Towards the Use of Generative Machine Learning for Synthesizer Patch Creation

Matthew Peachey
Graphics and Experiential Media Lab
Faculty of Computer Science, Dalhousie University
Halifax, Canada
peacheym@dal.ca

Joseph Malloch
Graphics and Experiential Media Lab
Faculty of Computer Science, Dalhousie University
Halifax, Canada
jmalloch@dal.ca

***Abstract*—Synthesizers are a unique type of musical instrument that are capable of generating sounds in a very wide range of timbres. Musicians will typically save the state of their synthesizers into "patches" when they discover sounds that they would like to share or recall at a later time. This paper demonstrates a work in progress for utilizing unsupervised Machine Learning models such as Generative Adversarial Networks and Variational Autoencoders can be used to generate completely new Synthesizer patches. The results achieved in this work are promising and suggest that applying these modern Machine Learning models to this specific problem is both feasible and extendable.**

I. Introduction

Synthesizers are powerful tools that allow musicians to create a wide variety of sounds by tweaking adjustable parameters. A typical workflow when making music with synthesizers is to adjust several parameters until a desired timbre (the quality of a sound [6]) is identified and save that state of the instrument for later use. Machine Learning (ML) techniques have been applied to synthesizers via projects such as Wekinator [3] that enable users to use supervised ML for creating mappings between control surfaces and synthesizers. Furthermore, projects such as WaveNet and GanSynth [7, 2] use ML to generate audio directly at the waveform level. This paper presents a work in progress for the use of generative ML models, namely Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) for generating and interpolating between synthesizer patches.

The rest of this paper is structured as follows. A background on digital synthesizers is presented along with the reasons for selecting a specific synthesizer to use in this project. Next, an overview of both GANs and VAEs is presented followed by the specific methods used for our experiments. Finally, a discussion of the results and future work for this project is presented.

II. Digital Synthesis

A. Basic Components of Synthesis

Both analog and digital synthesizers are typically comprised of the same fundamental building blocks. In subtractive synthesis, which this paper primarily focuses on, oscillators are used to generate signals in specific waveforms, filters are used to remove certain frequencies from a signal, ADSR envelopes are used to change signals over time (whether it be related to amplitude, frequency or pitch) and other synthesizer specific components are used to generate unique and interesting timbres.

B. Synthesizer Patches

Synthesizer patches are simply snapshots of the synthesizer's state at a specific time. Many digital synthesizers allow users to save patches in order to recall them at a later date or share them with other musicians. Patches will typically aim to capture a specific sound (whether it be a familiar one or not) and for convenience sake will typically be named after the sound or emotion that the patch aims to capture.

C. amSynth

There are countless synthesizers in both the analog and digital spaces. These range from commercial hardware products such as the Roland Juno-60[1] to software plugins for digital audio workstations such as Ableton Live[2].

AmSynth[3] is an open-source subtractive synthesizer with a list of features including dual oscillators, resonant filters, preset bank and patch management, MIDI connectivity and more.

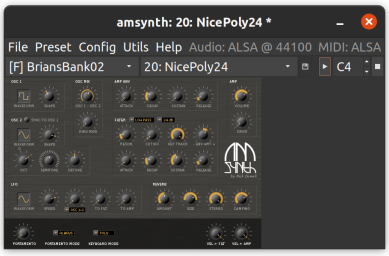

Fig. 1: Screenshot of amSynth's user interface

This synthesizer was selected as the subject of the experiments presented in this paper due to its ease of use, lack of clutter, and large collection (approximately 3000) of existing patches. The latter was crucial as working with ML requires a

[1] https://www.roland.com/ca/products/rc_juno-60/
[2] https://www.ableton.com/en/shop/live/
[3] https://amsynth.github.io/

sufficient amount of sample data and thus having an existing set of patches removed the need to externally source these ourselves.

III. Generative Machine Learning

Generative ML is a subset of unsupervised ML that aims to discover distributions in a sample dataset and therefore be able to generate new samples that could plausibly have come from the original dataset. This generation of new examples is what separates generative ML models from other unsupervised methods that focus on clustering and other traditional unsupervised tasks. The following sections give a brief overview of two such generative models used in this project.

A. Generative Adversarial Networks

Generative Adversarial Networks (GANs) were first presented by Ian Goodfellow in 2014 as a method for generating new data based on sample datasets. GANs work by creating two complementary neural networks, a generator and a discriminator [4]. In Figure 2 these two neural networks are coloured yellow and purple respectively. These two neural networks are trained as opponents attempting to fool one another. More specifically, the generator attempts to create a datapoint that could have plausibly come from the original dataset and the discriminator attempts to determine whether or not a datapoint is fake (generated) or real (belongs to the original dataset). When both the generator and discriminator are trained to their full potential, the generator is capable of creating realistic data while the discriminator is forced to simply guess whether or not a datapoint is real.

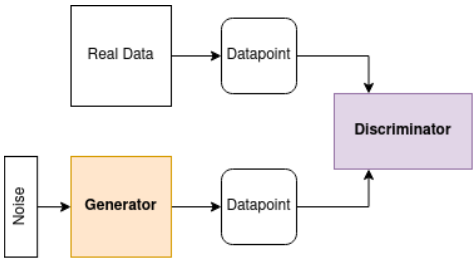

Fig. 2: A basic GAN Architecture.

B. Variational Autoencoders

Variational Autoencoders (VAEs) are another type of generative ML model. Rather than training opposing neural networks, VAEs aim to reduce the original feature space of a dataset into a latent space, typically denoted as z, as shown in Figure 3. This is done by using two inverted neural networks to encode the original data into the latent space with the goal of achieving a perfect reconstruction after decoding the data from the same latent space. Once the encoder and decoder have both been trained, a user is able to run inference on the decoder with new latent vectors in order to generate previously unseen data.

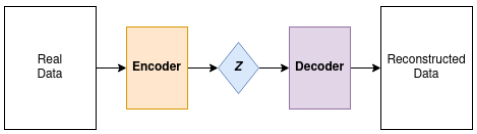

Fig. 3: A standard VAE Architecture.

IV. Methods

We conducted two experiments that used different generative ML models to create new synthesizer patches. Using both GAN and VAE type architectures, we were able to produce interesting timbral results. The methods used to achieve this are discussed in the following sections.

A. Data Processing

In order to make use of the presets included with amSynth, we were required to parse each of the raw text files. Furthermore, because some of the synthesizer parameters were selectable values (rather than continuous ones) we also had to convert these into one-hot encoded vectors in order for our neural networks to train most effectively on these parameters [5]. Finally, we also removed columns from the data-frame that were associated to non-timbrally interesting parameters such as "*master volume*" and "*keyboard tracking*". The final result was a dataset with 2586 datapoints, each of which is a vector of length 82.

B. Model Architectures & Hyperparameters

The following sections discuss the specifics of the model architectures and hyperparameters used in this project.

1) GAN Experiment: For the GAN trial, we utilized a variation of vanilla GAN called Wasserstein GAN (W-GAN) due to the additional training stability and meaningful loss function that the architecture provides [1]. We created a generator network that utilized a latent dimension of 32, three hidden layers of the following dimensions:

- $(128_{in}, 256_{out})$
- $(256_{in}, 512_{out})$
- $(512_{in}, 1024_{out})$

and an output layer with a dimension of 82 as expected for the preset size. The discriminator followed a similar architecture, with an input layer that accepted 82 inputs, three hidden layers, and an output layer with a single output node that reports whether or not the network believes a datapoint to be real or fake. This experiment used the RMSprop optimizer with a learning rate of 0.00005 for 300 epochs with a batch size of 64.

2) VAE Experiment: For the VAE trial, we utilized a minimal VAE with two hidden layers and a latent space with size $z = 16$. The layers used in the encoder network were $(16_{in}, 32_{out})$ and $(32_{in}, 82_{out})$ respectively. For the decoder network, the same layers were used in reverse order. This experiment used the Adam optimizer and was trained for 50 epochs with a batch size of 32.

V. Results

A. Model Statistics

Figure 4 and Figure 5 show the loss curves for the GAN and VAE architectures respectively. Note that the GAN's generator and discriminator losses do tend towards convergence after approximately 6000 training examples. Similarly, the loss associated with the VAE reaches an near minimum after training on approximately 1700 examples.

Fig. 4: W-GAN Loss

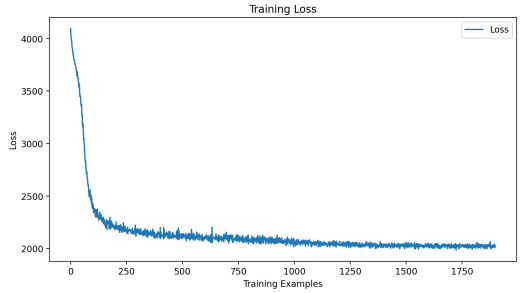

Fig. 5: VAE Loss

B. Discovered Timbres

While the loss and accuracy of ML models are important metrics for success, they do not tell the full story when reconstructing synthesizer patches based on inference of the models. In order to qualitatively asses the results, the patches must be played through the synthesizer and be evaluated timbrally. We discovered that while certain patches did results in timbres that were musically pleasing, many of the patches did not. Some of the main issues were that attack times and LFO amounts (two important factors for timbral qualities [8]) were higher than would typically be expected, resulting in sounds that were saturated with a tremolo effect (in this case where the sound oscillated between high volumes and low volumes resulting in an unsatisfactory sound).

C. Source Code & Audio Samples

The full source code associated with this paper is available in this GitHub repository which allows the models to be trained and utilized. Audio samples of the generated patches are also available at that link.

VI. Future Work

This paper presents experimental results for using VAE and GAN architectures for generating new synthesizer patches. Future work for this project includes refining the network architectures and hyperparameters for the presented methods in hopes of achieving better timbral results.

A. Synthesizer Selections

An interesting future work will be to see how well these techniques apply to other synthesizers, whether they be subtractive like the one used in this paper, or follow other paradigms such as granular synthesis, additive synthesis, etc.

Additionally, exploring synthesizers that are both much simpler in parameter space as well as much more complex with hundreds of tweakable parameters are both interesting research questions to be asked in the future.

B. Dataset Filtering

We know that machine learning models are only as good as the data upon which they are trained. As mentioned in the discussion about timbral qualities of the generated preset, the patches were very biased to high attack times. One potential cause for this is the dataset having patches meant to simulate "helicopter" tones with long attack times (up to the maximum of 15 seconds). By restricting the initial dataset to not include any presets with attack times over 1 second, we suggest that better results may be achieved when training generative models. This notion may also apply to other parameters that approach their maximum values, suggesting the need for additional research.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. DOI: 10.48550/ARXIV.1701.07875.

[2] Jesse Engel et al. "Gansynth: Adversarial neural audio synthesis". In: *arXiv preprint arXiv:1902.08710* (2019).

[3] Rebecca Fiebrink, Daniel Trueman, Perry R Cook, et al. "A meta-instrument for interactive, on-the-fly machine learning". In: *Proc. NIME*. 2009.

[4] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. DOI: 10.48550/ARXIV.1406.2661.

[5] Kedar Potdar, Taher S Pardawala, and Chinmay D Pai. "A comparative study of categorical variable encoding techniques for neural network classifiers". In: *International journal of computer applications* 175.4 (2017), pp. 7–9.

[6] Jean-Claude Risset and David L. Wessel. "Exploration of Timbre by Analysis and Synthesis". In: *The Psychology of Music (Second Edition)*. Ed. by Diana Deutsch. Second Edition. Cognition and Perception. San Diego: Academic Press, 1999, pp. 113–169. ISBN: 978-0-12-213564-4.

[7] Aäron Van Den Oord et al. "WaveNet: A generative model for raw audio." In: *SSW* 125 (2016), p. 2.

[8] Bin Wu, Andrew Horner, and Chung Lee. "Musical timbre and emotion: The identification of salient timbral features in sustained musical instrument tones equalized in attack time and spectral centroid". In: *ICMC*. 2014.