Latent Space Interpolation of Synthesizer Parameters using Timbre-Regularized Auto-Encoders

Gwendal Le Vaillant and Thierry Dutoit

***Abstract*—Sound synthesizers are ubiquitous in modern music production but manipulating their presets, i.e. the sets of synthesis parameters, demands expert skills. This study presents a novel variational auto-encoder model tailored for black-box synthesizer preset interpolation, which enables the intuitive generation of new presets from pre-existing ones. Leveraging multi-head self-attention networks, the model efficiently learns latent representations of synthesis parameters, aligning these with perceived timbre dimensions through attribute-based regularization. It is able to gradually transition between diverse presets, surpassing traditional linear parametric interpolation methods. Furthermore, we introduce an objective and reproducible evaluation method, based on linearity and smoothness metrics computed on a broad set of audio features. The model's efficacy is demonstrated through subjective experiments, whose results also highlight significant correlations with the proposed objective metrics. The model is validated using a widespread frequency modulation synthesizer with a large set of interdependent parameters. It can be adapted to various commercial synthesizers, and can perform other tasks such as modulations and extrapolations.**

***Index Terms*—synthesizer, interpolation, preset, black-box, morphing, auto-encoder, timbre.**

I. Introduction

Sound synthesizers produce a wide range of audio signals, from emulating acoustic instruments to creating entirely new sonic textures. These instruments, essential in contemporary music production, have even shaped new music genres. A set of synthesis parameters, called a preset, is nonetheless often extensive and complex. Presets usually control low-level signal characteristics which do not easily correlate to the perceived dimensions of sound [1], [2]. Their creation and manipulation require expert skills and knowledge [3]. Therefore, synthesizer manufacturers and developers must provide a substantial amount of presets to their users.

Many recent works have used neural networks to match synthesis parameters with input sounds [1], [4]–[9]. This study rather focuses on preset interpolation. It aims to provide a model for computing non-linear interpolations in the domain of synthesis parameters of a black-box synthesizer. A good interpolation example is provided in Fig. 1(b), where spectro-temporal characteristics change gradually from one step to the next. Intermediate presets should mix the characteristics of the starting and ending presets in a way that goes beyond a mere crossfade in the audio waveform domain, enabling a more complex and nuanced blending of sounds [10]. Interpolation enables the easy and intuitive generation of presets in-between existing ones. Moreover, it allows producers and musicians to perform real-time morphing, and sound designers to explore the sonic possibilities of a synthesizer.

The authors are with the Information, Signal and Artificial Intelligence (ISIA) lab, University of Mons, Boulevard Dolez 31, 7000 Mons, Belgium. E-mail: { gwendal.levaillant, thierry.dutoit }@umons.ac.be. Gwendal Le Vaillant is also with Institut Supérieur Industriel de Bruxelles (ISIB), HE2B, Rue Royale 150, 1000 Brussels, Belgium.

This article has been accepted for publication in IEEE/ACM Transactions on Audio, Speech and Language Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TASLP.2024.3426987

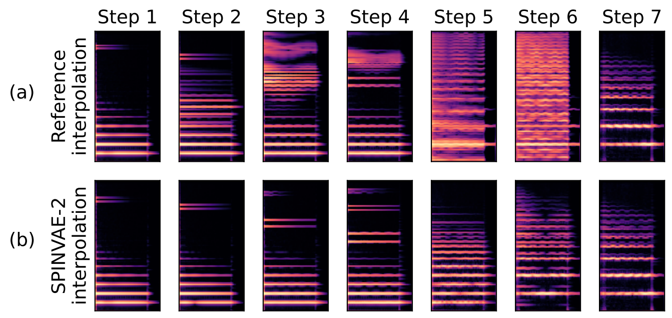

Fig. 1. Spectrograms of two preset interpolations based on the same starting (first column) and ending (last column) presets. (a) Linear interpolation in the synthesis parameters' domain. (b) Interpolation computed using the model proposed in this work, which sounds as a much better morphing than (a).

The main contribution of this work is a Variational Auto-Encoder (VAE) [11] model, dedicated to preset interpolation for black-box, non-differentiable synthesizers. It builds upon our previous work [12], which was the first to handle presets as sequences of parameters using multi-head attention [13] networks. Section III introduces a new structure, appropriate mixture distributions for preset modeling, and an extra regularization term [14], [15] based on timbre attributes [16], [17]. The model has been trained on a dataset of Frequency Modulation (FM) synthesis presets, and has been used for experiments presented in this manuscript.

An objective morphing evaluation method, based on metrics computed from audio features, is introduced in Section IV. It is an explainable and reproducible approach, used in Section V to compare the model to traditional waveform morphing techniques, and to linear preset interpolation for a black-box synthesizer. Section VI provides a more extensive analysis of the model. Finally, Section VII presents subjective evaluations of interpolations. The results demonstrate that our model outperforms the linear interpolation of presets. Moreover, they show that the proposed objective metrics are well correlated to subjective measurements, and allow to identify the more relevant objective features in the context of sound morphing.

II. Related Works

A. Neural Audio Synthesis

Convolutional Neural Networks (CNNs) can be trained to generate raw audio waveforms from time series of latent codes, e.g. WaveNet [18] or RAVE [19]. Recent research has sought alternatives to per-sample computations by developing synthesis processes resembling the most common sound synthesizers. Some of these approaches are based on differentiable source-filter models [1], [20], [21], while others [22] emulate a differentiable but simplified FM synthesis architecture.

However, these neural networks encompass the synthesis process itself and are trained through gradient descent. Consequently, they are not suitable for application in existing commercial synthesizers [1], [7]–[9], which rely on non-differentiable methods. Developing differentiable digital signal processors entails high engineering costs and potentially limits practical uses [23]. Moreover, reconstructions of audio inputs through differentiable synthesizers present a significantly degraded audio quality [24].

B. Sound Matching

The task of sound matching, also called automatic synthesizer programming, consists in searching for synthesis parameters that correspond best to an input sound. Recent works have employed neural networks to tackle this task.

The input of these models can be Mel-Frequency Cepstral Coefficients (MFCCs) [4], audio spectrograms [1], [6], [7], raw waveforms [5] or sets of audio features [9]. The outputs are synthesizer presets, and each model is designed to handle a specific synthesis method. Some works focused on additive-subtractive synthesis [1], [5], [6] whereas others considered FM synthesis [4], [7], [9].

C. Generative Models

Among the cited sound matching models, a few [6], [7] are based on generative models with explicit probability distributions. They encode input audio data $\mathbf{x}$ into a latent distribution $q(\mathbf{z}|\mathbf{x})$, sample a latent vector $\mathbf{z}$ from $q(\mathbf{z}|\mathbf{x})$, then try to reconstruct the audio and infer the preset from $\mathbf{z}$. After training, latent vectors can be sampled from the prior distribution $p(\mathbf{z})$ or a posterior distribution $q(\mathbf{z}|\mathbf{x})$ in order to generate new audio samples and new presets.

The VAE [11] is a usual framework to obtain latent representations and a generative model simultaneously. It learns an approximate posterior $q(\mathbf{z}|\mathbf{x})$, which represents how $\mathbf{x}$ is encoded into the latent space, and a decoder model $p(\mathbf{x},\mathbf{z}) = p(\mathbf{x}|\mathbf{z})\,p(\mathbf{z})$. Typical distributions choices are $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, I)$ and $q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu, \sigma^2)$, where μ and σ^2 are the outputs of an encoder neural network. $p(\mathbf{x}|\mathbf{z})$ can be any distribution whose parameters are the outputs of a decoder neural network. The loss $\mathcal{L}(\mathbf{x})$ is an upper bound on the negative log-likelihood of the true data distribution:

$$\mathcal{L}(\mathbf{x}) = \beta D_{KL}\left[q(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})\right] - \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right] \quad (1)$$

where D_{KL} denotes the Kullback-Leibler divergence and greater β values improve latent regularization but degrade reconstruction accuracy [25]. The expectation is typically approximated using a single Monte Carlo sample $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})$.

In Equation (1), the first term acts as a regularization loss which forces $q(\mathbf{z}|\mathbf{x})$ to remain close to $\mathcal{N}(\mathbf{z}; 0, I)$. This prevents $\mathbf{x}$ inputs from being encoded as distributions with disjoint supports, i.e. $q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu, \sigma^2)$ with $\sigma \to 0$. Thus, in contrast to deterministic auto-encoders, the latent space should be continuous [11] i.e. close inputs should correspond to similar encoded distributions.

D. Sound Morphing

Sound morphing consists in the hybridization of two input audio waveforms. The most common method is based on sinusoidal modeling of the input signals, followed by interpolation of the time-varying parameters (frequencies and amplitudes of the partials) and additive resynthesis [26]–[30]. The non-harmonic parts of input signals, e.g., transients or noisy sounds which cannot be easily reproduced by pure additive synthesis, can be handled separately. For instance, [29] uses linear predictive coding [31] to model a noise source and filter. Sinusoidal analysis and resynthesis nonetheless require to match the partials of the two input sounds, which poses a difficult problem [32]. Some methods also require to manually align some temporal events of the two input signals [27].

Morphing can also rely on source-filter models, whose time-varying parameters are obtained from instantaneous fundamental frequency and spectral envelope. For instance, STRAIGHT representations [33] have been designed for speech but can be used to perform any sound morphing by bilinear interpolation on the time and frequency axes. Other works [10], [32], [34] transform sinusoidal representations into partial locations and spectral envelopes, whereas non-harmonic residuals are modeled as white noise filtered through spectral envelopes.

Although sinusoidal plus noise and source-filter models can theoretically reproduce arbitrary input sounds, a difference can be perceived between original and reconstructed signals [34], even for quasi-harmonic orchestral instruments. In particular, filtered white noise is perceptually different from the true residuals of a sinusoidal decomposition [34]. Therefore, some studies focus on noisy, non-harmonic transients. For instance, [35] employs discrete wavelet transforms. Regarding more specific applications in video games, impact sounds [36] or granular textures [37] can be interpolated using inverse cumulative amplitude spectra. In order to morph between two similar percussive sounds, e.g. two snare or two hi-hat samples, [38] suggests to temporally align the input samples using time stretching, then to perform a simple cross-fade.

Other spectral techniques can morph sounds without assuming a signal model. For instance, an interpolation can be performed in a perceptual spectral domain obtained from MFCC and inverse MFCC computations [39]. [40] estimates a Gabor mask which enables morphing between two spectrograms. Non-negative matrix factorization (NMF) can be used to decompose the two inputs into spectral bases and activations, and the morphing is computed through an interpolation of the spectral bases [41]. The morphing can be processed separately on the quasi-harmonic and transients parts obtained from Harmonic-Percussive Source Separation (HPSS) [42]. Most of these techniques require spectrogram inversion and phase reconstruction in order to produce an output waveform. Non-data-driven techniques for phase reconstruction include the Griffin-Lim algorithm [43] and phase gradient heap integration [44].

The most recent studies show that morphing algorithms can integrate data-driven, differentiable models. For instance, [45] generates Mel-spectrograms conditioned on instrument timbre, and uses a WaveNet [18] vocoder to reconstruct audio waveforms. Latent Timbre Synthesis (LTS) [46] suggests to use a VAE with Constant-Q Transform (CQT) [47] inputs and outputs. [48] integrates a differentiable Harmonic plus Noise (HpN) synthesis model [20]. Generative Adversarial Networks (GANs) [49] can be used for the specific application of audio texture morphing [50].

E. Preset Interpolation

The main limitation of all classic sound morphing techniques is that each one is restricted to a specific synthesis method. It prevents their adoption by users who choose specific synthesizers for their characteristic audio qualities. Moreover, dedicated sound models often yield suboptimal results with different acoustic sources [30].

Sound morphing algorithms also typically rely on computationally intensive analysis and resynthesis, which is either acknowledged in the original publications [36], [37] or can be observed in practice (sub-section V-A). Moreover, several sound morphing methods cannot be easily employed to play a morphed sound using any MIDI note [35]–[38], [40], [41], although this is arguably the most basic functionality of a given synthesizer. A notable exception is the Kyma [51] implementation of Loris [27] additive synthesis, which circumvents these two issues. It nonetheless requires tedious manual annotations of input audio samples, and has to run on dedicated proprietary signal processing hardware.

Synthesis techniques cited in sub-section II-D have been selected for morphing because they are known to enable smooth transitions. For instance, [33] performs resynthesis from smooth time-frequency representations, and [10], [34] compute interpolations of smoothed spectral envelopes. Morphing through additive resynthesis [30] using matched partials is also inherently smooth. In contrast, widespread synthesizers typically provide some non-differentiable and non-linear synthesis parameters. Examples include categorical waveform selectors, discrete routing of low- or audio-frequency signals, non-linear distortion parameters, etc.

In this work, the goal is to build a model that enables interpolation of presets for any synthesizer (Figure 2). In

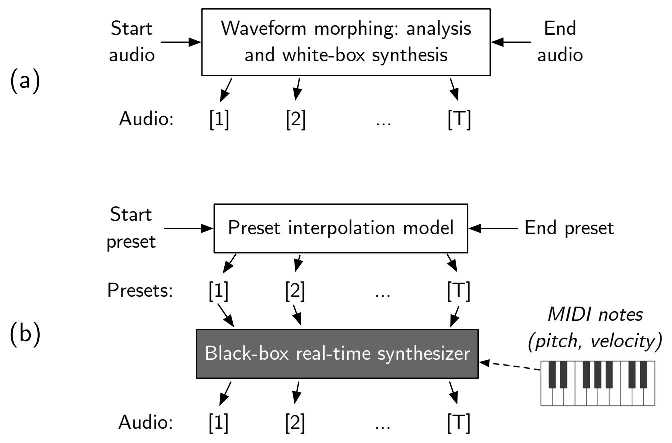
(a)
Start audio
Waveform morphing: analysis and white-box synthesis
End audio
Audio: [1] [2] ... [T]
(b)
Start preset
Preset interpolation model
End preset
Presets: [1] [2] ... [T]
Black-box real-time synthesizer
MIDI notes (pitch, velocity)
Audio: [1] [2] ... [T]

Fig. 2. Block diagrams of classic sound morphing techniques (a) and preset interpolation for a black-box synthesizer (b).

order to provide a general model, synthesizers are treated as non-differentiable black boxes whose presets are made of numerical and categorical parameters. The current *de facto* method consists of computing a linear interpolation for each synthesis parameter, where a *macro-control* guides the interpolation from the starting to the ending preset. This approach can result in smooth transitions, as synthesis controls are typically aligned with perceptual scales, e.g., logarithmic for frequencies and amplitudes. It can be well suited to additive and subtractive synthesizers made of independent oscillators and filters, if the latter have a similar role in the two presets. However, in practice, a perfect correspondence is unlikely to occur [29]. An interpolation in the parameters' space is not guaranteed to be a good morphing in the perceptual space [52].

Thus, an improved non-linear method, applicable to any black-box synthesizer, would be useful. Modern models such as neural networks could fulfill this task. VAEs [25], [53] or adversarial auto-encoders [54] can be trained to improve the interpolation between reconstructed audio outputs. However, these works rely on learned differentiable decoders, while ours focuses on interpolating presets for an external sound generator. Some previous sound matching studies [6], [7] use VAEs which could theoretically perform morphings between presets. They nonetheless don't study the interpolation itself, and use neural networks that can't handle numerical and categorical synthesis parameters simultaneously (see details in sub-section III-C).

A model dedicated to synthesizer preset morphing has been recently published [12]. Based on a VAE architecture and Transformer [13] blocks, it significantly outperforms the reference method and generative sound matching architectures. This paper builds upon our previous work [12], by introducing an enhanced model and training procedure. Additionally, it offers a comprehensive analysis of the model and the interpolation quality evaluation, addressing aspects that were not explored in our previous publication.

F. FM Synthesis

FM synthesis [55] can generate complex audio spectra with a reduced set of parameters. Given two oscillators, named *operators*, and ignoring initial phases, the carrier operator's output signal is:

$$x_{out}(t) = \sin(2\pi f_c t + M \sin(2\pi f_m t))$$

where f_c and f_m are the carrier and modulator frequencies, respectively, and M is the modulation index. The first widespread FM synthesizer was the Yamaha DX7, released in 1983. It is known to be able to synthesize a wide variety of instruments such as digital or acoustic pianos and drums, brass, wind and string instruments, as well as digital sound textures [55]. It is a hardware synthesizer but has many software counterparts. Dexed[1] is an open-source software clone that can play original DX7 presets, while Arturia DX7 V provides an updated version of the original DX7. Other renowned modern FM synthesizers include Native Instruments FM8 and Ableton Live Operator.

Considering the large amount of synthesis parameters and the intricate relationships between them, all FM synthesizers are notoriously hard to program. For instance, the DX7 comprises 155 parameters such as pitch and amplitude envelope knobs, frequency knobs, frequency scale switches, envelope curve switches, etc. Many parameters entail non-differentiable relationships between output sounds and input presets. In particular, an *algorithm* parameter controls the FM architecture. Fig. 3 shows that different DX7 algorithms correspond to various depths of modulation. E.g., algorithm 2 has four levels of modulation and is likely to create very rich harmonic content, whereas algorithm 32 transforms the DX7 into a primarily additive synthesizer with optional feedback for operator 6. Fig. 3 also illustrates that the role of operators (e.g. operator 3) changes from an algorithm to another, which further explains why such synthesis is inherently non-differentiable. Among previously cited works, some [9], [22] had to exclude the FM algorithm from their models, and trained one simplified model for each algorithm.

The DX7 is an interesting and challenging synthesizer, thus it has been chosen as the primary focus of this work. The model introduced in this manuscript handles the FM algorithm parameter. A model that performs well on the DX7 can be expected to have broad applicability across other real-world synthesizers.

A linear interpolation can be used to smoothly morph between some specific FM sounds, for instance from a drum to a trumpet [56]. Results presented in sub-section VII-D confirm that the linear interpolation can perform well. However, it can also fail because of the intricate relationships between synthesis parameters. An example is given in Fig. 1(a) where steps 5 and 6 are completely irrelevant. Among FM synthesizers, only Native Instruments' FM8 claims that it can perform "morphing between the timbral characteristics of four FM8 sounds" [57]. Nevertheless, the morphing is restricted to "new FM8 sounds" and excludes some synthesis parameters such as envelopes and modulations. The interpolation method or quantified morphing quality measurements are also not reported.

[1]https://asb2m10.github.io/dexed/

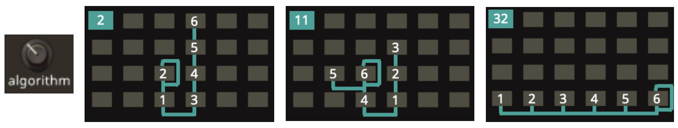

Fig. 3. Three DX7 algorithms out of 32 total. Upper oscillators (called *operators*) modulate lower ones.

III. Proposed Model

A. Synthesizer and Dataset

1) Presets and synthesized audio: we used a database of approximately 30k presets [7] of the DX7 FM synthesizer. It was divided into 80% for training, 10% for validation, and 10% for testing. Non-FM synthesis controls like volume, transpose, and filter were set to default and not altered. A preset comprises the 144 remaining parameters, including the FM algorithm which determines the signal flow among oscillators (Fig. 3) and is arguably the most important FM synthesis parameter. Unlike recent studies [9], [22] that used separate models for each algorithm, our approach involves a single model that incorporates the algorithm parameter. Employing a single model is essential for enabling interpolation between arbitrary FM presets using different algorithms.

All presets are rendered by Dexed to 16kHz audio using the MIDI note 56 (G#3) with velocity 75, played during three seconds and recorded for four seconds. They are also converted to 257-band mel-spectrograms used during training only.

2) Audio features: audio renders are used to compute features that represent different perceptual aspects of the corresponding presets. Audio Commons Timbral Models (ACTM) [17] and Timbre Toolbox (TT) [16] allow the extraction of 8 and 46 features, respectively. TT computes a set of spectral and temporal characteristics, e.g. *inharmonicity*, *spectro-temporal variation* and *Log Attack Time* (LAT). These audio features act as a proxy for the human perception of timbre [52]. As a complement to TT features, ACTM provides semantic features such as *boominess, warmth* or *depth*. ACTM descriptors' values are computed by regression models trained on audio samples with semantic labels. Regressions' input variables are spectral and temporal features.

All features are zero-mean, unit-variance normalized using statistics of the training set. An unsupervised feature selection has been performed on the combined set of ACTM and TT features, and any feature that was highly correlated (> 0.9) to another across the training dataset has been excluded. This reduces the number of ACTM and TT descriptors to 6 and 32, respectively. The set of attributes corresponding to a preset $\mathbf{x}$ is denoted $\mathbf{a} \in \mathbb{R}^{38}$.

B. General Architecture

The main model introduced in this work, named SPINVAE-2 (Synthesizer Preset Interpolation VAE), is structured as a preset VAE with an extra mel-spectrogram decoder (Fig. 4). It is inspired by, but substantially different from the original SPINVAE model [12]. Thanks to latent space properties, VAE-encoded vectors can be interpolated and the intermediate vectors can be decoded into presets. The training loss of the whole model is the following:

$$\begin{aligned}\mathcal{L}(\mathbf{x},\mathbf{u}) &= \overbrace{\mathcal{L}_{\text{preset}} + \mathcal{L}_{\text{audio}}}^{\text{Reconstruction}} + \overbrace{\beta\mathcal{L}_{\text{DKL}} + \gamma\mathcal{L}_{\text{timbre}}}^{\text{Regularization}} \\ &= -\log p(\mathbf{u}|\mathbf{z}) - \log p(\mathbf{x}|\mathbf{z}) \\ &\quad + \beta D_{KL}\left(q(\mathbf{z}|\mathbf{u})||p(\mathbf{z})\right) + \gamma\mathcal{L}_{\text{timbre}} \qquad (2)\end{aligned}$$

A vanilla Preset-VAE, composed solely of the Transformer encoder and decoder, would minimize $\mathcal{L}_{\text{preset}}$ and $\beta\mathcal{L}_{\text{DKL}}$ only. Additionally, $\mathcal{L}_{\text{audio}}$ makes the model learn how to generate mel-spectrograms $\hat{\mathbf{x}}$ from the latent representation $\mathbf{z}$ of a preset. However, in contrast to the first SPINVAE model [12], the ground truth audio $\mathbf{x}$ is not provided to the model. Thus, the audio decoder CNN—associated to the preset encoder—can be considered as a proxy for the synthesizer. They approximate a non-differentiable process with a simpler differentiable model. Neural proxies had previously been applied to the emulation of audio effects [23], [58] or their gradients [59]. The audio decoder $p(\mathbf{x}|\mathbf{z})$ is a nine-layer CNN with residual connections and models each spectrogram pixel as a unit-variance Gaussian distribution. It is only used during training.

The last term, $\gamma\mathcal{L}_{\text{timbre}}$ with $\gamma \in \mathbb{R}^+$, ensures that latent codes hold meaningful perceptual information. It will be discussed in sub-section III-D.

C. Presets Modeling

1) Transformer-based architecture: presets encoder and decoder are very similar to those of SPINVAE-1 [12], which was the first work to successfully model presets using Transformer [13] blocks. A preset is processed as an ordered array of synthesis parameters whose values are highly interdependent. We do not use attention masks, i.e., each hidden token can attend to tokens at any position. This setup allows tokens to fully consider the relationships between different synthesis parameters. For instance, the values of tokens corresponding to an operator can be processed differently depending on the values of tokens corresponding to the FM algorithm—which changes the routing of operators' output signals—and any other operator. The encoder and decoder each consist of six layers with four attention heads, and hidden tokens' size has been empirically set to 256.

Long short-term memory [60] and multi-layer perceptron networks can be used for sound matching tasks [4], [7]. Their performance is nonetheless much worse than that of Transformers in the context of preset interpolation [12]. Generative models for sound matching [6], [7] use normalizing flows [61], [62] to model continuous synthesis parameters such as *frequency* or *amplitude*. However, numerical instabilities were observed during the training of these auto-regressive neural networks, and they are not adapted to categorical parameters such as *algorithm* or *waveform type*. Categorical flows [63] would be applicable to these parameters, but can't handle numerical and categorical outputs simultaneously.

Therefore, Transformers have been chosen to process presets and their hidden representations. Transformer-based VAEs

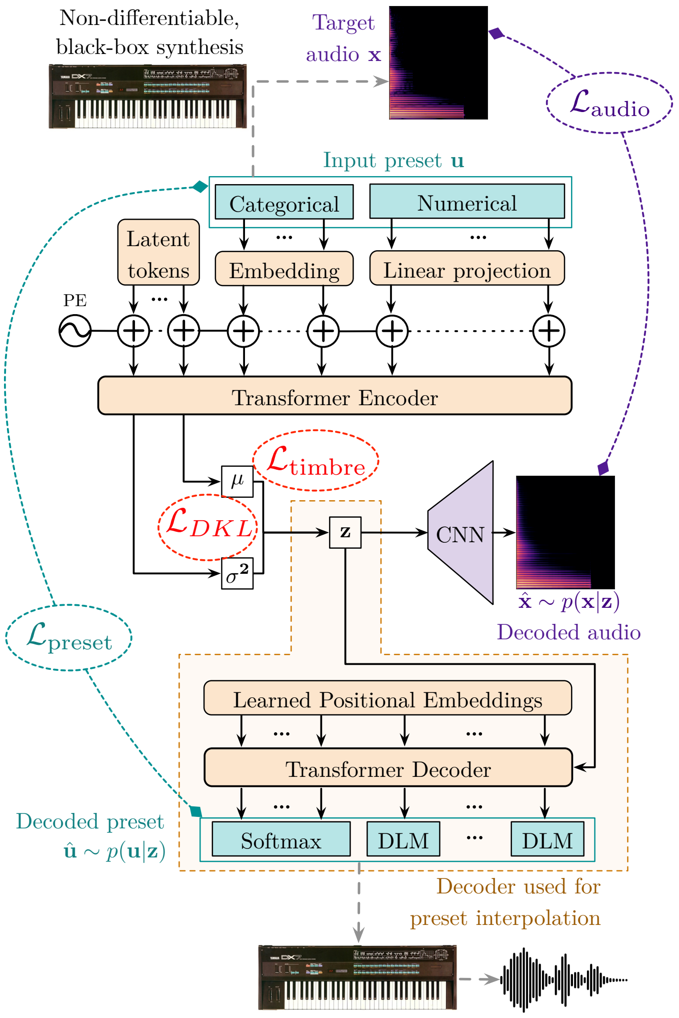
Non-differentiable, black-box synthesis
Target audio $\mathbf{x}$
$\mathcal{L}_{\text{audio}}$
Input preset $\mathbf{u}$
Categorical
Numerical
Latent tokens
Embedding
Linear projection
PE
Transformer Encoder
μ
$\mathcal{L}_{\text{timbre}}$
$\mathcal{L}_{DKL}$
$\mathbf{z}$
CNN
σ^2
$\hat{\mathbf{x}} \sim p(\mathbf{x}|\mathbf{z})$
Decoded audio
$\mathcal{L}_{\text{preset}}$
Learned Positional Embeddings
Transformer Decoder
Decoded preset $\hat{\mathbf{u}} \sim p(\mathbf{u}|\mathbf{z})$
Softmax
DLM
DLM
Decoder used for preset interpolation

Fig. 4. Architecture of the SPINVAE-2 model and training losses $\mathcal{L}$.

have been applied to the somewhat related fields of symbolic music [64] or 3D motion [65] generation.

2) Encoder: preset embeddings are computed using learned dictionaries for categorical synthesis parameters and a linear projection for numerical parameters. They are concatenated with latent tokens (Fig. 4) which are learned during training but remain constant during inference. Then, sinusoidal Positional Encodings (PE, [13]) are added to all tokens and the result is provided to the encoder. Only the first output tokens are kept and reshaped to build the μ and σ vectors of $q(\mathbf{z}|\mathbf{u})$ in Eq. 2. The number of latent tokens depends on the dimension L_z of μ, σ and $\mathbf{z}$. For instance, if $L_z = 512$, then four 256-dimensional extra tokens are provided to the encoder's input.

3) Decoder: after sampling, $\mathbf{z}$ is reshaped into a sequence of memory tokens to compute keys and values, while some learned positional embeddings (one token per synthesis parameter) are used to compute queries.

In contrast to previous works [4]–[7], the preset decoder $p(\mathbf{u}|\mathbf{z})$ employs different distributions for categorical and numerical synthesis parameters. Categorical outputs are obtained by applying a softmax function on each output token. Numerical parameters are nonetheless discrete, e.g., a DX7 *amplitude* control has 100 discrete values whereas a *detune* control provides a discrete range of 15 semitones.

We model the latter by Discretized Logistic Mixture (DLM) [66] distributions, which were originally applied to pixel values in [0..255]. They compute probabilities using discrete bins, and tend to assign more probability to the lowest and highest bins. We use three mixture components [12]. For a given value $u_i \in (0..N_i - 1)$ of a synthesis parameter, the probability mass function is:

$$p\left(u_i|\mathbf{z};\pi_i,\mathbf{m}_i,\mathbf{s}_i\right) = \sum_{k=1}^{3} \pi_{i,k}\big(S\left((u_i + 0.5 - m_{i,k})/s_{i,k}\right) - S\left((u_i - 0.5 - m_{i,k})/s_{i,k}\right)\big) \quad (3)$$

where $\pi_i, \mathbf{m}_i, \mathbf{s}_i \in \mathbb{R}^3$ are extracted from the i^{th} output token and S denotes the logistic sigmoid function. For edge cases $u_i = 0$ or $u_i = N_i - 1$, replace $u_i - 0.5$ by $-\infty$ or $u_i + 0.5$ by $+\infty$, respectively. In contrast to the original DLM implementation [66], ours allows a different N_i for each u_i. Log-probabilities computations remain, nevertheless, parallel and numerically stable.

D. Attribute-Based Regularization: $\mathcal{L}_{timbre}$

The $\mathcal{L}_{\text{DKL}}$ loss from the basic Preset-VAE does not guarantee that $q(\mathbf{z}|\mathbf{u})$ encodes timbre characteristics; latent coefficients have been shown not to easily relate to perception [2], [67]. Yet, we indeed want an interpolation that affects timbre smoothly and continuously. Therefore, an additional loss term, denoted $\mathcal{L}_{\text{timbre}}$ (Equation 2), is introduced in the training of SPINVAE-2. $\mathcal{L}_{\text{timbre}}$ incites the VAE to match ACTM and TT features, noted $\mathbf{a}$, to some dimensions of the latent space. This extra loss shifts the learning from purely unsupervised to self-supervised, where attributes are automatically computed rather than extracted from human annotation. Nonetheless, this cannot be interpreted as fully-supervised learning, as we do not train models directly for interpolation.

Two regularization methods, inspired by works on latent disentanglement, have been implemented and tested. The first [14], [68] minimizes the binary cross-entropy between $S(\mathbf{a})$ and $S(\mu)$, where S denotes the logistic sigmoid. The second, called Attributed-based Regularization (AR, [14], [15]), enforces monotonic relationships between timbre attributes and some latent dimensions. Given a minibatch of M presets, the AR loss for the j^{th} latent dimensions relies on two distance matrices. $D_{\mathbf{a},j} \in \mathbb{R}^{M\times M}$ holds the differences between values of the j^{th} feature, $\mathbf{a}_j$, for all examples from the minibatch:

$$D_{\mathbf{a},j}\left(n,m\right) = \mathbf{a}_j^{(n)} - \mathbf{a}_j^{(m)} \quad (4)$$

with $j \in [0, 37]$ and where $n, m \in [0, M)$ designate indices of items in a minibatch. Similarly, $D_{\mu,j} \in \mathbb{R}^{M\times M}$ is the matrix of differences between encoded means μ_j:

$$D_{\mu,j}\left(n,m\right) = \mu_j^{(n)} - \mu_j^{(m)} \quad (5)$$

Considering the j^{th} latent dimension only, the loss becomes:

$$\mathcal{L}_{\text{AR}}\left(\mu_j, \mathbf{a}_j\right) = \text{MAE}\big(\tanh\left(\delta D_{\mu,j}\right) - \text{sign}\left(D_{\mathbf{a},j}\right)\big) \quad (6)$$

where the δ hyperparameter influences the spread of $q(\mathbf{z}|\mathbf{u};\mu,\sigma^2)$ distributions. Finally, $\mathcal{L}_{\text{timbre}}$ is the average of $\mathcal{L}_{\text{AR}}$ terms computed for all regularized dimensions, which might be fewer than the total number of latent dimensions. If there are more than 152 latent dimensions, each timbre feature is used to regularize four dimensions rather than one. This corresponds to the 2×2 size of feature maps provided to the CNN decoder input. The total amount of regularized latent dimensions becomes $38 \times 4 = 152$.

In contrast to other regularization methods [2], [67], [68], AR does not enforce latent values during training, but rather imposes that an encoder output μ_j varies in the same direction as the corresponding $\mathbf{a}_j$. The AR method appears to yield marginally improved performance, though these enhancements are not substantial. It has been arbitrarily chosen as the regularization method in all experiments presented in our work. Similar to β, a hyper-parameter γ controls the amount of regularization.

E. Training Procedure

In order to reduce fine-tuning times, a general bi-modal (mel-spectrograms and presets) VAE has been pre-trained. A CNN encoder is added to the model, and its outputs are reshaped and summed to the μ and σ^2 vectors from Fig. 4. This architecture is very similar to SPINVAE-1 [12]. First, only the CNN encoder and decoder are trained, from a merged dataset of mel-spectrograms from NSynth notes [18] (714 different instruments), 2.2k Surge synthesizer[2] patches and 24k Dexed presets. Second, the bi-modal model is trained using the Dexed presets only. Even though it present very poor performance, it is able to approximately reconstruct presets and spectrograms. The weights of embedding, encoder and decoder layers are used as initial weights for all models presented in the results.

Fine-tuning is then performed without the CNN encoder, which corresponds to the architecture from Fig. 4. It is quite fast (approximately 2.5 hours on an Nvidia RTX 3090 GPU), which allows us to conduct large hyper-parameter sweeps. Training and implementation details are available online[3].

IV. Objective Interpolation Evaluation

A. Interpolation Computation

Following training and validation, 1.5k interpolations were generated using consecutive pairs of presets from the shuffled test set (comprising 3k samples). This is much more than recent sound morphing works, e.g. [10] whose evaluation was based on 26 pairs of sounds. Most studies [26]–[29], [33], [35]–[39], [41], [48], [50], [56] present analyses of a few (one to five) test sounds. In contrast, our test set is extensive and contains all types of sounds described in sub-section II-F.

Each interpolation is computed as follows. Initially, two samples, $\mathbf{u}^{(n)}$ and $\mathbf{u}^{(m)}$, are encoded into $\mathbf{z}^{(n)} = \mu^{(n)}$ and

[2]https://surge-synthesizer.github.io
[3]https://github.com/gwendal-lv/spinvae2

$\mathbf{z}^{(m)} = \mu^{(m)}$, respectively. Subsequent linear interpolation in the latent space produces a series of $\{\mathbf{z}[t], t \in [1, T]\}$ vectors, where $\mathbf{z}[1] = \mathbf{z}^{(n)}$ and $\mathbf{z}[T] = \mathbf{z}^{(m)}$. Each vector $\mathbf{z}[t]$ is decoded into a preset $\mathbf{u}[t]$, which is finally rendered to audio by Dexed (Fig. 4). An interpolation is made of $T = 9$ steps.

B. Morphing Metrics

Assessing interpolation quality is straightforward for basic artificial shapes like 2D lines [54]. Objective and unambiguous features, such as measured length and orientation, can be used. In contrast, defining what constitutes a ”good” audio interpolation is more complex. Perceptual evaluations are tedious [10] and cannot reasonably be performed on large sets of models. As a result, most works [26], [27], [29], [35]–[37], [39], [41], [48], [56] related to sound matching provide only a qualitative analysis based on a few selected examples.

Hence, an automatic, explainable and reproducible method is desirable. A general framework [69] suggests to use criteria such as correspondence, intermediateness and smoothness. The same authors [10] eventually evaluate the quality of a morphing by computing the non-linearity of four spectral and two temporal features. The first SPINVAE paper [12] has introduced the computation of both linearity and smoothness of numerous timbre features, but did not explain them in details. This section fully describes the objective morphing evaluation method, while its correlations with a subjective evaluation are provided in section VII.

Recent works [54], [65], [67] on model-based interpolation focused on the smoothness criteria. Given an objective feature, e.g. the surface of a 2D object, smoothness can be defined using the RMS of the second-order derivative of this feature's values across the interpolation sequence [54]. Here, the smoothness of the j^{th} timbre feature $\mathbf{a}_j$ is computed as:

$$-\sqrt{\frac{1}{T-2}\sum_{t=2}^{T-1}\left(\frac{\mathbf{a}_j[t-1] - 2\mathbf{a}_j[t] + \mathbf{a}_j[t+1]}{(T-1)^2}\right)^2} \qquad (7)$$

The minus sign ensures that the highest smoothness values (close to zero) correspond to the best smoothness, whereas decreasing negative values indicate a degraded smoothness. Our previous work [12] introduced the smoothness of an audio morphing as the average smoothness of timbre features. Smoothness alone can nonetheless occasionally be misleading. For instance, a degenerated interpolation made of zero-volume sounds would present a medium smoothness score. More generally, smoothness does not quantify whether feature values deviate from an ideal linear trajectory in feature space. This is illustrated by the *low linearity* curve from Fig. 5, which is moderately smooth.

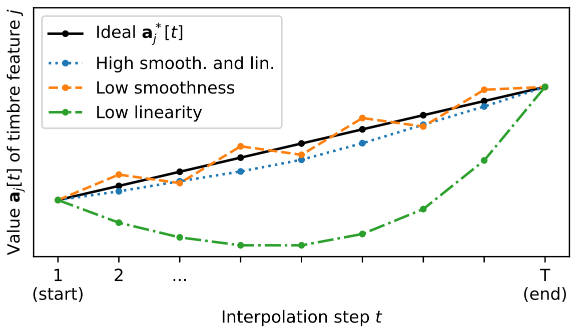

Fig. 5. Qualitative examples of interpolation linearity and smoothness for different trajectories of the same timbre feature.

Therefore, the linearity of morphings is also computed. This aligns with prior research on sound morphing [10], which considers that an interpolation is perceptually linear when timbre feature values change linearly. In our work, the feature-wise non-linearity is the RMS distance between feature values $\mathbf{a}_j[t]$, and a linear regression from starting to ending values of the feature (the *ideal* curve from Fig. 5) noted $\mathbf{a}_j^*[t]$. Formally, the linearity is computed as:

$$-\sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(\mathbf{a}_j[t] - \mathbf{a}_j^*[t]\right)^2} \qquad (8)$$

Similar to smoothness, it is negative and a low value corresponds to a low-quality morphing. The highest and best possible value is zero.

Nonetheless, the linearity metric complements but cannot replace the smoothness metric. Good smoothness ensures that the morphing does not contain small but steep feature variations. E.g., the *low smoothness* curve from Fig. 5 has the exact same linearity as the *good* curve, but its smoothness is approximately 20 times worse. Results presented in this work use both metrics: linearity and smoothness of interpolations. The method is provided as an open-source toolbox[4] and can be applied to any sound morphing.

C. Timbre Features

The two interpolation metrics are computed on audio features obtained with TT and ACTM, as described in sub-section III-A. TT's *attack time* feature has been excluded from the evaluation, in order to focus on LAT which is one of the most perceptually salient dimensions of timbre [10], [70]. The final set of 37 features used is displayed on Fig. 9 at the end of this manuscript. Although it may seem large, this is not unusual for music information retrieval tasks [71]. Moreover, sub-section VII-E demonstrates that the additional semantic ACTM features, not part of the previous evaluation [12], are among the most related to the perception of interpolation quality.

V. Comparison with Sound Morphing Algorithms

A. Waveform morphing techniques

Multiple techniques for morphing between two sounds have been introduced in sub-section II-D. Although they cannot be used with black-box synthesizers and have limited applicability in practice (Section II), they would provide a baseline to compare the SPINVAE model to. Unfortunately, the literature does not indicate what the current state of the art is. Hence, this

[4]https://github.com/gwendal-lv/sound-morphing-metrics

TABLE I
TEST-SET PERFORMANCE OF PRESET INTERPOLATION (LINEAR AND SPINVAE) AND WAVEFORM MORPHING TECHNIQUES, USING SMT AS A BASELINE. (↑) INDICATES THAT HIGHER VALUES ARE BETTER AND (↓) INDICATES THAT LOWER VALUES ARE BETTER.

Method	**Morphing improvement vs. SMT baseline**		**Reconstruction error**		**Computation times**		**Black-box, real-time playable synthesizer**
	Linearity (↑)	Smoothness (↑)	MFCCD (↓)	PEMO-Q ODG (↑)	Analysis (↓)	Interpolation and synthesis (↓)	
Spectral NMF [41] w/o HPSS	−27.5 %	−52.2 %	18.5	−2.42	1.23 s	2.03 s	No
STRAIGHT source-filter [33]	+7.9 %	+2.2 %	15.3	−2.02	13.3 s	1.11 s	No
SMT additive [30] (baseline)	0 %	0 %	10.2	−1.21	155.5 s	0.194 s	No
LTS (CQT-VAE) [46]	−1.2 %	−36.4 %	15.8	−2.60	0.403 s	1.78 s	No
Differentiable HpN [20]	+2.7 %	+5.6 %	10.0	−2.84	3.92 s	0.072 s	No
— with timbre AR	+4.1 %	+10.7 %	10.2	−2.86	3.92 s	0.072 s	No
Linear preset interpolation	−34.0 %	−285 %	0	0	0	0.012 s	Yes
SPINVAE-1	−13.1 %	−217 %	7.1	−1.66	0.018 s	0.063 s	Yes
SPINVAE-2	−2.2 %	−213 %	4.1	−1.19	0.018 s	0.063 s	Yes

sub-section presents objective morphing performance of various works. Non-automatic methods requiring manual labeling of test sounds, closed-source implementations, and algorithms not able to run on a CPU have been excluded.

Regarding classical, non-data-driven morphing, one spectral technique and two techniques based on different signal models have been selected and are listed in the first section of Tab. I. The Sound Morphing Toolbox (SMT) [30] implements a well-known morphing procedure and has been chosen as the baseline. Its *spectral oversampling* parameter was reduced from 4 to 2, otherwise the analysis needed more than the 32 GB of RAM available on the testing machine.

The second section of Tab. I presents three techniques based on neural networks. LTS [46] is a spectral autoencoder, and morphing is performed by linear interpolation in the latent space. The two others are based on the same differentiable HpN synthesizer [20]. The latter embeds time slices of input audio into latent vectors, which are subsequently decoded into synthesis parameters. The dimension of latent vectors has been increased from 16 to 32 in order to improve the reconstruction accuracy. Normalized pitches and loudness levels are computed in parallel and also provided to the decoder. Direct linear interpolation of these time-varying values enables the best morphing performances. A variant of the model has also been trained with timbre AR applied on each latent vector, using the 26 available spectral timbre features (purely temporal features excluded).

Values of the objective linearity and smoothness metrics are highly dependent on the timbre feature, making absolute values difficult to visualize and interpret. Thus, all linearity and smoothness results from Tab. I are presented as variations relative to the SMT baseline. A positive (resp. negative) variation indicates an improvement (resp. degradation) in interpolation quality. The best possible variation would be $+100\%$ and would correspond to linearity and smoothness metrics having an optimal zero value. A -100% variation corresponds to metric values that are twice those of the SMT baseline.

Tab. I also provides the 13-band MFCC Distance (MFCCD) between ground truth sounds from the test dataset and their reconstruction by each model, considered as a perceptual distance in timbre space [72]. Studies about sound matching [4], [9] considered that the MFCCD threshold for perceived similarity is between 10 and 15. PEMO-Q [73] evaluations of the perceived quality difference between reconstructions and original sounds have been computed. In particular, the Objective Difference Grade (ODG) quantifies a subjective quality impairment from 0 (*imperceptible*) to -4 (*very annoying*).

Identifying the best technique among these six is not straightforward. Spectral techniques (NMF and LTS) clearly underperform compared to the others. They struggle to reconstruct test sounds while morphings are less smooth and linear than the others. STRAIGHT enables the most linear morphings, but the reconstruction error is one of the largest.

Morphing based on differentiable HpN synthesis is smoother and more linear than morphing with SMT. Interestingly, the timbre AR introduced in sub-section III-D further improves the morphing. The perceptual MFCCD timbre distance is very similar for both. However, the average PEMO-Q ODG score is much better with SMT (close to -1, i.e. *perceptible but not annoying*) than differentiable HpN (close to -3, i.e. *annoying*). This is probably due to the frequent audible artifacts generated by differentiable synthesizers. Therefore, SMT can be considered as the best compromise between morphing performance and reconstructed audio quality. Informal listening tests nonetheless show that intermediate samples lack some timbral characteristics of the start and end sounds.

B. *Preset interpolation with SPINVAE*

The last section of Tab. I reports performances of the best SPINVAE-1 [12] and SPINVAE-2 models, chosen according to validation dataset performance then evaluated on the test set. SPINVAE-2 performs substantially better interpolations than the linear method. It differs from the first SPINVAE model in several aspects, including the AR loss, the absence of an audio encoder and the number of latent dimensions. They significantly enhance this second version of the model, and are analyzed separately in Section VI.

Compared to the SMT baseline, the linearity of SPINVAE-2 interpolations is similar (2.2 % worse on average). It seems to indicate that the *intermediateness* [69] of interpolated presets is good, and that SPINVAE-2 is an adequate method to intuitively create new presets.

However, the smoothness is 213% worse than the baseline's, i.e., smoothness values are approximately three times those of SMT on average. All six waveform morphing techniques from Tab. I are objectively much smoother than SPINVAE-2. This was expected because synthesis engines dedicated to morphing have very smooth relationships between synthesis parameters and output sound modifications. In contrast, when performing interpolations using a common synthesizer such as the DX7, the decoder occasionally changes the value of a non-differentiable synthesis parameter such as algorithm, envelope curve or waveform type. Hence, a good smoothness is arguably impossible to reach.

Compared to waveform morphing techniques, the main strength of SPINVAE-2 modeling is that it does not require a specific synthesis engine (Fig. 2). The model has learned how to interpolate presets without being explicitly informed that the controlled synthesizer is a DX7. Therefore, it has a broad applicability. Compared to the only other technique for black-box preset interpolation, i.e. linear interpolation, SPINVAE models are more linear and smoother.

Tab. I also reports the computational requirements for analysis and resynthesis of 4 seconds of 48 kHz audio for all models, except SMT which had to run at 44.1 kHz in order to properly reconstruct pitch. Analysis durations include the processing of the two input sounds, while resynthesis durations correspond to a single output sound. The same machine with an Intel Core i7-10700KF CPU (8-core, 3.80GHz) was used to run all models on the whole test set. Compared to waveform morphing, SPINVAE-2 requires a lightweight analysis (preset encoding) and resynthesis (preset decoding and waveform synthesis). For instance, the analysis is approximately 10^4 times faster than SMT, and 200 times faster than differentiable HpN. It is efficient because the dimensionality of a preset is much lower than that of 48 kHz audio. Encoding a preset, performing a linear latent interpolation and decoding presets are fast operations. Synthesis is also fast because synthesizers are heavily optimized by developers and manufacturers.

VI. SPINVAE-2 Model Analysis

A. Architecture

Interpolation linearity of some variants of the SPINVAE-2 model, relative to the SMT baseline, is provided in Tab. II. Each variant was trained five times using different initial random seeds for latent noise samplers [11] and for shuffling the training dataset. Objective smoothness is consistently approximately three times worse, which has been explained in sub-section V-B. Hence, it is not reported in this section.

TABLE II
Variation of interpolation linearity of different architectures, relative to the SMT baseline. Means and standard deviations for five training runs.

Model	Linearity (↑)
SPINVAE-2	$\mathbf{-3.0} \pm 0.5$ %
— without timbre regularization	-6.7 ± 0.3 %
Preset-only AR-VAE (no audio)	-3.6 ± 0.5 %
Preset-only VAE	-14.0 ± 2.1 %
Bi-modal AR-VAE, additive	-7.3 ± 0.2 %
Bi-modal AR-VAE, contrastive	-6.7 ± 0.3 %

Timbre regularization is an important factor. For the model trained without the AR loss, Linearity is reduced by more than three percentage points compared to SPINVAE-2. This model is nonetheless much better than the linear preset interpolation (Tab. I). Thanks to the audio decoder, it has learned latent representations that can be easily decoded into mel-spectrograms and are likely to hold meaningful perceptual information.

Removing this audio decoder to obtain the preset-only AR-VAE only slightly reduces interpolation performance. The straightforward timbre regularization seems to have a stronger influence on latent representations than the CNN audio decoder. The preset-only VAE learns representations of presets without additional knowledge of timbre or audio regularization, and is the worst model.

Two bi-modal AR-VAEs were also tested. The additive version is obtained by appending an audio encoder to SPINVAE-2, and by summing outputs from the preset and audio encoders. This additive bi-modal VAE presents deteriorated performances. This is remarkable, because an extra encoder should allow better latent representations. We hypothesize that this results from the basic fusion of modalities. This does not enforce the learning of fully-shared representations, i.e., some latent coordinates may encode spectral information while others may encode preset parameters.

Hence, a second bi-modal VAE with contrastive alignment of preset and audio latent codes has been implemented. It can be seen as two VAEs (a preset VAE and an audio VAE) running in parallel, akin to [74]. A contrastive loss [75] forces the two VAEs to learn shared representations. This loss maximizes the dot product of latent codes for paired data (e.g. a preset and its corresponding mel-spectrogram) while minimizing the dot product of unrelated latent codes. Contrastive learning has been proven to be a strong method for pre-training models. Then, they can be used for multiple downstream tasks or zero-shot inference [76] such as automatic labeling of audio samples. However, it does not seem to help in improving the specific task of smooth preset decoding. Compared to SPINVAE-2, the contrastive AR-VAE is underperforming, even though it has en extra CNN encoder and a more complex training procedure.

B. Latent Dimensions

Our previous study [12] has experimented with a latent size L_z constrained to 256. The SPINVAE-2 architecture (Fig. 4) enables any latent size, and Fig. 6 indicates that $L_z = 512$ provides the best interpolation performance.

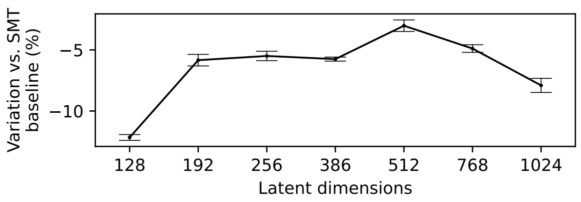

Fig. 6. Evolution of SPINVAE-2 interpolation linearity with latent size. Means and standard deviations for five training runs.

As described in sub-section III-D, the AR loss is computed on 38 coordinates for $L_z = 128$, and 152 coordinates otherwise. The reduced amount of timbre regularization could explain the poorer performance with $L_z = 128$. With $L_z \geq 192$, the effect of latent size on performance is not clear.

Several factors are involved. On one hand, latent vectors must be large enough to enable a good reconstruction process [2]. On the other hand, increasing latent size may cause latent collapse, i.e. some latent coordinates become uninformative and ignored by the decoders. This phenomenon is frequent with a powerful non-Gaussian decoder [77] such as $p(\mathbf{u}|\mathbf{z})$ in our study. It can also be selective [77], possibly collapsing the timbre-regularized dimensions. For DX7 presets modeling, $L_z = 512$ seems to provide a good ratio of regularized over unregularized dimensions, and enough latent units to properly reconstruct presets (as shown in Tab. III). This optimum would probably be different with another synthesizer.

C. Data Augmentation

Thanks to the way it encodes and decodes presets, the SPINVAE-2 architecture can applied to any synthesizer. However, some synthesizers do not provide as many presets as are available for the DX7. To address this, we provide two data augmentation techniques to increase the training dataset's size.

The first technique uses small presets variations based on knowledge of the DX7 synthesizer. For instance, the algorithm can be changed to a similar one, and a small noise can be added to numerical parameters. It is named *4× data augmentation* in Tab. III, and is used to train SPINVAE-2.

TABLE III
IMPACT OF DATA AUGMENTATION ON INTERPOLATION LINEARITY (RELATIVE TO SMT), AND ON RECONSTRUCTION ERROR FOR START AND END PRESETS.

Training dataset	**Interpolation**	**Reconstruction error**		
	Linearity (↑) %	Accuracy %	L1 error $\times 10^{-3}$	MFCCD
No data augment.	−4.4	99.42	3.33	4.5
4× data augment.	**−2.2**	99.86	2.33	4.1
+5k rand. presets	−4.4	99.95	1.41	3.2
+10k rand. presets	−6.0	99.95	0.97	2.8
+24k rand. presets	−7.5	**99.96**	**0.52**	**1.8**

The second technique is based on the marginal distributions $p(u_i)$ of synthesis parameters u_i. The marginals $p(u_i)$ are estimated from the training dataset, then new random presets are obtained by sampling each $u_i \sim p(u_i)$ independently. A third technique could be to sample some u_i from uniform distributions. Unfortunately, it generates a lot of noise-like or inaudible DX7 presets, thus has not been retained.

Tab. III demonstrates that the proposed 4× data augmentation technique yields the best interpolation performance. The augmented examples, obtained from handcrafted variations, are relatively similar to the original presets. They seem to help the model learn to encode inputs into a more linear latent space. Regarding parameter values, the reconstruction of starting and ending presets is nearly perfect. All variants present an average MFCCD below the 10–15 threshold.

The use of extra random presets, appended to the training set, further reduces the reconstruction error. However, it also slightly decreases the interpolation quality, which seems to indicate the VAE learns worse representations of the data. This could be mitigated by advanced techniques such as GANs [49] instead of the naive sampling from marginals. They could generate presets whose likelihood would be closer to that of human-made presets from the original dataset.

VII. Subjective Preset Interpolation Evaluation

Objective results from Section V indicate that SPINVAE-2 interpolation is better than the linear preset interpolation, designated as the reference interpolation in this section. An experiment has been conducted to provide a perceptive evaluation of the SPINVAE-2 model, a comparison with the linear reference, and to verify how subjective measurements relate to the objective evaluation.

Section V has also shown that SPINVAE-2 performs less smoothly than the waveform morphing techniques; however, these techniques have a narrower scope as they cannot be used with arbitrary synthesizers. This lack of smoothness appears evident during informal listening tests and has not been studied in the context of a formal comparative experiment.

A. Experimental Protocol

A 15-minute anonymous online survey presented sequences of sounds. The 25 participants were informed that they could take a break or stop the experiment at any time, and that only their answers were recorded. Contents were paired, i.e. if a reference interpolation between presets $\mathbf{u}^{(n)}$ and $\mathbf{u}^{(m)}$ was presented to the subjects, then the corresponding SPINVAE-2 interpolation between $\mathbf{u}^{(n)}$ and $\mathbf{u}^{(m)}$ would be presented as well. A total of 292 reference sequences and 292 corresponding SPINVAE-2 sequences, made of $T = 7$ steps, were randomly chosen from the test dataset containing 1515 interpolations. The experiment was divided into two main phases, which are described hereafter.

1) Subjective Ratings: a single sequence was presented, and subjects were asked how *smooth*, and how *natural* it was. They gave their ratings on 1 (*not smooth/natural at all*) to 5 (*very smooth/natural*) integer scales, similar to Mean Opinion Score scales. They were given the following details:

> ”*Very smooth* means that two consecutive samples always sound quite similar. *Not smooth at all* means that two consecutive samples always sound very different.”
>
> ”*Very natural* means that the progression of sounds, from the start to end samples, is performed as you could have expected. *Not natural at all* means that you mostly heard unexpected samples between the start and end samples.”

Participants were able play individual sounds from the sequence, or play the whole sequence automatically. They could play sounds as many times as they wanted.

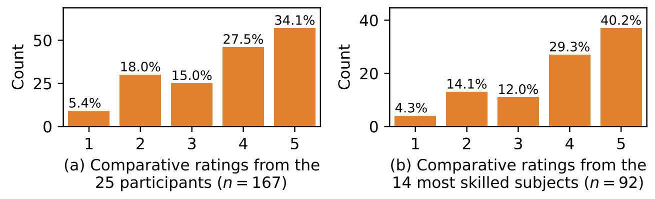

Fig. 7. Subjective comparisons between paired interpolations. ”5” means ”SPINVAE-2 is definitely better than the reference” and ”1” means ”the reference is definitely better than SPINVAE-2”.

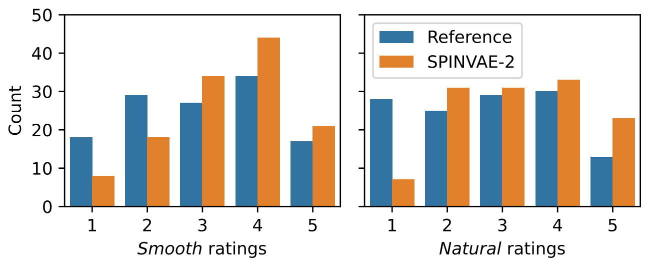

Fig. 8. Subjective evaluations of smoothness and naturalness. $n = 125$ for each model and each type of rating (500 measurements total).

2) Comparative Ratings: two sequences named A and B were simultaneously presented. They corresponded to paired reference/SPINVAE-2 interpolations randomly ordered as A/B or B/A. Participants had to identify the best sequence, i.e. the one that sounded the smoothest and the most natural to them. Five answers were allowed: A is definitely better; A is marginally better; they are equally good; B is marginally better; B is definitely better.

B. Task Difficulty

A recent study about sound morphing [10] considered such an subjective evaluation task to be very difficult. It indeed requires the subjects to keep multiple auditory objects in working memory. Non-verbal information is maintained for a few second, but is significantly degraded after twelve seconds [78]. Therefore, in order to reduce the cognitive load, we limit interpolations to $T = 7$ steps and sounds to 2.5 seconds, and provide a minimalistic interface.

Consequently, our results challenge the statement that sound morphing is hard to evaluate. At the end of the experiment, participants were asked how difficult the ratings and comparison were, on a Likert scale from 1 (very difficult) to 5 (very easy). The average answer is 3.2 for ratings (close to neutral), and 3.7 for comparisons (close to easy). The task should nevertheless be easier for musicians, whose auditory working memory is enhanced compared to non-musicians [79].

C. Comparison Results

Subjective comparisons results are presented on Fig. 7(a). It demonstrates a clear superiority of our model over the reference, which is consistent with the objective evaluation.

During the experiment’s introduction, participants were asked to evaluate their own skills regarding musical practice, music theory and sound synthesis. They had to choose one of the four levels described by a short text, which allowed us to split the cohort into two groups. Interestingly, Fig. 7(b) shows that the most skilled subjects have an even stronger preference for SPINVAE-2 interpolations. Because these participants possess cognitive advantages for the task [79], we consider Fig. 7(b) to present the most accurate results regarding the model’s subjective improvements over the reference method.

D. ”Smooth” and ”Natural” Subjective Ratings

The paired subjective ratings are presented in Fig. 8. The average smooth scores are 3.4 for SPINVAE-2 and 3.0 for the reference, while the average natural scores are 3.3 and 2.8. Compared to the reference, the distributions of SPINVAE-2 smoothness and naturalness are significantly shifted upwards (p-values from the Wilcoxon signed-rank tests are 1.4×10^{-3} and 1.9×10^{-4}, respectively). This further proves the enhanced quality of model-based interpolations.

Another observation is that while the reference method computes a lot of subpar interpolations, because of the non-linear relationships between synthesis parameters, it also sometimes performs well. E.g., if presets $\mathbf{u}^{(n)}$ and $\mathbf{u}^{(m)}$ employ the same FM algorithm and have similar configurations of oscillators, the linear parametric interpolation can be smooth and natural. Thus, it is an appropriate baseline to compare our model to.

In order to analyze the relationship between the natural and smooth criteria, a principal component analysis has been performed on the set of ratings. The first eigenvector lies almost perfectly on the ”smooth = natural” diagonal direction, and it explains 87.9% of the variance. Hence, instead of evaluating the two criteria separately, participants could have been asked to provide a single ”smooth and natural” rating.

E. Objective Correlates

The objective evaluation metrics introduced in our works are based on relevant studies about sound morphing and model-based interpolation. Nonetheless, their correlations with subjective ratings can be computed. For each pair of objective metric and subjective rating, e.g. objective smoothness of the ACTM depth and subjective naturalness rating, a set of 250 pairs of values is available. The Pearson correlations for all these sets are presented on Fig. 9.

Almost all proposed metrics are significantly correlated to subjective measurements, which further validates our objective evaluation method. Only two out of 74 metrics are not significantly correlated to the natural and smooth subjective scores. One audio feature, amplitude modulation, could be excluded from future objective evaluations.

The most correlated seem to be the semantic descriptors of timbre, such as depth, warmth and boominess from the ACTM extractor [17]. Spectral variation, estimated by TT [16], is also one of the best objective correlates. This aligns with previous research on timbre [70], [80]. Nevertheless, the relatively low correlation of LAT and spectral centroid metrics is unexpected and in contrast with their usual relevance [70], [80].

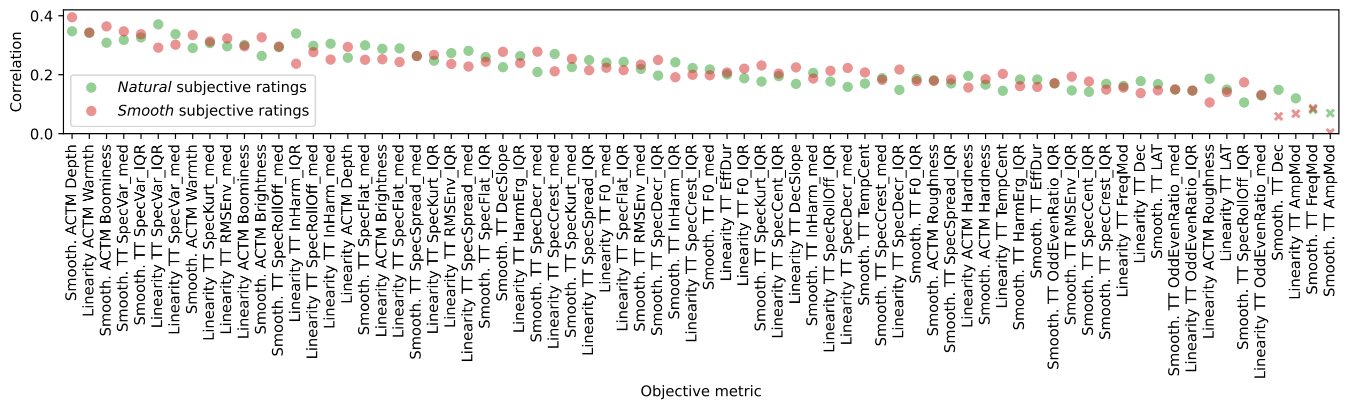

Fig. 9. Pearson correlation coefficients between objective interpolation quality metrics and *natural/smooth* subjective ratings. Significant correlations (p-value < 0.05, $n = 250$) are represented by dots, otherwise crosses are used.

Correlation values might not be as high as expected, but we argue that they are diminished by the subjective measurements' noisiness. Among the 292 uniformly sampled sequences, 29 have been presented to at least two subjects. They allow us to prove, as follows, that different participants rate the same sequence quite differently. For a given interpolation from preset $\mathbf{u}^{(n)}$ to $\mathbf{u}^{(m)}$ and method (reference or SPINVAE-2), the average rating difference is 1.1 (27.5% of the rating scale) for *smooth* and 1.4 (35%) for *natural*. The maximum difference between two participants is 3 for both criteria.

Another source of noise arises from the computation of audio features. Some numerical instabilities were observed with both TT and ACTM feature extractors. Moreover, some estimators may provide inaccurate results [81]. These sources of error can propagate to metrics computations and further reduce correlations with subjective measurements.

Regarding the smoothness and non-linearity metrics overall, neither appears to surpass the other. For instance, the smoothness of ACTM depth is a better objective correlate than its non-linearity, whereas the opposite can be observed for the RMS envelope. This confirms that using two types of objective metrics, as discussed in sub-section IV-B, is appropriate.

VIII. Conclusion

This work introduces a model named SPINVAE-2 and dedicated to preset interpolation for black-box synthesizers. It is structured as a VAE with timbre regularization. The powerful modeling, based on multi-head attention networks and DLM distributions, allows SPINVAE-2 to learn useful latent representations of presets without being explicitly informed what the underlying synthesizer is. These representations are further aligned with perceived dimensions of timbre using attribute-based regularization of the latent space. Finally, a linear interpolation of latent vectors can be decoded into presets to obtain an approximately perceptually linear morphing. Audio examples are available on the companion website[5].

The model can be applied to virtually any commercial synthesizer. It does not incur the high engineering costs associated with the development of new differentiable modules. An FM synthesizer, which is notoriously hard to program because of the intricate relationships between numerous synthesis parameters, has been selected as the main subject for this study.

In order to compare models, an objective and reproducible evaluation procedure has been introduced. It relies on two metrics, linearity and smoothness, computed on a set of 37 audio features. Results demonstrate a substantial enhancement in morphing quality of SPINVAE-2 compared to the linear preset interpolation for a black-box synthesizer. Compared to classic waveform morphing techniques with a constrained synthesis engine, the objective perceptual linearity of SPINVAE-2 interpolation is similar, while the smoothness is much worse. SPINVAE-2 is nonetheless orders of magnitude faster and is able to better reconstruct original sounds.

The model and evaluation method are open-source and provided separately. Results from a subjective experiment are also presented. They rely on a large set of presets, which span a wide range of instruments and sound textures. Perceptual comparisons show that our model outperforms the linear preset interpolation by a large margin. Perceptual ratings lead to a similar conclusion, and also demonstrate significant correlations with the objective evaluation's metrics.

While the model is focused on preset interpolation, it also guarantees a regularized latent space that enables many other practical uses. Extrapolations beyond dataset presets can be computed. Two- or three-dimensional exploration interfaces, based on dimensionality reduction techniques, could integrate interpolations instead of only displaying the manufacturer's presets. New modulation techniques also arise. E.g., small preset variations can be obtained by sampling latent codes from posterior distributions and new presets from decoder distributions. This can make synthesizer presets evolve over time and sound more dynamic.

[5]https://gwendal-lv.github.io/spinvae2

References

[1] N. Masuda and D. Saito, "Improving semi-supervised differentiable synthesizer sound matching for practical applications," *IEEE/ACM Trans. Audio, Speech, Lang. Process. (TASLP)*, vol. 31, pp. 863–875, 2023.

[2] F. Roche, T. Hueber, M. Garnier, S. Limier, and L. Girin, “Make that sound more metallic: Towards a perceptually relevant control of the timbre of synthesizer sounds using a variational autoencoder,” *Trans. Int. Soc. Music Inf. Retr. (TISMIR)*, vol. 4, pp. 52–66, 2021.

[3] G. Krekovic, “Insights in habits and attitudes regarding programming sound synthesizers: A quantitative study,” in *Proceedings of the 16th Sound and Music Computing Conference*, 2019.

[4] M. J. Yee-King, L. Fedden, and M. d’Inverno, “Automatic programming of VST sound synthesizers using deep networks and other techniques,” *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 2, no. 2, pp. 150–159, 2018.

[5] O. Barkan, D. Tsiris, O. Katz, and N. Koenigstein, “InverSynth: Deep estimation of synthesizer parameter configurations from audio signals,” *IEEE/ACM Trans. Audio, Speech, Lang. Process. (TASLP)*, vol. 27, no. 12, pp. 2385–2396, 2019.

[6] P. Esling, N. Masuda, A. Bardet, R. Despres, and A. Chemla-Romeu-Santos, “Flow synthesizer: Universal audio synthesizer control with normalizing flows,” *Applied Sciences*, vol. 10, no. 1, 2020.

[7] G. Le Vaillant, T. Dutoit, and S. Dekeyser, “Improving synthesizer programming from variational autoencoders latent space,” in *Int. Conf. Digit. Audio Eff. (DAFx)*, 2021, pp. 276–283.

[8] C. Mitcheltree and H. Koike, “SerumRNN: Step by step audio VST effect programming,” in *Int. Conf. Comput. Intell. Music, Sound, Art Des. (Part of EvoStar)*, 2021, pp. 218–234.

[9] Z. Chen, Y. Jing, S. Yuan, Y. Xu, J. Wu, and H. Zhao, “Sound2Synth: Interpreting sound via FM synthesizer parameters estimation,” in *Int. Joint Conf. Artif. Intell. (IJCAI)*, 2022.

[10] M. Caetano and X. Rodet, “Musical instrument sound morphing guided by perceptually motivated features,” *IEEE Trans. Audio, Speech, Lang. Process. (TASLP)*, vol. 21, no. 8, pp. 1666–1675, 2013.

[11] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *Int. Conf. Learn. Represent. (ICLR)*, 2014.

[12] G. Le Vaillant and T. Dutoit, “Synthesizer preset interpolation using transformer auto-encoders,” in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2023.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017.

[14] A. Pati and A. Lerch, “Is disentanglement enough? on latent representations for controllable music generation,” in *Int. Soc. Music Inf. Retr. (ISMIR) Conf.*, 2021, pp. 517–524.

[15] ——, “Attribute-based regularization of latent spaces for variational auto-encoders,” *Neural Comput. Appl.*, vol. 33, no. 9, pp. 4429–4444, 2021.

[16] G. Peeters, B. Giordano, P. Susini, and N. Misdariis, “The timbre toolbox: Extracting audio descriptors from musical signals,” in *J. Acoust. Soc. Am.*, vol. 130, no. 5, 2011.

[17] A. Pearce, S. Safavi, T. Brookes, R. Mason, W. Wang, and M. Plumbley, “Timbral characterisation tools for semantically annotating non-musical content,” in *AudioCommons project, Deliverable 5.8*, January 2019.

[18] 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 *Int. Conf. Mach. Learn. (ICML)*, 2017, p. 1068–1077.

[19] A. Caillon and P. Esling, “RAVE: A variational autoencoder for fast and high-quality neural audio synthesis,” in *arXiv:2111.05011v2*, 2021.

[20] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, “DDSP: Differentiable digital signal processing,” in *Int. Conf. Learn. Represent. (ICLR)*, 2020.

[21] K. Subramani, P. Rao, and A. D’Hooge, “Vapar synth - a variational parametric model for audio synthesis,” in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2020, pp. 796–800.

[22] F. Caspe, A. McPherson, and M. Sandler, “DDX7: Differentiable FM synthesis of musical instrument sounds,” *Int. Soc. Music Inf. Retr. (ISMIR) Conf.*, pp. 608–616, 2022.

[23] C. J. Steinmetz, N. J. Bryan, and J. D. Reiss, “Style transfer of audio effects with differentiable signal processing,” *J. Audio Eng. Soc.*, vol. 70, no. 9, pp. 708–721, 2022.

[24] A. Vinay and A. Lerch, “Evaluating generative audio systems and their metrics,” *Int. Soc. Music Inf. Retr. (ISMIR) Conf.*, pp. 858–865, 2022.

[25] 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,” *Int. Conf. Learn. Represent. (ICLR)*, 2017.

[26] E. Tellman, L. Haken, and B. Holloway, “Timbre morphing of sounds with unequal numbers of features,” *J. Audio Eng. Soc*, vol. 43, no. 9, pp. 678–689, 1995.

[27] K. Fitz, L. Haken, S. Lefvert, C. Champion, and M. O’Donnell, “Cell-Utes and Flutter-Tongued Cats: Sound Morphing Using Loris and the Reassigned Bandwidth-Enhanced Model,” *Comput. Music J.*, vol. 27, no. 3, pp. 44–65, 09 2003.

[28] T. Brookes and D. Williams, “Perceptually-motivated audio morphing: Brightness,” in *Audio Eng. Soc. Conv. 122*, May 2007.

[29] S. Kazazis, P. Depalle, and S. McAdams, “Sound morphing by audio descriptors and parameter interpolation,” in *Int. Conf. Digit. Audio Eff. (DAFx)*, 2016, pp. 145–151.

[30] M. Caetano, “Morphing musical instrument sounds with the sinusoidal model in the sound morphing toolbox,” in *Perception, Representations, Image, Sound, Music*. Springer International Publishing, 2021.

[31] D. O’Shaughnessy, “Linear predictive coding,” *IEEE Potentials*, vol. 7, no. 1, pp. 29–32, 1988.

[32] M. Caetano and X. Rodet, “Sound morphing by feature interpolation,” in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2011, pp. 161–164.

[33] H. Kawahara, H. Banno, T. Irino, and P. Zolfaghari, “Algorithm amalgam: morphing waveform based methods, sinusoidal models and straight,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2004.

[34] M. Caetano and X. Rodet, “A source-filter model for musical instrument sound transformation,” in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2012, pp. 137–140.

[35] W. Ahmad, H. Hacihabiboglu, and A. M. Kondoz, “Morphing of transient sounds based on shift-invariant discrete wavelet transform and singular value decomposition,” in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2009, pp. 297–300.

[36] S. Siddiq, “Real-time morphing of impact sounds,” in *Audio Eng. Soc. Conv. 139*, Oct 2015.

[37] ——, “Morphing of granular sounds,” in *Int. Conf. Digit. Audio Eff. (DAFx)*, 2015.

[38] A. Primavera, F. Piazza, and J. D. Reiss, “Audio morphing for percussive hybrid sound generation,” in *Audio Eng. Soc. Int. Conf.*, Mar 2012.

[39] M. Slaney, M. Covell, and B. Lassiter, “Automatic audio morphing,” in *Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 2, 1996, pp. 1001–1004.

[40] A. Olivero, P. Depalle, B. Torrésani, and R. Kronland-Martinet, “Sound morphing strategies based on alterations of time-frequency representations by gabor multipliers,” in *Audio Eng. Soc. Int. Conf.*, Mar 2012.

[41] G. Roma, O. Green, and P. A. Tremblay, “Audio morphing using matrix decomposition and optimal transport,” in *Int. Conf. Digit. Audio Eff. (DAFx)*, 2020.

[42] D. Fitzgerald, “Harmonic/percussive separation using median filtering,” in *Int. Conf. Digit. Audio Eff. (DAFx)*, 2010.

[43] D. Griffin and J. Lim, “Signal estimation from modified short-time fourier transform,” *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 2, pp. 236–243, 1984.

[44] Z. Pruša and P. L. Søndergaard, “Real-time spectrogram inversion using phase gradient heap integration,” in *Int. Conf. Digit. Audio Eff. (DAFx)*, 2016, pp. 17–21.

[45] J. W. Kim, R. Bittner, A. Kumar, and J. P. Bello, “Neural music synthesis for flexible timbre control,” in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2019, pp. 176–180.

[46] K. Tatar, D. Bisig, and P. Pasquier, “Latent timbre synthesis: Audio-based variational auto-encoders for music composition and sound design applications,” vol. 33, no. 1, p. 67–84, 2021.

[47] J. C. Brown, “Calculation of a constant Q spectral transform,” *J. Acoust. Soc. Am.*, vol. 89, no. 1, pp. 425–434, 1991.

[48] Y. Zou, J. Liu, and W. Jiang, “Non-parallel and many-to-one musical timbre morphing using ddsp-autoencoder and spectral feature interpolation,” in *International Conference on Culture-oriented Science and Technology (ICCST)*, 2021, pp. 144–148.

[49] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 27, 2014.

[50] C. Gupta, P. Kamath, Y. Wei, Z. Li, S. Nanayakkara, and L. Wyse, “Towards controllable audio texture morphing,” in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2023.

[51] C. Scaletti, “Computer music languages, Kyma, and the future,” *Computer Music Journal*, vol. 26, no. 4, pp. 69–82, 2002.

[52] M. Caetano, C. Saitis, and K. Siedenburg, “Audio content descriptors of timbre,” in *Timbre: Acoustics, Perception, and Cognition*, K. Siedenburg, C. Saitis, S. McAdams, A. N. Popper, and R. R. Fay, Eds. Springer International Publishing, 2019, pp. 297–333.

[53] M. Blaauw and J. Bonada, “Modeling and transforming speech using variational autoencoders,” in *Proc. Interspeech*, 2016, pp. 1770–1774.

[54] D. Berthelot, C. Raffel, A. Roy, and I. Goodfellow, "Understanding and improving interpolation in autoencoders via an adversarial regularizer," in *Int. Conf. Learn. Represent. (ICLR)*, 2019.

[55] J. M. Chowning, "The synthesis of complex audio spectra by means of frequency modulation," *J. Audio Eng. Soc.*, vol. 21, no. 7, pp. 526–534, 1973.

[56] B. Evans, *FM synthesis and morphing in Csound: from percussion to brass*. MIT Press, 2000, p. 198–206.

[57] Native Instruments, "FM8 Manual," https://www.native-instruments.com/fileadmin/ni_media/producer/fm8/downloads/FM8-book_EN_ebook.pdf, 2006, accessed on November 13, 2023.

[58] C. J. Steinmetz, J. Pons, S. Pascual, and J. Serrà, "Automatic multitrack mixing with a differentiable mixing console of neural audio effects," in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2021, pp. 71–75.

[59] M. A. Martínez Ramírez, O. Wang, P. Smaragdis, and N. J. Bryan, "Differentiable signal processing with black-box audio effects," in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2021, pp. 66–70.

[60] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[61] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved variational inference with inverse autoregressive flow," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 29, 2016.

[62] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using Real NVP," *Int. Conf. Learn. Represent. (ICLR)*, 2017.

[63] P. Lippe and E. Gavves, "Categorical normalizing flows via continuous transformations," in *Int. Conf. Learn. Represent. (ICLR)*, 2021.

[64] 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 *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2020.

[65] M. Petrovich, M. J. Black, and G. Varol, "Action-conditioned 3D human motion synthesis with transformer VAE," in *Int. Conf. Comput. Vis. (ICCV)*, 2021.

[66] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "PixelCNN++: Improving the PixelCNN with discretized logistic mixture likelihood and other modifications," in *Int. Conf. Learn. Represent. (ICLR)*, 2017.

[67] P. Esling, A. Chemla-Romeu-Santos, and A. Bitton, "Generative timbre spaces: regularizing variational auto-encoders with perceptual metrics," *Int. Conf. Digit. Audio Eff. (DAFx)*, pp. 369–376, 2018.

[68] F. Locatello, M. Tschannen, S. Bauer, G. Rätsch, B. Schölkopf, and O. Bachem, "Disentangling factors of variations using few labels," in *Int. Conf. Learn. Represent. (ICLR)*, 2020.

[69] M. Caetano and N. Osaka, "A formal evaluation framework for sound morphing," in *Proc. Int. Comput. Music Conf.*, 2012, pp. 104–107.

[70] S. McAdams, S. Winsberg, S. Donnadieu, G. De Soete, and J. Krimphoff, "Perceptual scaling of synthesized musical timbres: Common dimensions, specificities, and latent subject classes," *Psychological Research*, vol. 58, no. 3, pp. 177–192, 1995.

[71] K. Siedenburg, I. Fujinaga, and S. McAdams, "A comparison of approaches to timbre descriptors in music information retrieval and music psychology," *J. New Music Res.*, vol. 45, no. 1, pp. 27–41, 2016.

[72] H. Terasawa, M. Slaney, and J. Berger, "Perceptual distance in timbre space," in *Int. Conf. Audit. Display*, 2005, pp. 1–8.

[73] R. Huber and B. Kollmeier, "PEMO-Q—A new method for objective audio quality assessment using a model of auditory perception," *IEEE Trans. Audio Speech Lang. Process.*, vol. 14, no. 6, pp. 1902–1911, 2006.

[74] X. Zheng, Z. Wu, H. Meng, and L. Cai, "Contrastive auto-encoder for phoneme recognition," in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2014, pp. 2529–2533.

[75] A. Radford *et al.*, "Learning transferable visual models from natural language supervision," in *Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 8748–8763.

[76] B. Elizalde, S. Deshmukh, M. A. Ismail, and H. Wang, "CLAP: Learning audio concepts from natural language supervision," in *IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2023.

[77] B. Dai, Z. Wang, and D. Wipf, "The usual suspects? Reassessing blame for VAE posterior collapse," in *Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 2313–2322.

[78] A. Soemer and S. Saito, "Maintenance of auditory-nonverbal information in working memory," *Psychonomic Bulletin & Review*, vol. 22, no. 6, pp. 1777–1783, 2015.

[79] K. J. Pallesen, E. Brattico, C. J. Bailey, A. Korvenoja, J. Koivisto, A. Gjedde, and S. Carlson, "Cognitive control in auditory working memory is enhanced in musicians," *PloS one*, vol. 5, no. 6, 2010.

[80] J. M. Grey, "Multidimensional perceptual scaling of musical timbres," *J. Acoust. Soc. Am.*, vol. 61, no. 5, pp. 1270–1277, 1977.

[81] S. Kazazis, N. Esterer, P. Depalle, and S. McAdams, "A performance evaluation of the Timbre Toolbox and the MIRtoolbox on calibrated test sounds," in *Int. Symp. Musical Acoust.*, June 2017.

Gwendal Le Vaillant holds an engineering degree from Grenoble-INP Ense3 (Université Grenoble-Alpes, France) with a specialization in digital signal processing and computer science. He has been working as a lecturer at the ISIB engineering institute (Brussels, Belgium) since 2015. He was also a research project manager in the fields of embedded electronics and audio spatialization. He is currently a PhD student under the supervision of Prof. Thierry Dutoit at ISIA lab (Information, Signal and Artificial Intelligence), University of Mons, Belgium. His current research field is machine learning applied to sound synthesis.

Thierry Dutoit graduated as an electrical engineer and Ph.D. in 1988 and 1993 from UMONS, Belgium, where he currently leads the ISIA (Information, Signal and Artificial Intelligence) Lab and the NUMEDIART Institute for creative technology. In 1995, he initiated the MBROLA project for free multilingual speech synthesis, which provided speech synthesis for 29 OSs and 75 voices, in 38 languages. Between 1996 and 1998, he spent 16 months at AT&T-Bell Labs, in Murray Hill (NJ) and Florham Park (NJ), working on the NextGen Speech synthesizer. He co-created ACAPELA-GROUP, a European company specialized in TTS products. He has been an Associate Editor of the IEEE Transactions on Speech and Audio Processing (2003-2006) and the president of ISCA's SynSIG interest group on speech synthesis, from 2007 to 2010. He has been part of the Speech and Language Technical Committee of the IEEE (2009-2011). In 2005, he initiated the eNTERFACE 4-weeks summer workshops on Multimodal Interfaces, and coordinated its steering committee for 12 years. He co-created the CLICK Creative Innovation Center in Mons in 2016, where he serves as a scientific director, and the TRAIL interuniversity AI research Institute in 2020 (5 universities and 4 research centers in Wallonia-Brussels). T. Dutoit is also a member of the Scientific Committee of CIRMMT (Montréal), IRCAM (Paris) and IMT-Nord Europe (Lille).