SynthScribe: Deep Multimodal Tools for Synthesizer Sound Retrieval and Exploration

Stephen Brade[1*] **Bryan Wang**[1] **Mauricio Sousa**[1] **Gregory Lee Newsome**[1] **Sageev Oore**[2]
Tovi Grossman[1]
[1]University of Toronto [2]Dalhousie University
stephen.brade@mail.utoronto.ca
{bryanw,mauricio,tovi}@dgp.toronto.edu
greg.newsome@utoronto.ca
sageev@dal.ca

Abstract

Synthesizers are powerful tools that allow musicians to create dynamic and original sounds. Existing commercial interfaces for synthesizers typically require musicians to interact with complex low-level parameters or to manage large libraries of premade sounds. To address these challenges, we implement SynthScribe — a fullstack system that uses multimodal deep learning to let users express their intentions at a much higher level. We implement features which address a number of difficulties, namely 1) searching through existing sounds, 2) creating completely new sounds, 3) making meaningful modifications to a given sound. This is achieved with three main features: a multimodal search engine for a large library of synthesizer sounds; a user centered genetic algorithm by which completely new sounds can be created and selected given the user's preferences; a sound editing support feature which highlights and gives examples for key control parameters with respect to a text or audio based query. The combination of these features creates a novel workflow for musicians exemplifying the usefulness of systems developed with a foundation of multimodal deep learning.

1 Introduction

Synthesizers enable musicians to create rich and complex timbres — unique sounds that can be played on a keyboard or through other digital interfaces that allow composers to expand their timbral vocabulary beyond a typical set of instruments (e.g. guitars, violin, timpani, etc.). To use a synthesizer, musicians must learn to manage hundreds of control parameters that modify the various aspects of a sound. Many musicians typically rely on preset sounds which are designed by a skilled engineer and packaged with a synthesizer as starting points for musicians. Allegedly, 9 out of 10 Yamaha DX7s are returned with their default presets still intact, suggesting that presets are heavily relied on and that many musicians rarely succeed in creating unique sounds of their own(Seago et al. (2004)). This may partly be due to the high dimensionality of synthesizer sounds and their non-intuitive naming conventions. Synthesis parameters are often named after quantifiable aspects of sounds which is disconnected from typical experiential descriptions of sounds(Seago et al. (2004)).

To facilitate their use, synthesizers typically come with a large bank of premade synthesizer sounds (typically called presets) which are sometimes accompanied by a keyword search engine. A good example of this is Analog Lab V by Arturia [1] which provides a bank of 5840 preset sounds where each sound is labeled with various semantic attributes that can be searched by the user. Music technology

[1]https://www.arturia.com/products/software-instruments/analoglab-v/overview

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

companies may also design macrocontrols for their synthesizers — more intuitive controls which combine multiple control parameters into one control with an intuitive name. Machine learning and intelligent interface research has attempted to automatically create intuitive macrocontrols through active learning or by using latent representations of synthesizer sounds to craft macrocontrols automatically(Esling et al. (2020, 2019); Huang et al. (2014)). Machine learning researchers have also developed sound matching algorithms that would let users find the synthesis parameters that best replicate any piece of recorded audio (Horner et al. (1993); Lai et al. (2006); Masuda and Saito (2021)).

In order to help musicians use synthesizers, these prior approaches require a new model to be trained or the collection of user annotations of synthesizer sounds. In this work, we combine several features in a novel system that can help users search for, modify, and create completely new synthesizer sounds by using text and audio concurrently as an intuitive control modality without training a new model or requiring user annotations.

2 SynthScribe

To this end, we present SynthScribe — a full stack system built on top of the Diva synthesizer by u-He that leverages multimodal deep learning to help musicians express their desires for synthesizer sounds at a much higher level. We implement three features in SynthScribe that leverage the multimodal deep learning model LAION-CLAP (Wu* et al. (2023)).

Multimodal Search We record and embed a 3529 preset sounds and implement a multimodal search of synthesizer sounds. Using this feature, musicians can express their desires at a high level using text and then refine their search by running an audio search on a given synthesizer sound which more closely resembles their desires.

Genetic Mixing We also create a user-centered genetic algorithm where users can create hundreds of new synthesizer by mixing together a musicians favourite sounds; these sounds are then recorded and embedded with LAION-CLAP on the fly, allowing users to rapidly discover completely new sounds through the multimodal search feature.

Preset Modification Once a user has found a sound that nearly meets their desires, they can modify it using the preset modification feature which highlights important groups of parameters with respect to a text or audio query and provides examples of how to modify those parameters to achieve a desired effect.

3 Analysis and Conclusion

We complete a comparative user study that examines LAION-CLAP's performance on synthesizer sounds with BERT as a baseline (Devlin et al. (2019)). The results show that LAION-CLAP outperforms BERT on this task, providing a solid foundation for our system. We also solicit feedback from musicians in a 6 participant user study. This study exposes several use cases for our system; specifically, our participants highlight how SynthScribe can save them time while also providing them with exciting new sounds that inspire their creativity. We observe that musicians particularly like sounds which are desirable but unexpected. This suggests that future work that applies multimodal deep learning to synthesizers should not just focus on creating sounds which precisely replicate what a user specifies but should also attempt to find desirable but surprising sounds that go beyond what the user is capable of imagining. To further contextualize this work, we include detailed explanations of our features and experiments in the supplementary materials and request to give a demo exemplifying these new tools. We also compile a short video [2] containing a system walkthrough and performances with sounds created using SynthScribe.

[2] `https://youtu.be/PWPt7ErSKU0?si=udcF5m5rh2SjPCzI`

4 Ethical Considerations

SynthScribe is designed to help musicians but could also be a detriment for working musicians that specialize in making synthesizer sounds. If the tool were to be widely adopted or if SynthScribe inspired an impactful piece of future work, it is possible that the developments of these technologies could make it easier to find interesting synthesizer sounds for less skilled musicians that would have otherwise relied on the services of these specialists. In future work, it could be important to take a utilitarian perspective which weighs whether these tools would provide more benefit to specialized musicians than detriment.

5 Supplementary Material

5.1 Implementation Details of SynthScribe

The whole of SynthScribe is built using a combination of Max — a visual programming language used by musicians to develop bespoke synthesis algorithms with interface development capabilities — and Python. Max is used to host the Diva Synthesizer and to handle notes played on digital instruments which are sent to the Diva Synthesizer to be rendered as an audio signal. Python is used to implement features which require machine learning capabilities and to provide support when a programming task is difficult to implement in Max. These Python functions are executed with inputs from Max by making POST requests to a Flask (Grinberg (2018)) API. All of our ML related features make use of LAION-CLAP embeddings to make a useful connection between text and audio. These embeddings represent the modalities of text and audio in a joint space, thus allowing for retrieval of audio with text while also allowing for direct comparisons to be made between audio samples. We choose a checkpoint [3] which has been trained first on general audio and then finetuned on music, allowing users to describe instruments in addition to other non-musical sounds when searching for presets. Below we describe the functionality of the key components.

5.1.1 The Diva Synthesizer

We purchased the Diva Synthesizer by U-he to create a foundation for our system. We choose this synthesizer as it is relatively popular and has been used for similar tasks in a relevant prior work by (Esling et al. (2020, 2019)). In addition to this, it also has a large amount of preset sounds which are available for free on the internet. In total, we make use of the 1200 factory preset sounds made by U-he and downloaded 2328 user preset sounds created by independent musicians on the internet.

5.1.2 Preset Retrieval

To enable text and audio based preset retrieval, we make use of LAION-CLAP embeddings. For the existing presets, we record and embed them ahead of time using a combination of Max and Python. Each preset is recorded at middle C for 4 seconds with the note being sustained for the first second of that interval. For a text search, a Flask API endpoint handles a POST request from Max containing the text query. This text query is then embedded and the embedded presets are ranked with respect to their cosine similarity to the text embedding. This ranked list is then returned to Max to be visualized. If the user wants to execute an audio search, the embedding for the preset they have selected is retrieved and the presets most similar to the given preset (including the given preset itself) are returned back to the user. The interface for this feature is illustrated in Fig. 1.

5.1.3 Genetic Mixing

Users can make hundreds of new sounds by adding their favourite sounds to a favourites list and then executing the genetic mixing procedure. The mechanisms for genetic mixing can be described using the language of Genetic Algorithms (GA). Initially, the user selects a group of fit presets to be bred together in order to create a new generation of fit presets. To create this new generation of individuals,

[3] $music\_audioset\_epoch\_15\_esc\_90.14.pt$ available at `https://github.com/LAION-AI/CLAP`

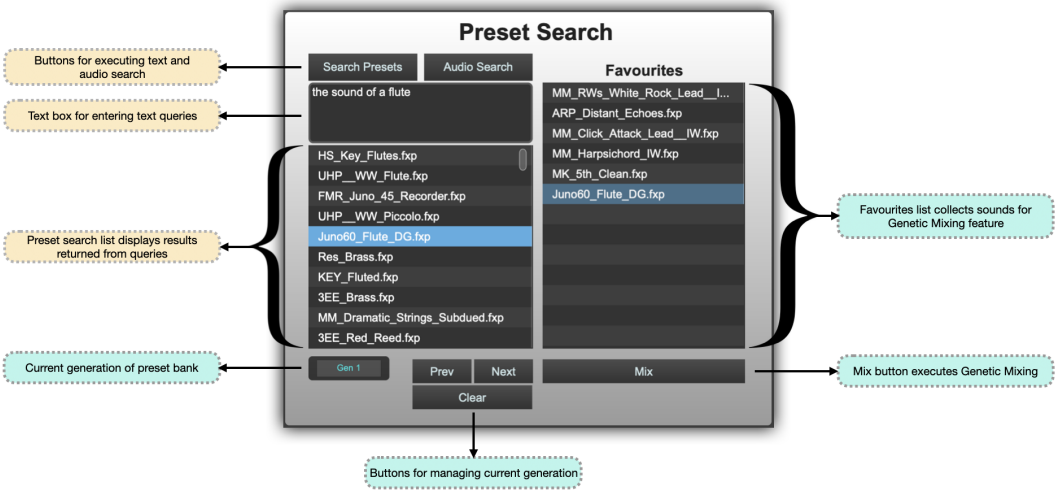

Figure 1: Interface for the window where users can search through the default preset bank (visualized in this figure) or presets generated with the Genetic Mixing feature. Key functionality for the Multimodal Search are highlighted in orange and Genetic Mixing features are highlighted in turquoise.

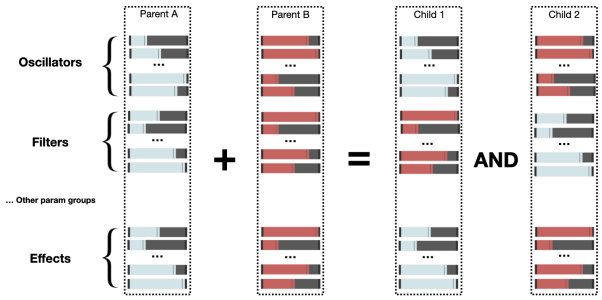

Figure 2: Demonstration of a uniform crossover between two parent synthesizer sounds. The groups of parameters (Oscillators, Filters, ..., Effects) are swapped in their entirety to create the children.

we breed pairs of preset sounds by having a child randomly inherit whole groups of parameters from one parent or the another. The breeding mechanism is displayed in Fig. 2. Diva and many other synthesizers are broken down into panels where each panel contains a group of parameters which control a specific aspect of the sound. For Diva, we recognize 13 such groups. In a breeding operation between parent A and parent B, two children are always created. For a specific group of parameters, the first child will randomly inherit either parent As or parent Bs group with equal probability. The second child will always receive each group that the first child did not inherit. For each pair of parent presets, we complete 5 breeding operations resulting in a total of 10 children per pair. This operation is equivalent to a uniform crossover over the groups of parameters. Upon creating a new generation of presets, they must be recorded and embedded to maintain search functionalities. This is achieved by first using the Python library DawDreamer (Braun (2021)) to record each preset sound in faster than real time and then using LAION-CLAP to embed each of these new recordings. The interface for this process is illustrated in Fig. 1 in turquoise.

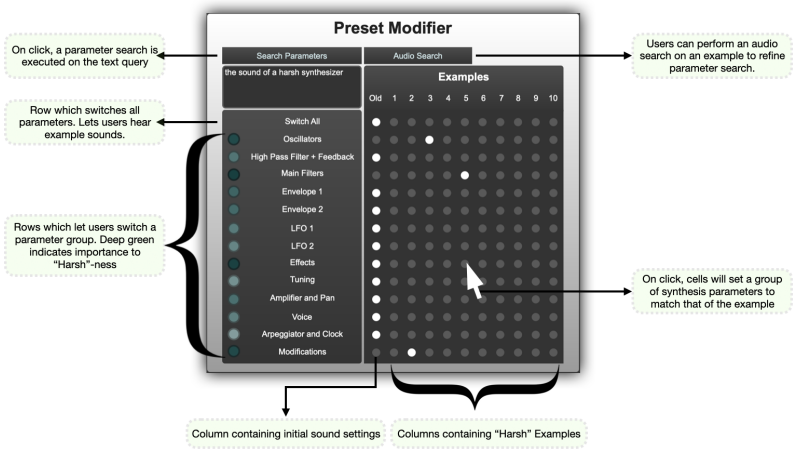

Figure 3: Interface for the Preset Modification feature. Users can retrieve examples that are relevant to a desired effect by first using a text query. Examples relating to this text query are placed in columns of the "Examples" matrix. These examples are labelled numerically and can be refined using an audio search. The "old" column contains the sound the user wishes to modify. They can change this sound by clicking on cells in other columns. Each row corresponds to 10 possible changes for a group of parameters and the importance of these parameters is highlighted using the green LEDs to the left of the parameter group names.

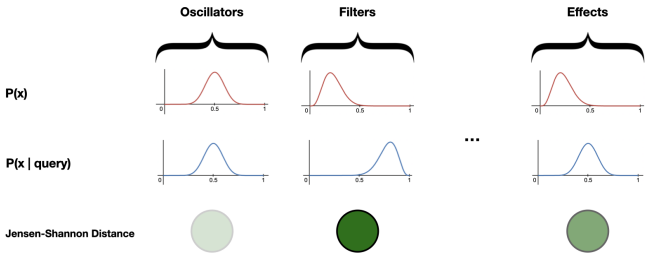

Figure 4: A visualization to show how the importance of each parameter group is calculated. This shows how the shade of green is deepest when the distribution for the parameter is greatly changed as we isolate for the synthesizer sounds returned with respect to some query.

5.1.4 Preset Modification

Users can use a text query to express a modification they would like to make to a given synthesizer sound. We then use LAION-CLAP to retrieve examples that are related to the desired affect that the user is trying to impress onto their current sound. We then let users change the parameters of the original sound by letting them mix and match groups of parameters from the example sounds onto the parameters of their current sound using the example matrix (the interface for this is depicted in Fig. 3). We highlight the importance of the parameters with respect to their desired change by highlighting certain groups of parameters with varying shades of green. We call this feature the parameter group highlighter . It provides the foundation for having the user modify presets because it guides the users exploration of the example matrix. We implement this by treating each parameter on the synthesizer as a random variable and observing differences in the distributions of a parameter over the whole preset bank against an approximate distribution of that parameter for the top 100 presets returned for the user's query. If there is a large difference in these distributions, it shows that this parameter is

in part responsible for the perceptual qualities in the sounds returned for this query. For example, if the user searches for "the sound of an arpeggio" it can be expected that the parameters in the top 100 presets for the Arpeggiator will more often be turned on and modified in a unique configuration than those returned outside of the top 100. To quantify this difference, we first approximate the distributions of each parameter on the synthesizer using all 3528 presets. Discrete parameters are left as is while continuous parameters (always valued between 0 and 1) are approximated with 10 equidistant bins. Empty bins are handled with additive smoothing. Most continuous variables have a default value which are large spikes in probability density at the extremes or precisely in the middle of a parameters range. To better represent this reality, we give default values their own narrow bin. At inference time, we take in a text or audio query from the user and find the top 100 presets using cosine similarity of LAION-CLAP embeddings. We then recalculate the distributions for each parameter over these 100 samples using the same bins and additive smoothing as before. These distributions are then compared using Jensen-Shannon Distance (Lin (1991)) — a stable and symmetric distance measure between distributions which is ranged between 0 and 1. We proceed to calculate the average distance for the top 20 parameters with the largest distance in each group. We select the top 20 so that the importance of large parameter groups are not diluted by the fact that many of their parameters will be left at default settings. Upon calculating this average for each group, we assign the parameter group with the largest average distance the deepest shade of green and interpolate the colors for other groups on a range between 0 and the maximum. We choose this colour interpolation strategy to ensure that maximum is always obvious.

5.2 User Evaluation of LAION-CLAP

We evaluate LAION-CLAP's ability to retrieve synthesizer sounds and use BERT (Devlin et al. (2019)) as a strong baseline. We do so by running a user study where we have users rate the sounds returned using LAION-CLAP and BERT with respect to a text query.

5.2.1 Participants

We recruited 8 participants in total for the study (Mean age=25.0, STD=2.9). Participants were not required to have any level of musical experience. Only two participants actively played an instrument and none had extensive experience using synthesizers. The study took place across two independent tasks which happened on different days where each task lasted approximately 30 minutes. Participants were compensated with a total of 20 CAD.

5.2.2 Methodology

Participants are tasked with rating the relevance of a set of synthesizer sounds with respect to a text query on a 7-point Likert scale. In total, each participant is given two text queries and evaluates a total of 10 sounds for each text query. The text queries are generated before each study in a quasi-random fashion using an adjective and an instrument class to describe a sound. An example query could be "The sound of a harsh brass instrument". The list of possible adjectives is compiled from relevant research in music psychology and keywords used in keyword searches for synthesizers. The list of possible instrument classes is sourced only from keywords used in keyword searches for synthesizers (Both lists are available in Table 1). To make these queries understandable to our participant class, we avoid instrument classes and adjectives which would be non-intuitive to people without musical experience. For each query, we retrieve 5 sounds using LAION-CLAP and 5 sounds using BERT from the Diva Synthesizer preset bank. LAION-CLAP retrieval is achieved using the strategy described in subsection**??**. BERT retrieval is achieved by extracting adjectives and instrument classes for each sound in the Diva Synthesizer preset bank using GUI automation and OCR. Given these adjectives and instrument classes, we embed sentences of the form "The sound of a adjective instrument class" using BERT. When querying both models, we use the form "The sound of..." to better fit with the training data for LAION-CLAP; however, since we use it to create out BERT embeddings as well this gives LAION-CLAP no undue advantage for the task. The five sounds are then compiled into a list of 10 in a random ordering. The participant is then allowed to play each sound on a musical keyboard and provides their evaluation.

Table 1: Adjectives and Instruments

Instrument Class	Wind Instrument, Electric Piano, Bass, Drum, Brass Instrument, String Instrument, Organ
Adjectives	Percussive, Constant, Moving, Clean, Dirty, Soft, Aggressive, Thin, Complex, Funky, Sharp, Simple, Punchy, Huge, Bizarre, Mellow, Atmospheric, Airy, Evolving, Short, Long, Noisy, Glitchy, Arpeggiated, Distorted, Acoustic, Dull, Loud, Low, Rough, Smooth, Clear, Rich, Nasal, Full, Hard, Weak, Muffled, Resonant, Large, Quiet, Calm, Harsh, Shrill, Powerful, Metallic, Ringing, Deep

5.3 Results

As shown in Table 2, LAION-CLAP outperforms BERT with a higher average and median rating. Further analysis with a Wilcoxon Signed-Rank test shows that this result is statistically significant ($p < .05$).

Table 2: LAION-CLAP and BERT Ratings

Model	Performance		
	Mean	Median	IQR
LAION-CLAP	4.45	4.5	3.0
BERT	3.88	4.0	2.0

5.4 Free Usage Observation with Musicians

5.4.1 Participants

Our study consisted of 6 musicians who had experience creating music. Two were professionals (P2, P5) and the rest were hobbyists. Both of the professional musicians had received formal instruction on synthesizers in an academic setting with P2 describing themselves as an expert and P5 an intermediate. The hobbyists had at most encountered synthesizers when making music and described themselves as either novices or beginners. All musicians had at least basic keyboard skills. Each were compensated with 20 CAD.

5.4.2 Methodology

The study consisted of a 15-minute demo of SynthScribe followed by the participant completing at least two independent musical tasks. The participant was provided with a 61-key keyboard that they used to play the synthesizer sounds. P2 brought in and was permitted to use their Akai EWI 5000 [4] — a digital wind instrument. The participant was free to select any task that they might encounter while using a synthesizer for their own purposes. Suggested tasks included finding a desirable synthesizer sound for a song they knew how to play, replicating a synthesizer sound from a song, or finding a synthesizer sound that might fit in a score for a film of a particular genre. When using the system, we requested that the participant write text queries in the form of "The sound of a ..." to ensure the best results due to the nature of the training data for LAION-CLAP. The study concluded with a 15-minute semi-structured interview.

5.4.3 Summary of Semi-Structured Interviews

The overall results of these studies were positive. P2, a professional musician, mentions that "even though like I know a lot about synthesis and can probably get pretty far with just the DIVA interface, this actually saved me a lot of time." The other professional musician, P5, mentions that the system would be most useful for them when learning to use a new synthesizer. Participants (P1, P2, P3) also mention that they enjoy the discoverability aspects of the system, describing situations where they arrived at fitting but surprising new sounds. We also received invaluable constructive feedback that can inspire future directions. Below, we further analyze the feedback we received on each feature.

[4] `https://www.akaipro.com/ewi5000`

5.4.4 Multimodal Search

We observe that some participants use the multimodal search in a step-wise fashion (P1, P3, P6). First, they run a text search with a general description of their idea and then they run an audio search on a sound they like which potentially returns other sounds related to that one. P1, for example, started by searching for the sound of a piccolo and then narrowed their search by running an audio search on a sound they liked which led to a recorder sound which they felt to be most useful. P3 describes the advantage of a general initial text search when they were looking for ambient sounds: "My goal was very loose and it gave me some things that, for example the bass being a candidate for the kind of ambient sound I want, I wouldn't have thought of that myself." Participants sometimes had difficulties with the query format. P4, for example, thought typing "the sound of ..." everytime was "completely irrelevant." They also suggested that it might be better to have a pop-up window containing the results of an audio search to avoid losing the original content of the initial text search.

5.4.5 Genetic Mixing

The Genetic Mixing feature was appreciated by participants for the pleasant but unexpected sounds it created (P1, P2, P3, P4). P4 elaborates this saying "I think the mix feature allows you to kind of have these weird overlaps that you wouldn't normally have in a synth and that was really cool." The fact that our mixing algorithm achieves aesthetically pleasing and surprising results is bolstered by the reviews of the Multimodal Search due to the fact that users can search through new generations of sounds using the same search strategies as before. Some participants expected the mixes to be slightly more thorough, however. P2, for example, noted that they expected the Genetic Mixing to do some level of interpolation between the continuous parameters in the favorites list instead of just swapping parameters. P3 mentions that a useful addition would be a system that automatically names the new synthesizer sounds as opposed to the current generic naming convention.

5.4.6 Preset Modification

The preset modification feature was used by participants for making quick directed changes. Both professional musicians, P2 and P5, use this feature to make the final adjustments when attempting to replicate synthesizer sounds from songs they'd chosen. To this end, P2 claims that "even though like I know a lot about synthesis and can probably get pretty far with just the DIVA interface, this actually saved me a lot of time. P1 notes that when using synthesizers in the past "the search space seemed unlimited" but that this feature provides "a really fast way to just iterate through all of the different possibilities out there". They also claimed that the parameter group highlighter was a useful indicator of importance. Other participants highlight some non-intuitive aspects of this feature. P4 and P6 believed the numerical labelling of the examples corresponded to an increasing quantity. P3 felt that having to use nouns in the text queries (e.g. "the sound of a harsh synthesizer") required more mental effort than just typing an adjective or command (e.g. "more harsh"). P4 outright stated that they disliked the feature because it felt like it required them to understand what the functionality of the different parameter groups.

References

David Braun. 2021. Dawdreamer: Bridging the gap between digital audio workstations and python interfaces. *arXiv preprint arXiv:2111.09931* (2021).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

Philippe Esling, Naotake Masuda, Adrien Bardet, Romeo Despres, and Axel Chemla-Romeu-Santos. 2019. Universal audio synthesizer control with normalizing flows. https://doi.org/10.48550/arXiv.1907.00971 arXiv:1907.00971 [cs, eess, stat].

Philippe Esling, Naotake Masuda, and Axel Chemla-Romeu-Santos. 2020. FlowSynth: Simplifying Complex Audio Generation Through Explorable Latent Spaces with Normalizing Flows, Vol. 5. 5273–5275. https://doi.org/10.24963/ijcai.2020/767 ISSN: 1045-0823.

Miguel Grinberg. 2018. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.".

Andrew Horner, James Beauchamp, and Lippold Haken. 1993. Machine tongues XVI: Genetic algorithms and their application to FM matching synthesis. *Computer Music Journal* 17, 4 (1993), 17–29.

Cheng-Zhi Anna Huang, David Duvenaud, Kenneth C. Arnold, Brenton Partridge, Josiah W. Oberholtzer, and Krzysztof Z. Gajos. 2014. Active learning of intuitive control knobs for synthesizers using gaussian processes. In *Proceedings of the 19th international conference on Intelligent User Interfaces (IUI '14)*. Association for Computing Machinery, New York, NY, USA, 115–124. https://doi.org/10.1145/2557500.2557544

Yuyo Lai, Shyh-Kang Jeng, Der-Tzung Liu, and Yo-Chung Liu. 2006. Automated optimization of parameters for FM sound synthesis with genetic algorithms. In *International Workshop on Computer Music and Audio Technology*. Citeseer, 205.

Jimmy Lin. 1991. Divergence Measures Based on the Shannon Entropy. *IEEE Transactions on Information Theory* 37, 1 (1991), 145–151. https://doi.org/10.1109/18.61115

Naotake Masuda and Daisuke Saito. 2021. Synthesizer Sound Matching with Differentiable DSP.. In *ISMIR*. 428–434.

Allan Seago, Simon Holland, and Paul Mulholland. 2004. A Critical Analysis of Synthesizer User Interfaces for Timbre. Vol. 2. Research Press International, Bristol, UK, 105–108. http://oro.open.ac.uk/5688/ Num Pages: 4.

Yusong Wu*, Ke Chen*, Tianyu Zhang*, Yuchen Hui*, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. 2023. Large-scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-to-Caption Augmentation. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*.