SYNTHESIZER PRESET INTERPOLATION USING TRANSFORMER AUTO-ENCODERS

Gwendal Le Vaillant [1,2], *Thierry Dutoit* [1]

[1] Information, Signal and Artificial Intelligence (ISIA), University of Mons, Belgium
[2] HE2B-ISIB Research Institute, Brussels, Belgium

ABSTRACT

Sound synthesizers are widespread in modern music production but they increasingly require expert skills to be mastered. This work focuses on interpolation between presets, i.e., sets of values of all sound synthesis parameters, to enable the intuitive creation of new sounds from existing ones.

We introduce a bimodal auto-encoder neural network, which simultaneously processes presets using multi-head attention blocks, and audio using convolutions. This model has been tested on a popular frequency modulation synthesizer with more than one hundred parameters. Experiments have compared the model to related architectures and methods, and have demonstrated that it performs smoother interpolations. After training, the proposed model can be integrated into commercial synthesizers for live interpolation or sound design tasks.

Index Terms— Synthesizer, Sound, Interpolation, Timbre, Transformer, VAE

1. INTRODUCTION

Sound synthesizers can generate audio signals whose timbre ranges from acoustic instruments to entirely novel sound textures. They are ubiquitous in modern music production, and their use even defines some new music genres. Synthesis processes are controlled using sets of parameters, called presets, which are usually large [1–3]. They require expert knowledge to be created and handled, so that a lot of presets are provided by synthesizer manufacturers and developers themselves.

Our research addresses the problem of preset interpolation, for musicians to be able to discover new sounds in-between two reference presets, or to create smooth transitions from a preset to another. This requires to manipulate dozens of parameters simultaneously, with intricate relationships between parameters and synthesized audio, and interactions between parameters themselves.

While several recent works have used neural networks to match synthesizer presets with input sounds [1–6], ours is the first to formally focus on preset interpolation. Its main contribution is a model that enables smoother interpolations, compared to related generative architectures. It is also the first model that successfully handles synthesizer presets as sequences using Transformer [7] encoders and decoders, and models numerical synthesis parameters using Discretized Logistic Mixture (DLM) distributions [8].

2. RELATED WORK

2.1. Neural Audio Synthesis

Models such as WaveNet [9] can be trained to synthesize raw audio waveforms using Convolutional Neural Networks (CNNs). In order to prevent per-sample computations, recent works have tried to learn synthesis processes akin to commercially available music synthesizers. Some are based on source-filter models [10, 11], whereas others [12] model a differentiable FM (Frequency Modulation) synthesis architecture similar to a synthesizer named DX7.

However, these neural networks include the synthesis itself, and they are trained using gradient descent. Therefore, they can't be applied to existing commercial synthesizers [3–5], which rely on non-differentiable processes.

2.2. Sound Matching

Various neural network architectures have been successfully used to search for synthesis parameters that correspond best to an input sound. Long Short-Term Memory (LSTM, [13]) neural networks have been used to infer a preset from Mel-Frequency Cepstral Coefficients (MFCCs) [1]. Other architectures were based on CNNs to process audio spectrograms or raw waveforms [2, 4, 6], or a combination of CNN, LSTM and Multilayer Perceptron (MLP) blocks to process different types of input audio features [3].

Several works [1, 3, 4] have focused on the sound matching task for a software implementation of the well-established DX7 FM synthesis architecture. It is known to be able to synthesize a wide variety of digital- and natural-sounding instruments [12], while being notoriously hard to handle considering the large amount of synthesis parameters (155). Experiments presented in this paper focus on this non-differentiable FM synthesis architecture.

2.3. Generative Models

Among the previously cited works, a few [2, 4] are based on generative models, which first encode input audio data $\mathbf{x}$ into a latent vector $\mathbf{z}$, then try to reconstruct the audio and infer the preset from $\mathbf{z}$. After training, latent vectors can be sampled from a prior distribution $p(\mathbf{z})$ in order to generate new audio samples and new presets.

A common framework to learn both latent representations and a generative model is the Variational Auto-Encoder (VAE) [14]. It learns an approximate posterior distribution $q(\mathbf{z}|\mathbf{x})$, which represents how $\mathbf{x}$ is encoded into the latent space, and a decoder model $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})\, p(\mathbf{z})$. The encoded distribution $q(\mathbf{z}|\mathbf{x})$ is usually Gaussian with a diagonal covariance matrix, i.e. $q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \mu, \sigma^2)$ where μ and σ^2 are the outputs of an encoder neural network. The latent prior $p(\mathbf{z})$ is usually set to $\mathcal{N}(\mathbf{z}; 0, I)$, while $p(\mathbf{x}|\mathbf{z})$ 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 β controls the tradeoff between latent regularization and reconstruction accuracy [15].

The first term from (1) can be considered as a regularization term, because it forces $q(\mathbf{z}|\mathbf{x})$ to remain close to a multivariate standard normal distribution. This prevents $\mathbf{x}$ inputs from being encoded as distributions with disjoint supports, such that the latent space should be continuous [14] i.e. similar inputs should correspond to similar encoded distributions. Moreover, if $\mathbf{x}$ has a much higher dimensionality than $\mathbf{z}$, then the latter can be considered as a compressed and meaningful representation of $\mathbf{x}$.

2.4. Interpolation using Auto-Encoders

Auto-encoder models, e.g. VAEs [15] or adversarial auto-encoders [16], can be trained to improve the interpolation between data points. However, such works are based on learned generators, in contrast to ours which focuses on learning how to interpolate presets for an external sound generator. Some previous works [2, 4] about VAE-based sound matching have stated that they could perform interpolations, as any regularized VAE model does. However, the interpolation itself was not studied, and no quantified results were presented.

3. PRESET INTERPOLATION

3.1. Synthesizer and Datasets

Focusing on DX7 FM synthesis, we used a published database of approximately 30k presets [4], and randomly split it into a 80%/10% training/validation set and a 10% held-out test set. The main volume, transpose and filter controls, which are not part of the FM synthesis process, were set to their default values, and left untouched. Each preset then consists of 144 parameters, including *Algorithm* which controls the discrete routing of signals between oscillators. *Algorithm* alone can completely change the synthesized timbre, such that it is arguably the most important FM synthesis parameter. However, it introduces highly non-linear relationships between presets and output sounds, so the most recent works [3, 12] trained a different model for each *Algorithm* value. In our work, a single model was trained and the dataset includes the *Algorithm* parameter.

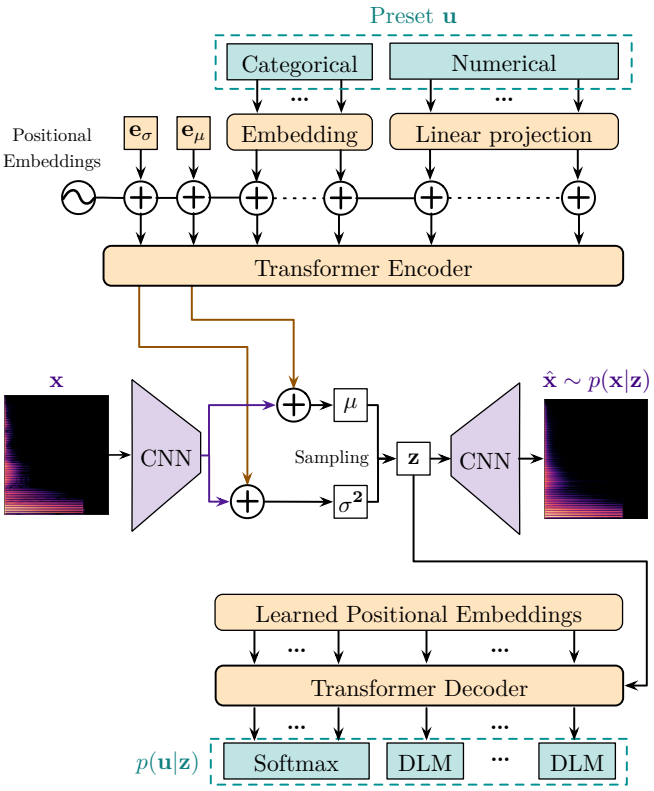

Fig. 1. Overview of the SPINVAE (Synthesizer Preset INterpolation VAE) architecture.

3.2. Model

Thanks to its latent space properties, the VAE framework is well suited to interpolation tasks. For instance, a VAE could be trained to encode presets $\mathbf{u}$ into latent codes $\mathbf{z}$, and to decode them. Nonetheless, it is not certain that the latent space would be continuous in the perceptual audio domain, i.e. that similar $\mathbf{z}^{(n)}, \mathbf{z}^{(m)}$ latent vectors would be decoded to $\mathbf{u}^{(n)}, \mathbf{u}^{(m)}$ that sound similar to a human. Ideally, any $\mathbf{z}$ should hold meaningful compressed audio information. Therefore, we employed a VAE that encodes and decodes presets $\mathbf{u}$ and spectrograms $\mathbf{x}$ (synthesized using $\mathbf{u}$) simulta-

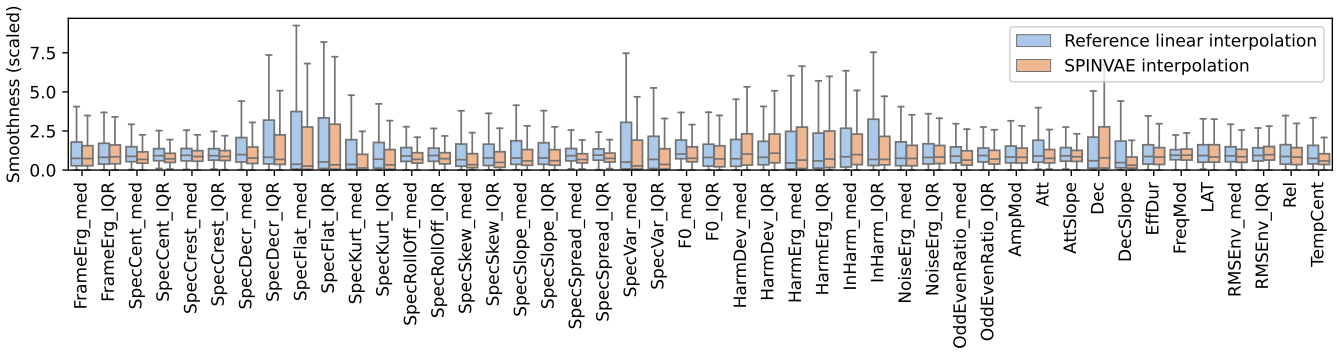

Fig. 2. Interpolation smoothness (lower is better) for each audio feature extracted by Timbre Toolbox [17]. For features to lie on a similar scale, values have been divided by the mean value of each feature for the reference model.

neously. The VAE loss becomes:

$$\begin{aligned}\mathcal{L}(\mathbf{x},\mathbf{u}) =& \beta D_{KL}\left[q(\mathbf{z}|\mathbf{x},\mathbf{u})\|p(\mathbf{z})\right] \\ &- \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{u})}\left[\log p(\mathbf{x},\mathbf{u}|\mathbf{z})\right]\end{aligned} \tag{2}$$

We model $p(\mathbf{x}|\mathbf{z})$ and $p(\mathbf{u}|\mathbf{z})$ as independent distributions (Fig. 1). The audio decoder $p(\mathbf{x}|\mathbf{z})$ models each spectrogram pixel as a unit-variance Gaussian distribution. The audio encoder and decoder are nine-layer CNNs with residual connections.

The preset decoder $p(\mathbf{u}|\mathbf{z})$ uses appropriate distributions for different synthesis parameters. Categorical parameters (e.g. *Algorithm*, waveform type, etc.) are modeled by applying a softmax function on each output token. Numerical parameters (e.g. frequency, attack time, etc.) are nonetheless discrete. Therefore, we'll optimize the log-likelihood of DLM distributions [8], which were originally proven effective to model pixel values. Such distributions are well suited to discrete numerical data with a limited range, because they compute probabilities using discrete bins, and tend to assign more probability to the lowest and highest bins. Considering the histogram of numerical parameters values, three mixture components seemed appropriate. Moreover, we extended the original DLM implementation to model parameters with different sets of discrete values (e.g. 8, 15, 100 quantized steps).

Among related works, only one [1] modeled presets as sequences, using LSTMs. This was a natural choice because parameter values are highly dependent on others (e.g. *Algorithm*). However, our early tests using LSTMs had demonstrated poor performance and unstable training. Therefore, presets are encoded and decoded using multi-head attention (Transformer, [7]) blocks without masking, i.e. each hidden token can attend to tokens at any position in the sequence. Inspired by [18], learnable input tokens $\mathbf{e}_\mu$ and $\mathbf{e}_\sigma$ are concatenated to the preset embeddings' sequence. These two extra tokens are processed by the Transformer encoder, and the corresponding outputs are added to the CNN outputs. On the decoder side, $\mathbf{z}$ is used to compute keys and values, while some learned input embeddings are used to compute queries.

The Transformer encoder and decoder are made of six layers each, and the latent dimension and Transformer hidden size have been empirically set to 256. Implementation and training details are available in our source code repository[1].

3.3. Audio Interpolation Metrics

After training and validation, the model has been used to compute 1.5k interpolation sequences between pairs of consecutive samples from the shuffled held-out test dataset (3k samples). First, two samples $(\mathbf{x}^{(n)},\mathbf{u}^{(n)})$ and $(\mathbf{x}^{(m)},\mathbf{u}^{(m)})$ are encoded into $\mathbf{z}^{(n)}=\mu^{(n)}, \mathbf{z}^{(m)}=\mu^{(m)}$. Then, a latent linear interpolation is performed to obtain $\{\mathbf{z}_t, t\in[1,T]\}$ vectors, with $\mathbf{z}_1=\mathbf{z}^{(n)}$ and $\mathbf{z}_T=\mathbf{z}^{(m)}$. Each $\mathbf{z}_t$ is finally decoded into a $\mathbf{u}_t$ preset, programmed into the synthesizer and rendered to audio. Each sequence contains $T=9$ steps.

Evaluating the quality of an interpolation is straightforward for simple artificial objects, e.g. 2D lines whose length and orientation can be easily measured [16]. However, it is harder to define what a "good" audio interpolation is. Thus, this work relies on Timbre Toolbox [17] to extract audio features engineered by experts. They have been computed for all rendered audio files. Timbre features can be grouped into three main categories: temporal (e.g. attack time and slope), spectral (e.g. spread, centroid), and harmonic (e.g. inharmonicity, odd-to-even ratio).

All available features but *Noisiness*, which was almost constant to an inconsistent 1.0 value, have been included in the results (Fig. 2). A logarithm function has been applied to spectral features in Hz, for values to lie on an approximately linear perceptual scale. Timbre Toolbox features are computed inside several time frames (slices), such that multiple values are available for each feature of a given audio file. Following Peeters *et al.* [17], only the median value and Inter-Quartile Range (IQR) are used.

Similar to [16, 18], two metrics have been computed for each interpolation sequence: smoothness and non-linearity.

[1] https://github.com/gwendal-lv/spinvae

The smoothness of a feature is defined as the RMS of the second-order derivative of this feature's values across the sequence. Non-linearity is the RMS distance between measured feature values, and an ideal linear regression from start to end values of the feature.

3.4. Results

The most common preset interpolation method, which is implemented in some commercial synthesizers, consists in performing an independent linear interpolation for each synthesis parameter. This can lead to smooth interpolations, because synthesis controls are usually mapped to a perceptual scale, e.g. a log-scale for frequencies and amplitudes. Thus, this method has been considered as the reference interpolation.

Fig. 2 shows the smoothness of audio features, for interpolations computed by our model and using the reference method. Thirty-five features are significantly smoother (one-sided Wilcoxon signed-rank test, p-value < 0.05), and the average smoothness is improved, i.e. reduced, by 12.6% (Table 1). This table also presents improved nonlinearity results, not displayed in Fig. 2 due to space constraints. The companion website[2] presents examples of interpolations between presets, and also extrapolations beyond test presets.

Table 1. Performance of different interpolation models compared to the reference linear per-parameter interpolation. Number of significantly improved features (out of 46 total) and average feature value variation (lower is better) for the Smoothness and Nonlinearity metrics. More extensive results and details about audio features are available on the companion website[2].

Model	**# improved features**		**Average variation (%)**	
	Smooth.	**Nonlin.**	**Smooth.**	**Nonlin.**
SPINVAE	**35**	38	$\mathbf{-12.6}$	-12.3
Preset-only	25	30	-4.6	-6.4
Sound match.	8	7	$+66.8$	$+29.7$
DLM 2	31	37	-8.2	-10.4
DLM 4	30	**40**	-9.2	-14.5
Softmax	23	**40**	-1.2	$\mathbf{-15.6}$
MLP	18	27	$+21.0$	-1.8
LSTM	15	1	$+123$	$+93.5$

4. ABLATION STUDY

Table 1 demonstrates that our interpolation model outperforms other related architectures, and provides insight into its ability to increase performance.

Models from the first section of Table 1 are general architecture variants of SPINVAE. The Preset-only model does not auto-encode $\mathbf{x}$, i.e. it is a Transformer-VAE [18, 19] applied to presets. Compared to the reference, it improves the interpolation but does not perform as well as the bimodal SPINVAE. This indicates that learning audio representations alongside preset representations is well suited to this interpolation task. The Sound matching model uses the same architecture as [2, 4], which can be obtained by setting the preset encoder outputs to zero i.e. $q(\mathbf{z}|\mathbf{x}, \mathbf{u}) = q(\mathbf{z}|\mathbf{x})$. However, we could not use the exact same decoder model as these previous works, because they rely on bijective networks which impose strong constraints on the latent dimension and are expected to model continuous numerical synthesis parameters only. The degraded performance of this Sound matching model probably comes from a discrepancy between the learned audio-only latent representations, and the corresponding decoded presets.

The second section of Table 1 presents results obtained with different probability distributions used to model numerical synthesis parameters. DLM 2 and 4 designate DLMs of two and four components, respectively (instead of three for SPINVAE), while Softmax indicates that discrete numerical values are learned as categories. The latter had been used to improve sound matching performance [3, 4, 6]. The interpolation performance is slightly reduced when using DLM 2 distributions, which are not flexible enough to model the data. DLM 4 and Softmax, nonetheless, improve linearity while degrading smoothness. However, subjective listening tests seemed to indicate that smoothness is more important to interpolation quality than linearity. Therefore, SPINVAE uses the DLM distribution, with three components rather than four.

Models from the last section of Table 1 encode and decode presets using MLP or LSTM networks. They perform poorly, which confirms that Transformer blocks are better suited to handle synthesizer presets.

5. CONCLUSION

The SPINVAE architecture has been introduced to auto-encode synthesizer presets and audio simultaneously, in order to perform interpolation between presets. Sequences of synthesized sounds, obtained from interpolated presets, were evaluated by computing the smoothness and nonlinearity of 46 audio features. The evaluation demonstrated that SPINVAE outperforms related architectures, e.g. generative sound matching models. It is also the first model to encode and decode presets using Transformer blocks, and to apply DLM distributions to presets. An ablation study showed that these two elements helped improve the interpolation.

The proposed model was trained on a complex and non-differentiable FM synthesis process, and can be virtually applied to any commercial synthesizer. It can be integrated into synthesizer plugins for live preset interpolation or sound design. Combined with sound matching, the model could even perform interpolations between any re-synthesized recorded sounds.

[2]https://gwendal-lv.github.io/spinvae/

6. REFERENCES

[1] M. J. Yee-King, L. Fedden, and M. d'Inverno, "Automatic programming of vst sound synthesizers using deep networks and other techniques," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 150–159, 2018.

[2] 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.

[3] Zui Chen, Yansen Jing, Shengcheng Yuan, Yifei Xu, Jian Wu, and Hang Zhao, "Sound2synth: Interpreting sound via fm synthesizer parameters estimation," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 2022.

[4] Gwendal Le Vaillant, Thierry Dutoit, and Sebastien Dekeyser, "Improving synthesizer programming from variational autoencoders latent space," in *24th International Conference on Digital Audio Effects (DAFx)*, 2021.

[5] Christopher Mitcheltree and Hideki Koike, "Serumrnn: Step by step audio vst effect programming," in *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*, 2021.

[6] O. Barkan, D. Tsiris, O. Katz, and N. Koenigstein, "Inversynth: Deep estimation of synthesizer parameter configurations from audio signals," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 12, pp. 2385–2396, 2019.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[8] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma, "PixelCNN++: Improving the pixelCNN with discretized logistic mixture likelihood and other modifications," in *International Conference on Learning Representations*, 2017.

[9] 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 *International Conference on Machine Learning*, 2017, p. 1068–1077.

[10] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "Ddsp: Differentiable digital signal processing," in *International Conference on Learning Representations*, 2020.

[11] Krishna Subramani, Preeti Rao, and Alexandre D'Hooge, "Vapar synth - a variational parametric model for audio synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[12] Franco Caspe, Andrew McPherson, and Mark Sandler, "DDX7: Differentiable FM Synthesis of Musical Instrument Sounds," *Proceedings of the 23rd International Society for Music Information Retrieval Conference*, 2022.

[13] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, 1997.

[14] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *International Conference on Learning Representations*, 2014.

[15] 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," *International Conference on Learning Representations*, 2017.

[16] David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow, "Understanding and improving interpolation in autoencoders via an adversarial regularizer," in *International Conference on Learning Representations*, 2019.

[17] Geoffroy Peeters, Bruno Giordano, Patrick Susini, and Nicolas Misdariis, "The timbre toolbox: Extracting audio descriptors from musical signals," in *The Journal of the Acoustical Society of America*, 2011, vol. 130.

[18] Mathis Petrovich, Michael J. Black, and Gül Varol, "Action-conditioned 3D human motion synthesis with transformer VAE," in *International Conference on Computer Vision (ICCV)*, 2021.

[19] Junyan Jiang, Gus G. Xia, Dave B. Carlton, Chris N. Anderson, and Ryan H. Miyakawa, "Transformer vae: A hierarchical model for structure-aware and interpretable music representation learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.