Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)
Demonstrations Track

FlowSynth: Simplifying Complex Audio Generation Through Explorable Latent Spaces with Normalizing Flows

Philippe Esling[1*], **Naotake Masuda**[2] and **Axel Chemla–Romeu-Santos**[1]
[1]IRCAM - CNRS UMR 9912, Sorbonne Université - 1, place Igor Stravinsky 75004 Paris
[2]University of Tokyo, Tokyo, Japan
{esling, chemla}@ircam.fr, naotakemasuda@g.ecc.u-tokyo.ac.jp

Abstract

Audio synthesizers are pervasive in modern music production. These highly complex audio generation functions provide a unique diversity through their large sets of parameters. However, this feature also can make them extremely hard and obfuscated to use, especially for non-expert users with no formal knowledge on signal processing.

We recently introduced a novel formalization of the problem of synthesizer control as learning an invertible mapping between an audio latent space, extracted from the audio signal, and a target parameter latent space, extracted from the synthesizer's presets, using normalizing flows. In addition to model a continuous representation allowing to ease the intuitive exploration of the synthesizer, it also provides a ground-breaking method for audio-based parameter inference, vocal control and macro-control learning. Here, we discuss the details of integrating these high-level features to develop new interaction schemes between a human user and the generating device: parameters inference from audio, high-level preset visualization and interpolation, that can be used both in off-time and real-time situations. Moreover, we also leverage LeapMotion devices to allow the control of hundreds of parameters simply by moving one hand across space to explore the low-dimensional latent space, creating allowing to both empower and facilitate the user's interaction with the synthesizer.

1 Introduction

Audio synthesizers have become a staple in music production, due to their versatility in sound generation. With the advent of software synthesizers, largely less expensive than hardware equivalents and seamlessly integrated with Digital Audio Workstation (DAW), an unprecedented amount of people are users of audio synthesizers. However, the complexity of synthesis algorithms and their large number of parameters make the use of synthesizers difficult for novices. The effect that a certain change in a parameter has on the synthesized sound is often unclear. Thus, there is great need for methods which provides intuitive controls for synthesizers and facilitates the interaction between users and synthesizers.

Previous researches on audio synthesizers have focused on *parameter inference*, the task of finding the set of parameters of the synthesizer that reproduces a given audio [Garcia, 2002], [Yee-King *et al.*, 2018]. However, such systems take away the interactivity from audio synthesizers and do not lead to discovery of new sounds. Another way to mitigate this limitation is to provide *macro-parameters* that control multiple parameters at once. By designing the mapping between macro-parameters and synthesis parameters so that change in a macro parameter correspond to change in certain characteristic of audio, we can obtain an intuitive control for the synthesizer [Johnson and Gounaropoulos, 2006]. However previous works on learning such controls have failed to account for the non-linear relationships between parameters.

Recently, we proposed a new approach for providing intuitive controls for audio synthesizers through a novel formalization of the synthesizer control problem [Esling *et al.*, 2020]. In this paper, we discuss the applications of these innovative high-level controls that simplify the use of audio synthesizers to anyone. Furthermore, we develop an interactive system where the user can control latent dimensions of the probabilistic model (seen as macro-parameters), through hand gestures captured by a *LeapMotion* device. The use of expressive gestures in music has been explored in previous literature and many systems have been proposed [Arfib *et al.*, 2002]. In such systems, gestures must be mapped to an intuitive control of the output audio in order for the system to fully realize its expressive potential. We have made available the implementation of these systems as a Max4Live plugin [1].

2 Model and Formulation

We formulated the problem of synthesizer control as the task of obtaining an invertible mapping between the *auditory latent space* that represents the manifold of synthesizer output, and the corresponding synthesizer *parameter space* [Esling *et al.*, 2020]. Synthesizers can be thought of a function $\boldsymbol{x} = f(\boldsymbol{v})$ that takes parameters $\boldsymbol{v}$ as input and outputs the

*Contact Author

[1]Source code results and plugins are available on a supporting webpage: https://acids-ircam.github.io/flow_synthesizer/
Video demo: https://www.youtube.com/watch?v=UufQwUitBIw

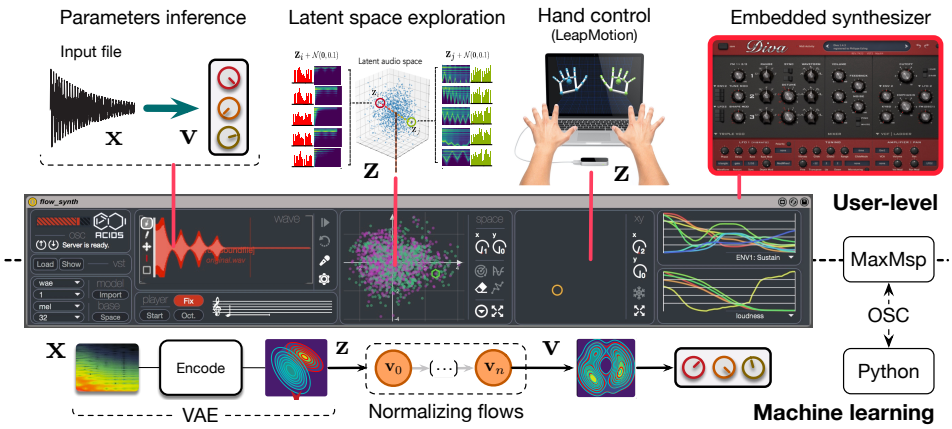

Figure 1: *FlowSynth* interface for high-level audio synthesizer control. The MaxMSP interface features parameters inference (audio to parameters), latent exploration and hand control with LeapMotion[2] through OSC communication with a Python server.

synthesized audio $\boldsymbol{x}$. Instead of directly learning the mapping between $\boldsymbol{x}$ and $\boldsymbol{v}$, we modeled the latent variable of audio $\boldsymbol{z}$ using Variational Auto-Encoders (VAEs) [Kingma and Welling, 2013]. The full generative model

$$p(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{z}) = p(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{z})p(\boldsymbol{v}|\boldsymbol{z})p(\boldsymbol{z})$$

We rely on normalizing flows [Rezende and Mohamed, 2015] to learn the mapping $\boldsymbol{z} = f_{\boldsymbol{\theta}}(\boldsymbol{v})$ between the latent variable $\boldsymbol{z}$ and parameter distribution $\boldsymbol{v}$ leading to the objective

$$\mathcal{D}_{\mathrm{KL}}\left[q(f_{\boldsymbol{\theta}}|\mathbf{z}, \mathbf{v})\|p(f_{\boldsymbol{\theta}}|\boldsymbol{z})\right] = E_q\left[\log q(\boldsymbol{z}_0)\right] - E_q\left[\log p(\boldsymbol{v}_k)\right] - E_q\left[\sum_{i=1}^{k} \log \left|\det \frac{\partial f_i}{\partial \boldsymbol{v}_{i-1}}\right|\right]$$

The explicit modeling of the latent variable $\boldsymbol{z}$ not only improves the accuracy of parameter inference, but also provides a rich representation of sounds that is used for interaction.

To train the model, we constructed a dataset of synthesizer sounds and parameters for *Diva*, but note that our model can work for any synthesizer. As detailed in [Esling *et al.*, 2020], the probabilistic model outperforms baseline models, while providing the huge benefit of explicit semantic macro-controls. In the final interface, we increased the sets of parameters to 128 and observed that our proposal appears as the most resilient compared to baseline approaches for this higher complexity. Importantly, unsupervised dimensions provided by the model allow to effectively obtain the *principal components of audio variations*. This allows to reduce the complexity of a function with hundreds of parameters to only a few dimensions. These dimensions provides intuitive controls, while allowing to go through all possibilities of a synthesizer by simply exploring the low-dimensional latent space.

3 Interface

We implemented all of the interactions described in [Esling *et al.*, 2020] as a `Max4Live` interface and `VST` plugin that is displayed in Figure 1. This interface is split between a `Python` server that handles all of the machine learning implementation and interaction. The model is implemented in `Pytorch` and can run on GPU. However, as the encoding and flow are very light-weight, these can be used in real-time on a basic commercial CPU (inference time is under 20ms on a Intel i5). The `Python` back-end communicates to a `MaxMSP` patch through a `OSC` server, with a custom exchange protocol. This patch wraps the Diva VST as a sub-object, allowing to control its parameter by sending `MIDI CC` commands and generate the audio output. The interface is divided into sub-objects that allow to provide the different high-level controls based on both proposed models.

Parameters inference. The user can input a wave file or direct vocal recording, and the model finds the set of synthesizer parameters that sounds the closest. The audio file is stored on disk and transmitted to the Python server, which transforms and encodes into $\boldsymbol{z}$, which is then mapped to $\boldsymbol{v}$.

Latent space exploration. The latent space $\boldsymbol{z}$ provides the principal components of audio variations, and our invertible mapping allows to display presets for the given synthesizer directly in the latent space (by first performing a 2D PCA). Hence, the preset library is organized based on audio similarity, but users can send any given position $\boldsymbol{z}$ from the latent space to obtain corresponding parameters.

Latent automation. Based on the latent space module previously explained, the user can also draw latent paths between points in the space. This path is beat-synchronized to the music, thanks to the timing transport object of `Max4Live`. Hence, the interpolation of the current position is performed on the client (interface) side, and the full position is sent to the `Python` server in real-time.

Fast discovery with LeapMotion. We implemented a `C++` external allowing to retrieve positions of two hands, by relying on the *LeapMotion* sensor. Hence, we obtain extremely precise coordinates for each joints of the hand and fingers. From these measurements, we compute the centroid of the hand (x, y, z) and its orientations $(roll, pitch, yaw)$, leading to 6 dimensions that can be controlled with each hand. Hence, the user can explore synthesizer sounds in a very efficient manner, by simply interactively exploring the latent space.

References

[Arfib *et al.*, 2002] Daniel Arfib, Jean-Michel Couturier, Loic Kessous, and Vincent Verfaille. Strategies of mapping between gesture data and synthesis model parameters using perceptual spaces. *Organised Sound*, 7(2):127–144, 2002.

[Esling *et al.*, 2020] Philippe Esling, Naotake Masuda, Adrien Bardet, Romeo Despres, and Axel Chemla-Romeu-Santos. Flow synthesizer: Universal audio synthesizer control with normalizing flows. *Applied Sciences*, 10(1):302, 2020.

[Garcia, 2002] Ricardo A Garcia. Automatic design of sound synthesis techniques by means of genetic programming. In *Audio Engineering Society Convention 113*, 2002.

[Johnson and Gounaropoulos, 2006] Colin G Johnson and Alex Gounaropoulos. Timbre Interfaces using Adjectives and Adverbs. In *Proceedings of the 2006 Conference on New Interfaces for Musical Expression*, pages 101–102, 2006.

[Kingma and Welling, 2013] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv:1312.6114*, 2013.

[Rezende and Mohamed, 2015] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)*, 2015.

[Yee-King *et al.*, 2018] Matthew John Yee-King, Leon Fedden, and Mark d'Inverno. Automatic programming of vst sound synthesizers using deep networks and other techniques. *IEEE Transactions on ETCI*, 2(2), 2018.