– source code

`https://youtu.be/nytHX412u5I`
– ICASSP 2021 oral presentation

`https://youtu.be/a4XyFeuRM8k`
– ICASSP 2021 demo video

`https://www.youtube.com/playlist?list=PLPdw6Kin7U86tcz-vlMmKqQmq4yL325aH`
– additional listening examples (videos)

`https://doi.org/10.5281/zenodo.3957999`
– synthetic accompaniment dataset (*Groove2Groove MIDI Dataset*)

`https://groove2groove.telecom-paris.fr/data/checkpoints/`
– trained model parameters

⟨**ICASSP2021**⟩ Self-supervised VQ-VAE for one-shot music style transfer.

`https://adasp.telecom-paris.fr/s/ss-vq-vae`
– companion website with listening examples

`https://github.com/cifkao/ss-vq-vae`
– source code

`https://youtu.be/pln-h9eclGc`
– ICASSP 2021 oral presentation

`https://colab.research.google.com/github/cifkao/ss-vq-vae/blob/main/experiments/colab_demo.ipynb`
– demo Colab notebook

`https://adasp.telecom-paris.fr/rc-ext/demos_companion-pages/vqvae_examples/ssvqvae_model_state.pt`
– trained model parameters

⟨**ICML2021**⟩ Relative positional encoding for Transformers with linear complexity.

`https://cifkao.github.io/spe/`
– companion website with listening examples

`http://proceedings.mlr.press/v139/liutkus21a/liutkus21a-supp.pdf`
– supplementary document (appendix)

`https://github.com/aliutkus/spe`
– source code

`https://slideslive.com/38958574`
– ICML 2021 oral presentation

`https://cifkao.github.io/metrical-pe/`
– listening examples from Section 5.4

D. Software packages

Besides the source code associated with the published papers, I have developed and released the following open-source software packages in connection to my work on this thesis:

HTML MIDI Player Web component for playing and displaying MIDI files.

The package provides the `<midi-player>` HTML element, which allows embedding MIDI files in web pages, similarly to how audio files can be inserted using the `<audio>` element. Moreover, real-time visualization (piano roll or score) is possible using the `<midi-visualizer>` element. Both functionalities are powered by the Magenta.js package.[1]

`https://cifkao.github.io/html-midi-player/`
`https://github.com/cifkao/html-midi-player`

NoPdb Non-interactive (programmatic) Python debugger.

This Python package provides an easy-to-use API for debugger-like functionality. With NoPdb, it is possible to write Python one-liners to:

- capture function calls, including arguments, local variables, return values and stack traces;
- set 'breakpoints' that trigger certain pre-defined actions, namely evaluating expressions or executing arbitrary code with local side effects.

This makes it a convenient tool for inspecting machine learning model internals such as Transformer attention matrices.

`https://github.com/cifkao/nopdb`

Confugue Hierarchical configuration framework.

Confugue is a Python package providing a wrapper class for nested configuration dictionaries (usually loaded from YAML files), which can be used to easily configure complicated object hierarchies.

The package is ideal for configuring deep learning experiments. These often have large numbers of hyperparameters and managing all their values globally can quickly get tedious. Instead, Confugue allows each part of the deep learning model to be automatically supplied with hyperparameters from a YAML configuration file, eliminating the need to pass them around or to define the same hyperparameter in multiple places. The nested structure of the configuration file follows the hierarchy of the model architecture, with each part of the model having access to the corresponding section of the file.

`https://github.com/cifkao/confugue`

[1] `https://github.com/magenta/magenta-js`

DOCTORAL SCHOOL

Titre : Méthodes d'apprentissage profond pour le transfert de style musical

Mots clés : apprentissage machine, transfert de style, transformation de musique, génération de musique, synthèse sonore, réseaux de neurones

Résumé : Récemment, les méthodes d'apprentissage profond ont permis d'effectuer des transformations du matériel musical basées sur les données (*data-driven*). L'objet de cette thèse est le *transfert de style musical*, dont le but est de transférer de manière automatique le style d'un morceau à un autre.

Dans la première partie de ce travail, nous nous concentrons sur les méthodes supervisées pour le transfert de style des *accompagnements* dans une représentation *symbolique*, visant à transformer un morceau donné en lui générant un nouvel accompagnement. La méthode proposée est basée sur l'apprentissage supervisé de séquence à séquence à l'aide de réseaux de neurones récurrents (RNN) et s'appuie sur une base de données synthétiques *parallèle* (alignée par paires) générée à cet effet à l'aide d'un logiciel de génération d'accompagnement existant. Nous proposons ainsi un ensemble de mesures objectives pour évaluer la performance sur cette nouvelle tâche et nous montrons que le système réussit à générer un accompagnement dans le style souhaité tout en suivant la structure harmonique de l'entrée.

Dans la deuxième partie, nous étudions une question plus fondamentale : le rôle des *encodages positionnels* (*PE*) dans la génération de musique à l'aide des *Transformers*. Nous proposons *l'encodage positionnel stochastique* (*SPE*), un nouveau PE capable de coder des *positions relatives* et compatible avec une classe récemment proposée de Transformers efficaces. Nous démontrons que le SPE permet, mieux que la méthode conventionnelle (*le PE absolu*), de modéliser des séquences plus longues que celles rencontrées pendant l'entraînement.

Enfin, dans la troisième partie, nous passons de la musique symbolique à l'audio et abordons le problème du *transfert de timbre*. Plus précisément, nous nous intéressons à transférer le timbre d'un enregistrement audio à un autre, tout en préservant le contenu mélodique et harmonique de ce dernier. Nous présentons une nouvelle méthode pour cette tâche, basée sur une extension de *l'autoencodeur variationnel quantifié* (VQ-VAE), ainsi qu'une stratégie *d'apprentissage auto-supervisé* conçue pour obtenir des représentations démêlées du timbre et de la hauteur. Comme dans la première partie, nous concevons un ensemble de métriques objectives pour la tâche. Nous montrons que la méthode proposée est capable de surpasser des méthodes existantes.

Title: Deep learning methods for music style transfer

Keywords: machine learning, style transfer, music transformation, sound synthesis, music generation, neural networks

Abstract: Recently, deep learning methods have enabled transforming musical material in a data-driven manner. The focus of this thesis is on a family of tasks which we refer to as *(one-shot) music style transfer*, where the goal is to transfer the style of one musical piece or fragment onto another.

In the first part of this work, we focus on supervised methods for *symbolic music accompaniment style transfer*, aiming to transform a given piece by generating a new accompaniment for it in the style of another piece. The method we have developed is based on supervised sequence-to-sequence learning using recurrent neural networks (RNNs) and leverages a synthetic *parallel* (pairwise aligned) dataset generated for this purpose using existing accompaniment generation software. We propose a set of objective metrics to evaluate the performance on this new task and we show that the system is successful in generating an accompaniment in the desired style while following the harmonic structure of the input.

In the second part, we investigate a more basic question: the role of *positional encodings* (*PE*) in music generation using *Transformers*. In particular, we propose *stochastic positional encoding* (*SPE*), a novel form of PE capturing *relative positions* while being compatible with a recently proposed family of efficient Transformers. We demonstrate that SPE allows for better extrapolation beyond the training sequence length than the commonly used *absolute PE*.

Finally, in the third part, we turn from symbolic music to audio and address the problem of *timbre transfer*. Specifically, we are interested in transferring the timbre of an audio recording of a single musical instrument onto another such recording while preserving the pitch content of the latter. 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. As in the first part, we design a set of objective metrics for the task. We show that the proposed method is able to outperform existing ones.

Institut Polytechnique de Paris
91120 Palaiseau, France