Neurorack: deep audio learning in hardware synthesizers

DEVIS NINON AND ESLING PHILIPPE, IRCAM - CNRS 9912 STMS - Sorbonne Université, France

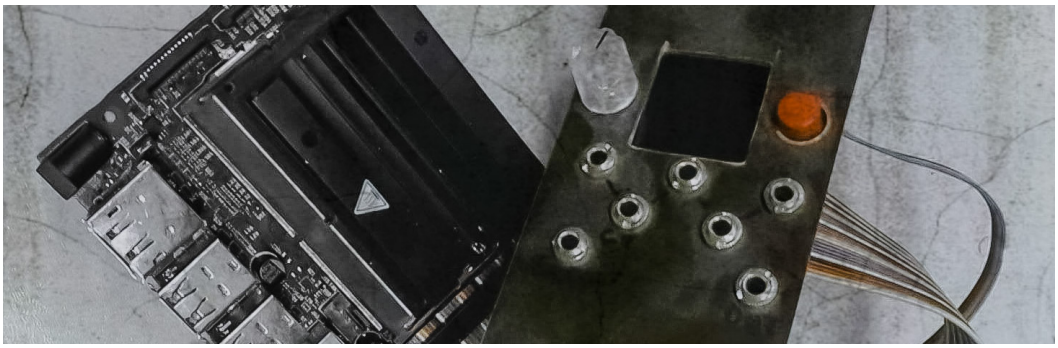

Fig. 1. We introduce the *Neurorack*, a hardware Eurorack module that embeds deep learning

Deep learning models have provided extremely successful methods in most application fields by enabling unprecedented accuracy in various tasks. For audio applications, although the massive complexity of generative models allows to handle complex temporal structures, it often precludes their real-time use on resource-constrained hardware platforms, particularly pervasive in this field. The lack of adequate lightweight models is an impediment to the development of stand-alone instruments based on deep models, entailing a significant limitation for real-life creation by musicians and composers.

Recently, we built the first deep learning-based music instrument by implementing a lightweight generative musical audio model on an adequate hardware platform that can handle its complexity. By embedding this deep model, we provide a controllable and flexible creative hardware interface. More precisely, we focused our work on the Eurorack synthesizers format, which offers *Control Voltage* (CV) and *gate* mechanisms allowing to interact with other classical Eurorack modules.

Additional Key Words and Phrases: Creative machine learning, hardware audio synthesizer, deep learning

1 INTRODUCTION

Over the past decade, spectacular results have been attained using deep machine learning models [8]. More specifically, in the audio domain, recent deep generative architectures [7] are able to accomplish surprisingly realistic audio synthesis based on mimicking low-level acoustic qualities of example signals. Yet, music waveform generation requires to handle very complex and high-dimensional structures. Therefore the synthesis quality often comes at the price of extremely complex models endowed with a humongous number of parameters. This implies the need for specific hardware to train and run those models (such as Graphical Processing Units (GPUs)), along with significant energetic and computational costs. However, the model accuracy is often much more valued than the underlying complexity and remains the sole goal to attain [2].

However, the complexity of those models have numerous drawbacks. First, the extensive inference time hinders real-time usage of the models. Second, the large memory footprint and important disk size both considerably refrain the possibility of embedding those models on a constrained architecture [6]. Those properties become paramount when developing innovative musical instruments as these devices have the same constraints as previous electronic instruments: real-time audio generation on a dedicated hardware with strong memory and computational constraints.

Author's address: Devis Ninon and Esling Philippe, IRCAM - CNRS 9912 STMS - Sorbonne Université, 1, Place Igor Stravinsky, Paris, France, 94200, ninon.devis@ircam.fr.

In this paper, we address these issues by focusing our work on the development of a lightweight deep learning-based musical instrument. The general approach focuses on the generation of audio timbre rather than symbolic notes. Hence, this stand-alone synthesizer is focused on the real-time synthesis and high-level control of a particular given particular timbre. As a case study, we decided to focus on *impact sounds*, as they are notoriously hard to create with traditional audio synthesis methods. We chose to spotlight the versatility of our architecture by adopting the Eurorack format, which simplifies the communication with other classical synthesis models through 3.5mm mono jacks transmitting *control voltages*. Due to their small sizes (around 133mm vertically), Eurorack modules are highly constrained and software-based modules usually feature an Arduino or a Raspberry Pi. Even with such drastic restrictions, we succeeded in building the first deep learning-based synthesizer by targeting an extremely lightweight deep model and performing large software optimizations, along with the relevant choice of a Jetson Nano[1] as the hardware platform.

2 STATE OF THE ART

2.1 Generative models

Generative models [1] are machine learning algorithms able to produce new data instances based on the observation of existing examples. It aims at understanding the underlying structure of data allowing to generate new data with properties similar to the original ones [10]. Hence, these models rely on a set of example data $\{\mathbf{x}_i\}_{i\in[1,n]}$ defined in a high-dimensional space $\mathbf{x} \in \mathcal{X}$, with usually $\mathcal{X} = \mathbb{R}^{d_x}$. As these examples follow an unknown probability distribution $p(\mathbf{x})$, generative models will aim to find this distribution, in order to generate new examples.

In the case of audio signals, the examples $\mathbf{x} \in \mathcal{X}$ can be depicted using several representation, with each having their own specificities. Generative models based on spectral representations often rely on either Variational Auto-Encoders (VAE) [5] or Generative Adversarial Networks (GAN) [3], but generally suffer from the lack of phase information, which is discarded to work on real numbers. Hence, these must rely on approximate phase reconstruction algorithms. Targeting raw audio waveform overcomes this limitation, but also greatly increases the complexity of the data. We can define an audio waveform as $\mathbf{x} = \{x_1, \ldots, x_T\}$, with $p(\mathbf{x})$ being seen as a product of conditional distribution, where each sample is only dependant from the previous one

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t \mid x_1, \ldots, x_{t-1}). \tag{1}$$

In order to reduce the computational cost, some models exploit the differentiability of the Short-time Fourier transform (STFT), in order to define a *spectral loss* rather than a sample-wise difference. Hence, given a spectrogram $S_w(\mathbf{x}) = |\mathrm{STFT}_w[\mathbf{x}]|^2$ computed from our signal $\mathbf{x}$ with a window w, we can evalute the quality of the waveform $\hat{\mathbf{x}}$ produced by the model with parameters θ, through a multi-scale learning loss defined as

$$\underset{\theta}{\operatorname{argmin}} \sum_i \left\| \log\left(S_{w_i}(\mathbf{x}) + \epsilon\right), \log\left(S_{w_i}(\hat{\mathbf{x}}) + \epsilon\right) \right\|_1 \tag{2}$$

Although most audio generative models rely on generic operations, such as convolutions, a recent trend is to mimic simple interpretable Digital Signal Processing (DSP) elements in order to generate complex realistic waveform [4]. In this direction, the Neural Source-Filter (NSF) [11] model simulates the traditional source-filter models by splitting the generation between separate harmonic and noise sources, which are filtered with trainable modules. This approach allows fast waveform generation and is conditioned on high-level descriptors, providing expressive synthesis.

[1] https://developer.nvidia.com/embedded/jetson-nano-developer-kit

3 DEVICE DEVELOPMENT

In this section, we introduce the *Neurorack*, our deep learning-based instrument, which is divided between the hardware module (responsive to CV inputs sent from the synthesizer) and the Jetson nano handling the software computation. We depict in Figure 2 the overall structure of the module and the relations between the hardware and software (green) components.

Hence, we divide our presentation into two main parts: the *software development* and the *hardware prototype*. First, we address the issues related to audio generation, through the model selection and the desired interactions with it. Then, for the hardware part, we discuss our choices of relevant components to build the prototype and define the organization of the face plate panel in order to provide an interface ensuring a flexible and controllable creative process for users.

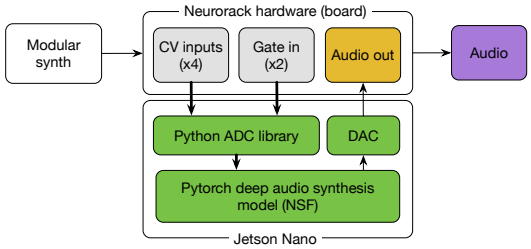

Fig. 2. Overall architecture of the *Neurorack*.

3.1 Software part

3.1.1 *Model Selection.*

Generation. First, our main concern was to ensure the true musical interest of exploiting deep learning to synthesize sounds that would be extremely hard to generate otherwise. Hence, we focused our attention on the synthesis of *impacts*, widely used within cinematic and electronic music. The specificity of impacts is that they are extremely rich in frequencies and particularly diverse. Consequently, excluding the usage of recorded samples, they are almost impossible to synthesize using classical signal processing. Hence, the benefit of deep synthesis mainly resides in the possibility of easily tweaking the generated impacts from a high-level perspective.

Architecture. The second step of our work consisted in selecting a model the could strike the right balance between complexity and audio generation quality. We selected Sinc-NSF [11], a variant of NSF based on SincNet [9], as it gave us the best results while being quite energy-efficient. Similar to NSF, it relies on sinusoidal and noise sources, fed into separate filter modules, allowing to model different types of signals. More precisely, the Gaussian noise passes through a multi-band decomposition and SincNet filters and a band-pass filter. The harmonic source is processed by neural filter blocks and a low-pass filter. The output from the band-pass and the low-pass are finally added and compared to the input with a spectral loss.

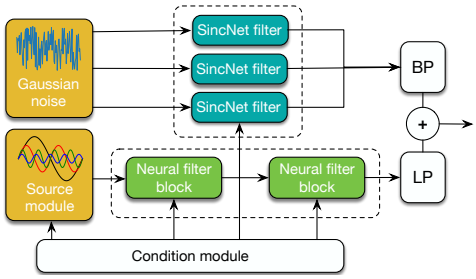

Fig. 3. Sinc-NSF architecture

Dataset. Our dataset has been created by collecting 3500 high quality samples of impacts, in the WAV format with a sampling frequency of 44100 Hz.

Training. We trained the model with the ADAM optimizer, an initial learning rate of 0.0001 , Xavier initialization of the weights and a scheduler that halves the learning rate if the validation

loss stalls for 20 epochs. The overall training takes only 5 hours on 1 GPU NVIDIA Titan V using the whole dataset of 3500 examples of impacts.

3.1.2 Model Operations.

Descriptors. The sound is generated based on the distribution of 7 descriptors: Root mean square, Zero-crossing rate, Spectral roll-off, Spectral flatness, Spectral bandwidth, Spectral centroid and Fundamental frequence. All of these descriptors can be associated to perceptual features, as depicted in Table 1.

Descriptors	Features
Root mean square	Loudness
Zero-crossing rate	Percussivity
Spectral roll-off	Noisiness
Spectral flatness	Ton-like sound
Spectral bandwidth	Richness
Spectral centroid	Brightness
Fundamental frequency	Pitch

Table 1. Relationships between audio descriptors and perceptual features

Generation. First, we extracted from the dataset all the descriptors listed above and train our model to generate the corresponding sounds based on this set. After the training is completed, the model is then able to generate coherent unheard impacts from unseen distributions of descriptors. Hence, generation can be particularly interesting for our instrument as it provides a controllable generative process based on high-level descriptors. Therefore, it is possible to craft the desired tone of the impact through direct manipulation of perceptual features.

3.2 Hardware part

Module. The interactive face panel of our instrument offers a graphical user interface displayed on a 1.3-inch OLED screen. An encoder button integrating an RGB LED alongside with a button for handling the menus all communicate with specific Python libraries. The module also features four CV and two Gates that are compatible with the Eurorack format. A mono audio out is available to output the generated sound. The first CV controls interpolation between different points of the latent descriptor space, and the three remaining CVs directly control high-level descriptors (loudness, brightness and inharmonicity). The first gate, is used to output the corresponding current impact while the second gate triggers the start of the interpolation from the CVs.

Jetson Nano. Our module front-end is connected to a Jetson Nano, which handles all software computation. This NVIDIA platform is a mini-computer in SO-DIMM format offering a 128-core GPU alongside with 4 CPUs. Using SBC, it allows the use of libraries taking advantage of the GPU, which does not exist on a Raspberry Pi.

4 PERSPECTIVE AND CONCLUSION

In this paper we presented the *Neurorack*, the first deep learning-based synthesizer, developed following the Eurorack format. It generates impacts using the Sinc-NSF model which takes as input seven high-level descriptors. The generated sounds can be modified through these descriptors, by sending CV to our instrument, which allows to shape the tone of the impact. In the near future, we hope to embed more deep models and provide a whole variety of high-level interactions.

ACKNOWLEDGMENTS

This research has been funded by SCAI, the Emergence(s) ACIMO project of Sorbonne Université and the ACIDITEAM project of Ville de Paris.

REFERENCES

[1] Christopher M Bishop. 2006. *Pattern recognition and machine learning*. springer.

[2] Constance Douwes, Philippe Esling, and Jean-Pierre Briot. 2021. Energy Consumption of Deep Generative Audio Models. *arXiv preprint arXiv:2107.02621* (2021).

[3] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. 2019. Gansynth: Adversarial neural audio synthesis. *arXiv preprint arXiv:1902.08710* (2019).

[4] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. 2020. DDSP: Differentiable digital signal processing. *arXiv preprint arXiv:2001.04643* (2020).

[5] Philippe Esling, Adrien Bitton, et al. 2018. Generative timbre spaces: regularizing variational auto-encoders with perceptual metrics. *arXiv preprint arXiv:1805.08501* (2018).

[6] Philippe Esling, Ninon Devis, Adrien Bitton, Antoine Caillon, Constance Douwes, et al. 2020. Diet deep generative audio models with structured lottery. *arXiv preprint arXiv:2007.16170* (2020).

[7] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. 2016. SampleRNN: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837* (2016).

[8] Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-Yiin Chang, and Tara Sainath. 2019. Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing* 13, 2 (2019), 206–219.

[9] Mirco Ravanelli and Yoshua Bengio. 2018. Speaker recognition from raw waveform with sincnet. In *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 1021–1028.

[10] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082* (2014).

[11] Xin Wang and Junichi Yamagishi. 2019. Neural harmonic-plus-noise waveform model with trainable maximum voice frequency for text-to-speech synthesis. *arXiv preprint arXiv:1908.10256* (2019).