Listening by Synthesizing

by

Manuel Cherep

M.S., Data Science,
École Polytechnique Fédérale de Lausanne (2021)

B.S., Mathematics and Computer Science,
Universidad Politécnica de Madrid (2016)

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning, in partial fulfillment of the requirements for the degree of

Master of Science in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2024

AUTHOR

Manuel Cherep
Program in Media Arts and Sciences
August 14, 2024

CERTIFIED BY

Tod Machover
Muriel R. Cooper Professor of Music and Media
Thesis Supervisor

ACCEPTED BY

Joseph A. Paradiso
Academic Head
Program in Media Arts and Sciences

Listening by Synthesizing

by

Manuel Cherep

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
on August 14, 2024, in partial fulfillment of the
requirements for the degree of
Master of Science in Media Arts and Sciences

Abstract

Generative audio models offer a scalable solution for producing a rich variety of sounds. This can be useful for practical tasks, like sound design in music, film, and other media. However, these models overwhelmingly rely on deep neural networks, and their massive complexity hinders our ability to fully leverage them in many scenarios, as they are not easily controllable or interpretable. In this thesis, I propose an alternate approach that relies on a virtual modular synthesizer; a computational model with modules for controlling, generating, and processing sound that connect together to produce diverse sounds. This approach has the advantage of using only a small number of physically-motivated parameters, each of which is intuitively controllable and causally interpretable in terms of its influence on the output sound. This design takes inspiration from devices long used in sound design and combines it with state-of-the-art machine learning techniques. In this thesis, I present three projects that use this formulation. The first is SynthAX, an accelerated virtual modular synthesizer that implements the core computational elements in an accelerated framework. The second, *CTAG*, combines the synthesizer with an audio-language model into a novel method for text-to-audio synthesis via parameter inference. This method produces more abstract sketch-like sounds that are distinctive, perceived as artistic, and yet similarly identifiable to recent neural audio synthesis models. The third is *audio doppelgängers*, sounds generated by randomly perturbing the parameters of the synthesizer to create positive pairs for contrastive learning, encompassing more of the variety found in real-world recordings, with controlled variations in timbre, pitch, and temporal envelopes. This method offers an efficient alternative to collecting real-world data, producing robust audio representations that compete with real data on established audio classification benchmarks. This thesis contributes tools for *understandably* generating rich

and diverse sounds, using them and their parameters for sound design and understanding at scale.

Thesis Supervisor: Tod Machover
Title: Muriel R. Cooper Professor of Music and Media

Listening by Synthesizing

by

Manuel Cherep

This thesis has been reviewed and approved by the following committee members:

THESIS SUPERVISOR

Tod Machover
Muriel R. Cooper Professor of Music and Media
Massachusetts Institute of Technology

THESIS READER

Rosalind W. Picard
Grover M. Hermann Professor of Health Sciences and Technology
Massachusetts Institute of Technology

THESIS READER

Josh H. McDermott
Associate Professor of Brain and Cognitive Sciences
Massachusetts Institute of Technology

"I may not find, but I seek."
— Vera Molnár

ACKNOWLEDGMENTS

I am lucky to have many people to thank.

My advisor Tod Machover, for allowing me to explore and for being as enthusiastic as me about sounds and animals.

Roz Picard, for her selfless advice, encouragement, thoughtful discussions about my work, and for always finding time to meet with me. Josh McDermott, for his rigor and scientific insights in everything related to auditory perception. Pattie Maes, for her trust, kindness, and all the research ahead of us.

Joe Paradiso, Zach Lieberman, and Kent Larson, for encouraging this research in many different ways.

Elena Glassman, for reminding me that kindness has a place in scholarship. Matt Groh, for assuring me that someone would care about my ideas when I needed it the most. Bence Ölveczky, Antonio Torralba, and Tom Griffiths for truly inspiring lectures that have changed the way I think.

Jean-Philippe Thiran, Jeffrey Huang, Konstantinos Sagonas, and Ángel Herranz, for their continuing support until this day.

The Fulbright Program, for funding my time at MIT.

The Opera of the Future group (and extended family), for staying together through it all: Jess Shand, Nikhil Singh, Kimy Lecamwasam, Ana Schon, Manaswi Mishra, Jessie Mindel, Max Addae, Rose Hegele, Alaa Algargoosh, Alexandra Rieger, and Clémence Taillandier.

To my incredible friends and colleagues here: Sam Chin, Kush Tiwary, Dexter Callender, Mike Jiang, Leticia Izquierdo, Vincy Xiao, Vera van de Seyp,

Cassie Lee, Naana Obeng-Marnu, Alfonso Rubio, Samantha Gutiérrez, Eyal Perry, Ziv Epstein, Cathy Fang, Rob Lewis, Noah Jones, Lancelot Blanchard, Cedric Honnet, Paris Myers, Wazeer Zulfikar, Kayla Briët, Patrick Chwalek, Shayne Longpre, and Will Brannon.

Javier González, Carlos Medina, Oriol Barbany, Andreea Salajan, Tzu-Chin Chang Chien, and Patricia Rocasolano, friends across the ocean yet always present. Anne Roffler and Matthias Hofer, for a great home after long workdays.

Sam Chin, for her never-ending enthusiasm and for always knowing what to do to brighten the day for everyone around her.

Kimy Lecamwasam, for her kindness and friendship, one that everyone should be able to experience.

Nikhil Singh, for influencing more than anybody else the way I approach research, for showing me how to evolve any idea into a significant contribution, and for selflessly sharing everything he knows with me, and everyone around him. His friendship has been one of the best privileges at MIT.

Jess Shand, for understanding and supporting me, for her contagious passion and creativity, and for always saying the most hilarious unexpected things. In a room full of everyone I know, I would be in a corner talking to her.

Finally, this thesis is dedicated to my mishpocha: Tito Dragoevich and Alicia Fraerman, my grandparents, who taught me how to climb the mulberry tree, how to fall, and how to get up; Julieta Cherep, my sister, for the laughter, the unconditional support, and for holding down the fort alone during family events; Alejandro Cherep, my dad, who could not see most of my accomplishments but was convinced that the best was yet to come; and Mirta Dragoevich, my mom, for teaching me the most important things in life, I owe you *almost everything*.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The extensive range of sounds that can be synthesized by generative audio models suggests the potential to revolutionize not only creative applications like sound design, but also to supply vast amounts of training data for increasingly complex systems [14]. These models predominantly rely on deep neural networks, often with billions of parameters and large latent spaces. While undeniably powerful, this complexity poses significant challenges. Their black-box nature obscures interpretability, as parameters lack physically plausible interpretations, and direct manipulation of their latent spaces for precise control remains difficult.

For example, two of the state-of-the-art solutions for text-to-audio synthesis, namely AudioGen [15] and AudioLDM [16], employ different architectures reaching up to over a billion parameters. As an alternative, I propose virtual synthesizers as lightweight, interpretable, and controllable generative audio models. Synthesizers have been used for years by sound designers—and musicians alike—to achieve more evocative and expressive effects in music, film, and other media. However, bridging the gap between this tool and modern computational paradigms is challenging. Though virtual, software-based versions have been built, they have not been suitable for this kind of work. For example, they may be computationally inefficient and designed for real-time

use, which limits the rate at which they can be automatically programmed. Instead, to get all the benefits of the synthesizer with the capabilities of a generative audio model we developed SynthAX [3], a fast modular synthesizer built for accelerators, which generates over 90,000 seconds of audio in just one second.

Synthesizer programming—that is, the act of creating new sounds through the careful analysis and modulation of synthesizer parameters—is a complex task. This process is especially non-intuitive for users without training, but it still takes hours for expert sound designers. For general sounds, this is a non-trivial task due to the complex relationship between the parameter space, the perceptual space, and the semantic space [17]. This has therefore animated the field of automatic synthesizer programming for decades. In a visionary conversation as part of the *I Love Quincy (1984)* documentary [18], Herbie Hancock—jazz musician and pioneer in the use of synthesizers—notes, "The machine doesn't do anything but sit there until we plug it in. It doesn't plug itself in. It doesn't program itself, *yet!*", to which Quincy Jones—legendary producer—responds, "It's on the way though!"

In this thesis, I introduce *CTAG* [19], a text-to-parameter method taking a text prompt to iteratively refine the parameters of our synthesizer to produce sounds that align semantically with the prompt, guided by a pre-trained audio-language model [2]. Contrary to the abovementioned state-of-the-art models with over a billion parameters, *CTAG* achieves high-quality results with only 78 physically-motivated variables. Sounds produced this way are also more abstract, capturing essential conceptual features over fine-grained acoustic details, akin to how simple sketches can vividly convey visual concepts. Results

show how *CTAG* produces sounds that are distinctive, perceived as artistic, and yet similarly identifiable to recent neural audio synthesis models, positioning it as a valuable and complementary tool. We also observe that conceptually similar sounds such as "gargling" and "boiling water" cluster together in this parameter space with semantic-preserving meaning, which also extends to interpolating sounds, providing access to intermediate acoustic transitions. This holds value for creativity, wherein users can continuously explore the space between concepts, and for scientific understanding, wherein we can probe behavior in response to stable variations in sound between established concepts.

CTAG shows that synthesizer programming is capable of recovering semantically recognizable sounds. If synthesizers can model such sounds, can we train models to learn to listen from them? Doing this would be useful because we could generate synthetic audio data at scale, which has emerged as a valuable tool since learning robust audio representations currently demands extensive datasets of real-world sound recordings. By applying artificial transformations to these recordings, models can learn to recognize similarities despite subtle variations through techniques like contrastive learning. However, these transformations are only approximations of the true diversity found in real-world sounds, which are generated by complex interactions of physical processes, from vocal cord vibrations to the resonance of musical instruments. Using *CTAG* would first require generating sounds with prompts corresponding to realistic sounds, which would be time-consuming. Instead, I present a solution by randomly perturbing the parameters of our synthesizer, generating *audio doppelgängers*—synthetic positive pairs with causally manipulated variations in timbre, pitch, and temporal envelopes. These variations, difficult to

achieve through transformations of existing audio, provide a rich source of contrastive information. Despite the shift to randomly generated synthetic data, this method produces strong representations, competitive with real data on standard audio classification benchmarks.

In summary, this thesis explores the development and application of a fast virtual synthesizer, demonstrating its controllability to generate sounds from text descriptions and sounds for creating robust audio representations via contrastive learning. As an underlying principle, we observe that even with a small number of parameters, virtual synthesizers can be surprisingly powerful, achieving results comparable to state-of-the-art neural networks in certain tasks. This research has been co-led with my co-first author. We worked closely throughout the research process, jointly developing concepts, engineering systems, and designing experiments through extensive collaborative sessions. The work has culminated in three papers:

- In Chapter 3: SynthAX: A Fast Modular Synthesizer in JAX, *AES 2023* [3]
- In Chapter 4: Creative Text-to-Audio Generation via Synthesizer Programming, *ICML 2024* [19]
- In Chapter 5: Contrastive Learning from Synthetic Audio Doppelgängers, *Under Review* [20]

2 RELATED WORK

2.1 PROGRAMMATIC SYNTHESIS

One important element of a synthesizer is allowing programmatic control. Indeed, many software synthesizers are ultimately written to be controllable by other software, such as VST plugins by DAW automation. However, not many synthesizers are designed to be fully specifiable and controllable in code written by end-users. Some well-known options include *Surge XT*[1] and *torchsynth* [1]. The former is written as a plugin that offers an API, and the latter is written as a library for non-realtime synthesis. In Chapter 3, we introduce SynthAX, implemented following the latter example, which means that there is not a direct application of our method to realtime synthesis. However, since JAX [21] compiles code to XLA, it is likely possible to implement SynthAX in a realtime synthesizer plugin to have it bridge these two different approaches to programmatic synthesis. In Chapter 4, we use SynthAX to generate semantically recognizable sounds guided by text descriptions, and in Chapter 5, we leverage SynthAX to rapidly produce diverse training data with controllable similarity between examples.

[1] https://surge-synthesizer.github.io/

Developed for audio synthesis, *torchsynth* [1] serves as a modular synthesizer that is capable of generating audio on a single GPU at $\geq$16200$\times$ faster than realtime. It consists of a variety of audio and control modules. The default synthesizer in *torchsynth* is *Voice*, which the authors used to generate a dataset containing a billion audio clips. We base SynthAX on *torchsynth* as it provides an existing and familiar reference point. We also compare to *torchsynth* in our experiments studying the performance.

2.2 SOUND SYNTHESIS

The earliest inroads in digital sound synthesis were made at the Bell Telephone Labs throughout the 1950s and 60s, culminating in the development of the open-source MUSIC V software [22]. Back then, analog synthesizers remained accessible primarily only to a handful of research institutions, recording studios, and experimental musicians due to their prohibitive size and cost [23]. It wasn't until the advent of mass-produced digital synthesizers, like the Yamaha DX7 in 1983 and the Roland D-50 in 1987, that less-established artists and amateurs could readily explore and contribute to the increasingly complex landscape of sounds [24]. Rapid advancements in digital signal processing and the explosion of personal computing made possible the widespread use of software synthesizers in the 2000s. Sound synthesis is now not only an integral component of live and recorded music, from popular to experimental styles,

but also a cornerstone of sound design for video games, film, advertising, and more.

Neural audio synthesis [10]—the combination of neural networks with audio processing techniques—consists of two main strands: approaches that generate audio waveforms directly in the time domain, and those that do so in the frequency domain. WaveNet [25] notably introduced an autoregressive approach to audio synthesis by predicting one sample at a time. This slow iterative sampling approach, later refined in SampleRNN [26] and WaveRNN [27], reflects the sequential nature of audio data, in contrast to images wherein GANs with global latent conditioning and efficient parallel sampling quickly became a dominant method for synthesis. Later, WaveGAN [28] and GANSynth [29] demonstrated that GANs could in fact be used to synthesize locally-coherent audio, outperforming sequential models' speed by several orders of magnitude while maintaining a focus on high-fidelity, natural-sounding audio.

A third strand of so-called *oscillator* models, largely propelled by Differentiable Digital Signal Processing (DDSP) [30] is physically and perceptually motivated by the rich history of synthesis and signal processing techniques. Our approach for generating semantically recognizable sounds, introduced in Chapter 4, is motivated by this direction, but relies on a simple synthesizer architecture, CLAP [2], for text-conditioning, and gradient-free optimization to provide a simple, training-free solution.

2.3 LANGUAGE–SOUND CORRESPONDENCE

Advances in multi-modal sound-language models have been partly motivated by CLIP [31] for images. Wav2CLIP [32] builds directly onto CLIP by adding an audio encoder, and VQGAN+CLIP [33] generates and edits images guided by text prompts. Audio representation models, such as Microsoft's CLAP [34] and LAION-CLAP [2], emulate CLIP's approach by using contrastive learning on audio-text pairs. We use LAION-CLAP as our audio-language model in this work.

Other recent approaches cast audio generation as a language modeling task. AudioGen [15] is an autoregressive model conditioned on text inputs. AudioLM [35] uses a multi-stage Transformer-based language model. WavJourney [36] uses text instructions to create scripts, which are then used for compositional audio creation. Make-An-Audio 1 and 2 [37, 38] offer text-to-audio synthesis with prompt-enhanced diffusion models, using CLAP to map text to latent representations with a spectrogram autoencoder. AudioLDM [16] learns continuous audio representations from CLAP latents and can perform text-guided audio manipulations. In Chapter 4, we compare to two state-of-the-art solutions, namely *AudioGen* and *AudioLDM*, in our experiments. Our goals differ significantly from those of these models, as we seek to generate abstract yet high-quality sounds, rather than literal recording-like renditions.

2.4 ABSTRACT SYNTHESIS

Visual sketching offers an intuitive analog to abstract sound synthesis. Minimal representations like monochromatic line drawings might use only straight lines and curves with no additional shading or color. These renderings are non-photorealistic; they evocatively convey meaning while emphasizing a subject's essence over its real-world presentation. They can also reveal insights about a subject's underlying geometry, proportions, and symbolism that may be obscured in more realistic depictions.

The problem of computing recognizable and insightful abstract renderings has seen more progress in the visual than the audio domain. CLIPasso [39] leverages CLIP to distill semantic meanings from images and sketches alike and thereby guide text-to-image generation, varying the number of strokes according to the desired level of abstraction. CLIPTexture [40] enables a user to manipulate a simple sketch or layout through textual descriptions. CLIPVG [41] follows the same progressive optimization approach, but performs image manipulation using vector graphics rather than pixels. ES-CLIP [42] tackles the problem via evolution strategies, generating configurations of colored triangles on a canvas, then assessing their fitness for further iteration. We were inspired by this approach for generating sketch-like sounds in Chapter 4, though we rely on the well-established, easily interpretable, and tweakable paradigm of modular synthesis.

In the auditory domain, the Sound Sketchpad [43] combines sounds together using audio-visual sketches, and the SkAT-VG project [44] applies vocal and

gestural manipulation as natural sketching tools. In our approach, we focus on language input and synthesis rather than the composition of existing sounds.

2.5 INTERPRETABLE & CONTROLLABLE SYNTHESIS

Interpretability and controllability of results are essential to human-machine co-creation, in which it is often desirable to closely examine, understand, and fine-tune an artifact. For creative sound design using neural synthesis methods, it can be impossible to retrace decisions made by a complex neural synthesis model en route to synthesizing an output. The model may also not provide any opportunity to iteratively refine the output. Some prior work [45] highlights the potential of program synthesis for interpretability in sequence data, including music. Some neural synthesis models integrate techniques like timbre-regularization [46] to bridge powerful synthesis methods with perceptually-motivated organization of latent spaces. By contrast, we are able to synthesize semantically recognizable sounds in Chapter 4 with a fully interpretable and controllable parameter space without requiring us to develop additional neural infrastructure. In Chapter 5, this controllable parameter space allows us to produce pairs of sounds with controllable similarity between them.

2.6 THE SYNTHESIZER PROGRAMMING PROBLEM

Despite the near-ubiquitous presence of synthesized sound in modern music, synthesizer programming—that is, the act of creating new sounds through careful analysis and modulation of synthesizer parameters—is a complex task that can often impede the creative process, if not bar entry entirely. In particular, the conceptual disconnect between parameter settings and the associated auditory output [47] makes synthesizer programming especially non-intuitive without special training. Recent work has investigated techniques for inverse synthesis—given a target sound, infer the parameter setting that will emulate the sound to the closest extent possible—on both musical sounds [48] and real-world sounds, such as animal vocalizations [49], including deep learning methods to learn invertible mappings [50]. However, this task still requires a specific audio clip to start. In Chapter 4, we provide text-to-parameter inference to bridge this gap, generalizing beyond specific audio files to broader semantic notions of arbitrary sounds.

2.7 LEARNING FROM SYNTHETIC DATA

Synthetic data, artificially generated information rather than collected from real-world sources, has emerged as a valuable tool for learning across various domains [51, 52, 53, 54]. By addressing data scarcity, privacy concerns [55, 56], or removing biases [57, 58], synthetic data offers a promising avenue to

complement scarce [59] real-world data and further drive progress in machine learning research.

Audio presents unique challenges due to the complexity of waveforms and temporal dependencies. Synthetic data has found applications in subareas like speech recognition [60, 61, 62, 63, 64] leveraging text-to-speech systems for detecting unspoken punctuation [65], recognizing low-resource languages [66], increasing acoustic diversity [67] or detecting out-of-vocabulary words [68]. However, non-speech audio domains can be highly diverse, requiring more complex approaches to data synthesis. In this domain, synthetic data has been used for specific tasks like timbre-text alignment [69] and vocoding [70]. The partially synthetic NSynth [10] dataset has also been used for pitch estimation and instrument classification. In our work, we tackle the general audio domain, proposing a synthetic data approach that can produce diverse sounds for general-purpose audio representation learning.

In computer vision, synthetic data is more popular and has been employed in different tasks to improve performance [71, 72, 73, 74, 75, 76, 77, 78]. While initially focused on using graphics engines to generate photorealistic scenes, recent work has investigated sampling synthetic data from deep generative models [79, 80, 81, 82, 83, 84, 85, 86, 87, 14, 88, 89, 90]. However, these models aim to produce realistic images and still depend on real image datasets for training or synthesis. Thus, recent work has pushed away from realism, generating synthetic data such as fractals [91], or through other procedural noise models [92, 93] to use as training data for visual representation learners. In Chapter 5 we leverage synthetic data while also abandoning realism by

leveraging randomly generated synthetic sounds to learn audio representations for downstream tasks.

2.8 CONTRASTIVE LEARNING

A common strategy for learning from unlabeled data is *contrastive* learning. In this technique, we seek representations that are *invariant* to minor differences, i.e. they encode a space in which similar objects are closer together, and dissimilar objects are further. A classic strategy for this is to use data *augmentations*, transformations which noticeably alter a datapoint (for example, randomly cropping an image) without changing its essential content (e.g. what the image is of, such as a cat). These transformed versions then become a *positive pair*, while other examples (e.g. an image of a dog) become *negatives*. In audio, contrastive learning has been used extensively to produce high-quality representations for downstream tasks [94, 95, 96, 97, 98]. However, contrastive learning typically relies on augmentations with limited variation, by effecting already-existing material or sampling with offsets. Instead, in Chapter 5 we introduce a method that produces synthetic positive pairs that deviate through randomly perturbed parameters which causally influence acoustic properties of the output, such as the pitch of oscillators, or the temporal envelope of signals in a mixture. This allows us to mimic more naturalistic variation in auditory features without (or in addition to) applying post-hoc effects.

3 SYNTHAX: A FAST MODULAR SYNTHESIZER IN JAX

3.1 INTRODUCTION

> *"It's hardware that makes a machine fast. It's software that makes a fast machine slow."*
>
> — **Craig Bruce** [99]

Realtime sound synthesis is a cornerstone of modern audio production. It affords producers the ability to tweak sounds and hear them change; a loop of perception and action that results in diverse auditory creations to support music, film, and other media. Modern audio technologies increasingly employ techniques that benefit from *automatically* tweaking synthesizers, such as optimization and machine learning. In these scenarios, the ability to rapidly tweak sounds and compute with them at scale offers a vast space of opportunities for designing and developing powerful new audio technologies. As such, fast sound synthesis can be an essential tool. We define faster-than-realtime as generating more than one second of audio per second of processing time. In particular, we deal with cases where the processing is a lot faster than this (i.e. >1000x).

In this paper, we introduce SynthAX, a fast modular synthesizer written using the JAX [21] framework for accelerated and differentiable computing. By offering synthesis at speeds that peak at over 80,000× realtime, SynthAX provides a high-performance, flexible virtual modular synthesizer in the form of an expanding and easily extensible open source Python library. Additionally, we implement an API based on *torchsynth* [1], a recent high-performing synthesizer written in PyTorch, to allow for an easy substitution for end-users. Our results in this paper show considerable speedups over *torchsynth*, ranging up to just under 9× depending on the hardware configuration and batch size.

3.2 SYSTEM DESIGN

The design of the API is inspired by the inherent modularity of hardware synthesizers. SynthAX leverages the power of JAX [21] to build on *torchsynth* [1], which is a state-of-the-art high-throughput synthesizer implemented in PyTorch to take advantage of its accelerated computational routines. Maintaining a similar API makes the transition for end-users seamless without any major rewriting or learning curve.

Each module serves a different function but can be connected together to create a synthesizer. SynthAX modules mimic their counterparts in analog and digital synthesizers, consisting of amplifiers, envelopes, filters, keyboards, low-frequency oscillators (LFOs), mixers, and voltage-controlled oscillators

[1] https://github.com/PapayaResearch/synthax

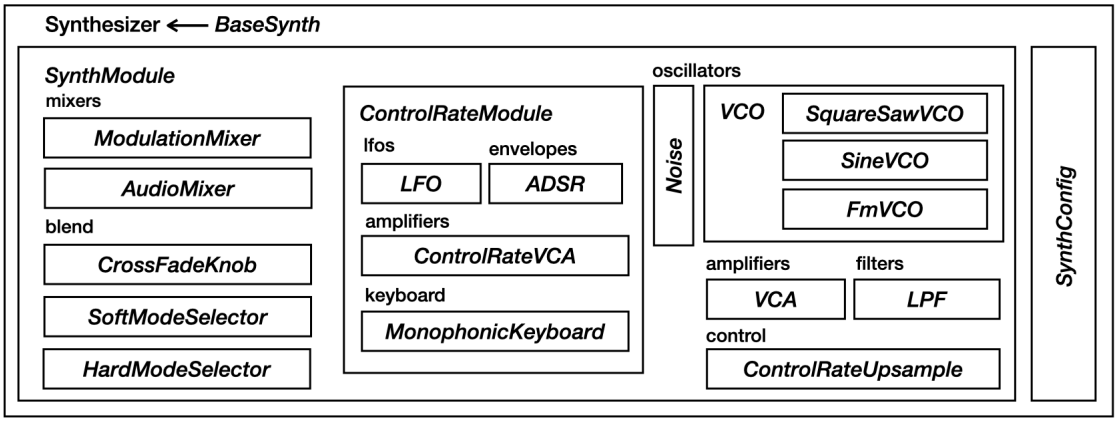

Figure 1: Structure of the API. We separate the synthesis modules into Python modules which group related elements. These modules are shown in lower-case letters above the relevant classes. The class inheritance structure, which mirrors *torchsynth* [1], is indicated by the *TitleCase* names. Inner boxes are subclasses of the larger boxes they are embedded in.

(VCOs). The output from these modules can represent audio signals or control voltages, depending on the module's intended function. Audio modules, such as VCOs, produce audio signals. Control modules, such as LFOs, produce "control voltages" that modulate the parameters of other modules. The keyboard outputs parameters that are used as input for other modules. All modules follow the Flax [100] module system known as Linen to organize the modules into independent components. Figure 1 shows the structure of the API, where a synthesizer consists of modules and a configuration.

In our implementation, we aim for allowing users flexibility in how they specify synthesizers. Modules with parameters can be initialized in a few different ways. If only initial values are given, they are expected to be in human-readable (i.e. unnormalized, e.g. frequency in Hz) range within the default ranges of the parameters. Alternatively, the modules also accept range objects, which specify only a range within which parameter values are initialized

```python
import jax
from synthax.config import SynthConfig
from synthax.synth import ParametricSynth

# Generate PRNG key
config = SynthConfig(
    batch_size=16,
    sample_rate=44100,
    buffer_size_seconds=4.0
)
# Instantiate synthesizer
synth = ParametricSynth(
    config=config,
    sine=1,
    square_saw=1,
    fm_sine=1,
    fm_square_saw=0
)
# Initialize and run
key = jax.random.PRNGKey(42)
params = synth.init(key)
audio = jax.jit(synth.apply)(params)
```

Listing 1: Code snippet for generating audio with a *ParametricSynth*. This synthesizer supports a user-configured architecture, in contrast to the *Voice* synthesizer which encodes a fixed topology design (78 parameters). This allows control of the degrees of freedom available to manipulate the sound synthesis.

uniformly randomly. Finally, users can also provide the initial values and ranges together as an object. In all cases, the parameters store the values in the (normalized) interval [0, 1].

In addition to the differences between SynthAX and *torchsynth* that arise from JAX features such as easy and flexible vectorization, parallelization, and just-in-time (JIT) compilation, we introduce these additional features: a filter module, currently containing a simple low-pass filter that can be shaped by

control modules; a parametric definition of a synthesizer to easily explore different synthesizer topologies; functions to write and load a synthesizer including its hyperparameters and parameters, in the human- and machine-readable YAML format. This also means that synth specifications can, in principle, be directly composed in YAML and loaded to a synth with a matching parameter architecture.

We adhere to JAX's explicit randomness handling in our design. JAX uses a pseudo-random number generator (PRNG), an algorithm that produces sequences of numbers that approximate true randomness given an initial key (i.e. value). Therefore, users need to provide such random keys to their synthesizers. Though this adds an extra consideration, it also ensures better reproducibility. Listing 1 shows how to define a configuration, instantiate a parametric synthesizer and, finally, synthesize audio.

JAX supports a wide variety of hardware and leverages powerful function transformations such as just-in-time compilation (JIT), auto-vectorization, and hardware parallelism. We can vectorize (jax.vmap) and parallelize (jax.pmap) in a single line of code. It also conforms to the *Single-Program, Multiple-Data* (SPMD) model, which means that the same computation for different input data runs in parallel on multiple devices. In order to maximize performance and throughput when using JAX, SynthAX renders audio in batches.

Extending SynthAX can be done seamlessly due to its modularity, since the API is designed to easily integrate other synthesizers or modules. SynthAX joins the JAX ecosystem and can be easily integrated with other well-known libraries such as Optax [101], evosax [102], EvoJAX [103], and QDax [104].

3.3 RESULTS

3.3.1 Performance Evaluation

First, we characterized the speed and memory performance of SynthAX. We used *torchsynth* [1] as a strong baseline to compare against, since it is the *de facto* state-of-the-art fast synthesizer and can take advantage of similar hardware acceleration capabilities (e.g. GPUs). For both synthesis libraries, we use the *Voice* synthesizer with 78 parameters. In our setup, we computed the time needed to synthesize 100 batches of sounds at different batch sizes (powers of 2 from 2 to 1024). We randomized the synthesis parameters for each batch. As *torchsynth* does, we also report the speed as compared with realtime synthesis. We calculated this as

$$\frac{\text{Num. Batches} \times \text{Batch Size} \times \text{Sound Duration}}{t}$$

where t denotes the time taken for one loop of 100 batches. Finally, we also report memory usage in GB after each 100-batch loop.

We report averages over 10 100-batch loops for all three quantities (time, speed $\times$ realtime, and memory), to account for variance. Additionally, we computed a full set of results for a GPU and a CPU, although we expect GPUs to be the primary usage platform. To account for the effect of JAX's [21] JIT compilation, we produced one batch of sounds (for both SynthAX and the *torchsynth* baseline) at the very beginning, outside the evaluation loop. This is

so that we measure the typical performance, as the JIT compilation only needs to occur once.

These results are given in Figure 2. We do not show error bars as the results are generally stable, resulting in very small variance. Overall, we see that SynthAX substantially outperforms *torchsynth* on time-based metrics for both CPU and GPU. At peak performance within this evaluation, SynthAX shows more than 80,000× realtime synthesis speed. SynthAX shows a comparable memory utilization profile to *torchsynth*, especially lower at higher batch sizes on GPU and CPU. We disabled JAX's memory preallocation for our experiments to measure the real memory footprint.

For direct comparison, Figure 3 plots the speedup over *torchsynth*. This is computed as the ratio of time taken to synthesize 100 batches, computed per 100-batch loop, and then averaged across the 10 runs. We provide min/max error bars to show the full range. This figure shows that the speedups range from just over 2× (some batch sizes on CPU and very large batches on GPU) to almost 9× at the peak speedup level (batch of 32 sounds on GPU).

3.3.2 *torchsynth* Replication

We replicated the examples from *torchsynth* [1] for reproducibility. These include instantiating ADSR envelopes both randomly and with set parameters, VCOs, LFOs, VCAs, mixers, and their synthesizer architecture *Voice*. Figure 4 shows some of the resulting spectrograms considering different VCOs and setups in *torchsynth* and corresponding match in SynthAX.

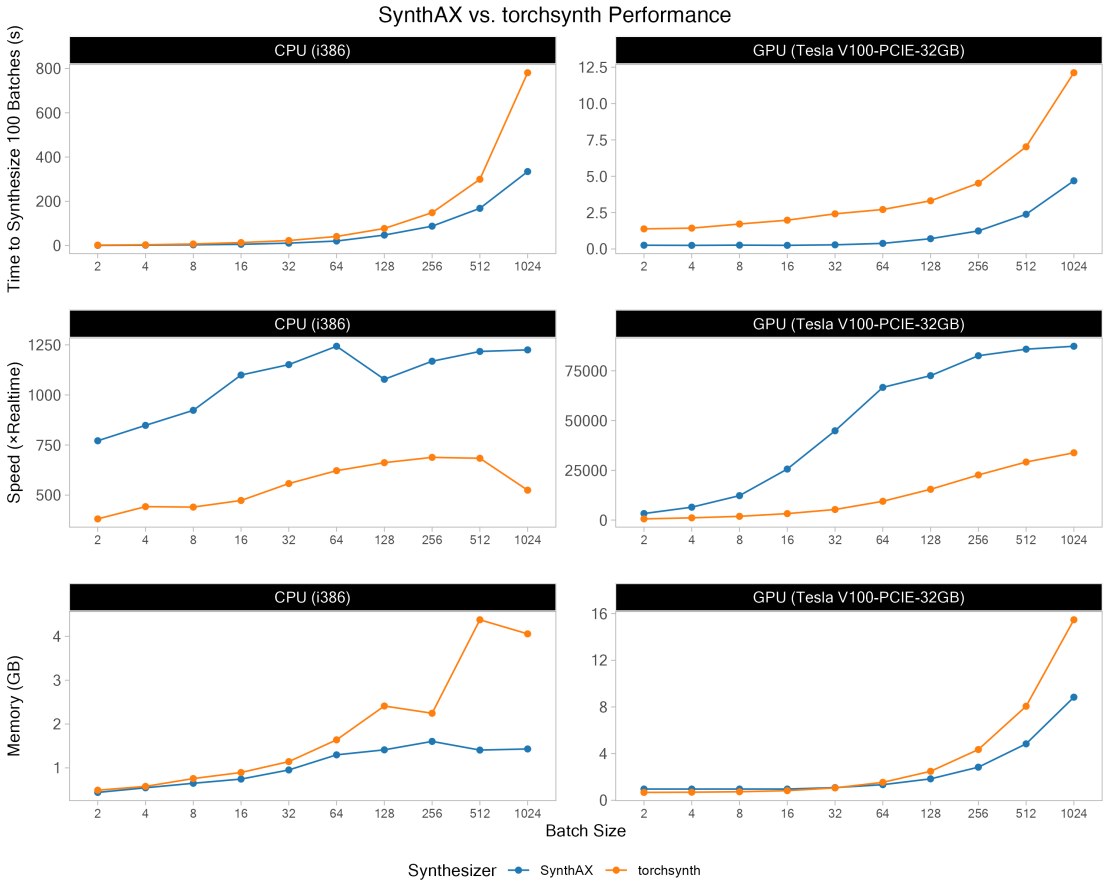

Figure 2: Results from performance evaluation, compared with *torchsynth*, on **(Left)** a 2017 iMac with an Intel Core i7-7700K CPU @ 4.20GHz, and **(Right)** an NVIDIA Tesla V100 GPU. Values shown are averaged over 10 runs. We use the *Voice* synthesizer in both SynthAX and *torchsynth*, randomizing parameters each batch. **(Top)** Time to synthesize 100 batches of sound at different batch sizes (given in seconds). **(Middle)** Time reinterpreted as speed $\times$ realtime, i.e. seconds of sound generated per second of computation time (see Section 3.3.1 for details). **(Bottom)** Memory utilization in GB. Overall, SynthAX shows significantly faster performance while retaining a similar memory utilization profile.

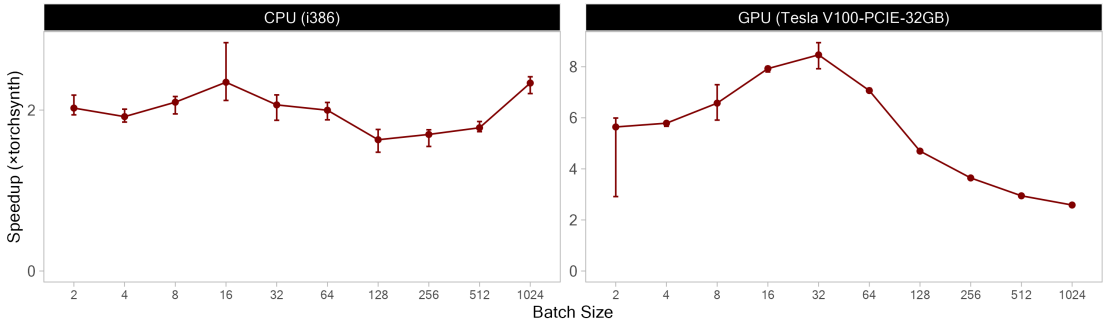

Figure 3: A direct comparison showing speedups relative to *torchsynth* [1] per batch size, again for 100-batch total times averaged across 10 runs. Error bars here show min/max results. Overall, SynthAX is more than double the speed in all cases, and peaks at almost 9× the speed of the already accelerated *torchsynth* implementation. As previously, these results are on the *Voice* synthesizer, a 78-parameter synthesizer, where parameters are randomized for each batch.

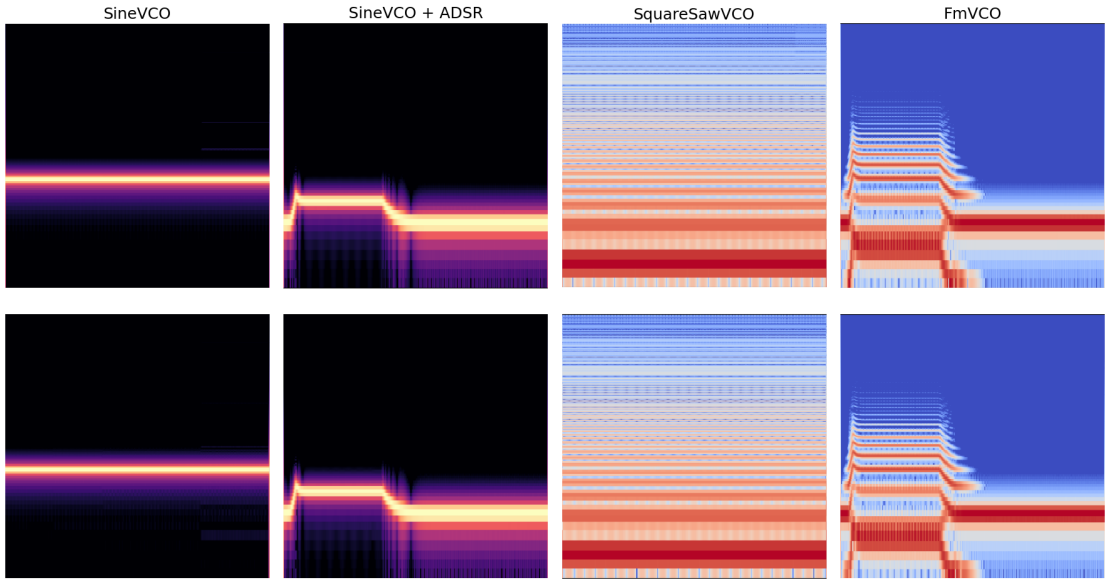

Figure 4: Spectrograms for the examples in *torchsynth* (**Top**) and the replication in SynthAX (**Bottom**). From left to right, we show a simple sine wave, a sine wave with an ADSR envelope modulating the frequency, a square wave, and an ADSR envelope-modulated FM patch. The results show clear replication of the output spectrotemporal features.

3.4 APPLICATIONS

3.4.1 Audio Representations

An advantage of synthesized sounds is that they also contain the associated synthesis parameters. In self-supervised representation learning problems, datasets that result from synthesis can be used to formulate parameter prediction problems. For instance, pitch recognition is a prominent auditory processing problem for which synthesized datasets hold significant promise. Recent work on audio representation learning has employed the Surge XT pitch dataset [1] to evaluate representations on such a task [105, 106]. Many other such prediction problems could be formulated for both training and evaluation, as they expose ground truth information as labels. A synthesizer can generate a large variety of sounds that vary in timbre while holding pitch constant, or conversely which vary in pitch but hold timbre constant for a task such as instrument recognition.

3.4.2 The Synthesizer Programming Problem

One particular area where SynthAX can be useful is in the synthesizer programming problem [47], and specifically the task of parameter inference [107]. A canonical formulation of this asks an algorithm to program a synthesizer to match a given sound. The difficulties of manually programming complex synthesizers are well-established [108], and as such a variety of tech-

niques [109, 49, 110, 111, 112, 113, 114, 107] and even software libraries [115] have been developed to approach this through the lens of automatic matching. Typically, algorithms used are those common to other search and optimization problems, such as genetic algorithms and even gradient-based optimizers. Given a sound, these algorithms seek to minimize some measure (often perceptually-motivated) of the "distance" between the target sound and a synthesized candidate by tweaking the synthesis parameters. SynthAX can accelerate such applications by speeding up the synthesis, often the most costly step in these problems. Additionally, SynthAX can be combined with other parts of the pipeline written in JAX [21] (such as evosax [102]) to provide a broader speedup for synthesizer programming by matching target sounds.

3.5 CONCLUSION

In this chapter, we presented SynthAX, a fast modular synthesizer implemented in JAX. We showed that SynthAX generates sounds orders of magnitude faster than realtime, and significantly faster than existing solutions to accelerated sound synthesis. We discussed the possible applications of this synthesizer in research and production problems involving intelligent sound processing and synthesis. In the future, we intend to expand it with more modules and a user interface. By open sourcing this library, we invite contributions towards a high-performance, robust, and well-documented synthesizer that we hope will eventually parallel commercial software synthesizers in the range of possible

sounds producible, while retaining the performance benefits which we observe in our experiments on this initial implementation.

4 CREATIVE TEXT-TO-AUDIO GENERATION VIA SYNTHESIZER PROGRAMMING

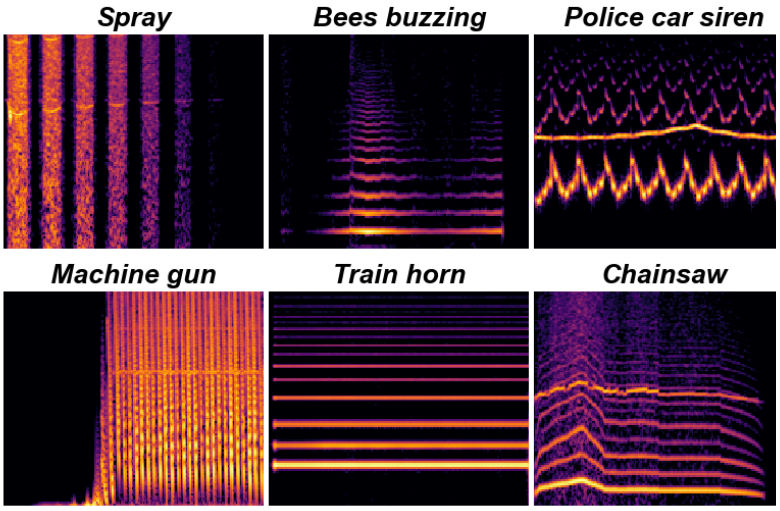

Figure 5: *CTAG* leverages a virtual modular synthesizer to generate sounds capturing the semantics of user-provided text prompts in a sketch-like way, rather than being acoustically literal. Spectrograms of auditory outputs corresponding to six text prompts showcase the range of sounds this approach can yield, accompanied by a fully interpretable and controllable parameter space.

4.1 INTRODUCTION

"Of course, bubbles don't make sound, but this is the magic of sound design...you can create the concept of a sound and it seems real."

— **Suzanne Ciani** [116]

In creative sound design, realism isn't everything. In the late 1970s, composer Suzanne Ciani famously demonstrated this principle with her iconic *Coca Cola pop and pour* sound effect. This sound, which has become synonymous with the refreshing experience of opening a soda, was not recorded from an actual soda bottle, but skillfully crafted using a Buchla synthesizer. Ciani's work illustrates the immense power of abstraction in auditory representation, where the essence of a concept can be expressed without mimicking real-world acoustic details, while achieving greater impact.

This approach extends beyond single examples into the domain of procedural sound design: creating sounds algorithmically using parameters that can be manipulated to achieve desired sonic effects. By applying procedural techniques, sound designers can often transcend what's physically plausible to obtain by recording real-world events. These methods can lead to highly evocative and expressive sounds in music, film, video games, advertising, product design, and other media.

Neural audio synthesis methods have transformed the state of sound design, enabling specifying sound ideas using intuitive inputs like textual prompts. However, there remains unrealized potential in integrating expressive sound design principles into neural audio synthesis. Current techniques prioritize acoustic recreation and end-to-end application, often overlooking creative possibilities for evoking emotions or concepts, and interactive aspects like manipulating, iterating, and interpolating between sounds. While recent advances showcase remarkable capabilities in replicating real-world sounds, this emphasis can limit the creative palette and expressive potential of generated audio. We propose a method to bridge this gap.

Overall, this work contributes:

- A novel method that integrates a virtual modular synthesizer with a pretrained audio-language model for generating sounds that resonate with human intuition without being literal representations.
- A lightweight, fully interpretable, and controllable synthesizer resulting from our approach, allowing for easy inspection and tweaking for creative purposes.
- Experiments evaluating different approaches to solving this problem, varying optimization algorithms, sound durations, and synthesis architectures.
- Qualitative and quantitative results that highlight how sounds from our method have distinct features from those produced by other neural audio generators, while still being identified at similar rates. We conduct a user study as a gold standard evaluation, given the novelty of the task, which shows the identifiability and potential artistic value of *CTAG*'s sounds.
- Examples of this approach generating several datasets of sounds with their synthesis parameters, and interpolating between different sounds in the parameter space.

We open-source our approach[1], both to provide a tool for novices and experts alike to realize their ideas, as well as to provoke future audio generation paradigms that recognize abstraction as an important factor for creative expression.

[1] ctag.media.mit.edu

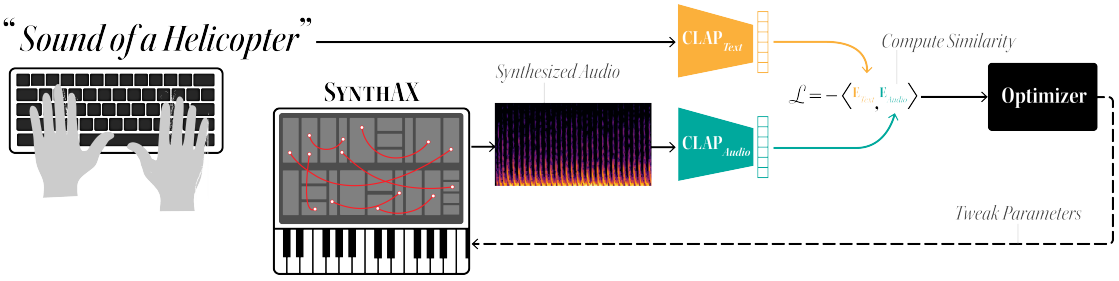

Figure 6: High-level overview: we use the LAION-CLAP model [2] to compute the similarity between a user-provided text prompt and SynthAX's [3] output. The optimization procedure iteratively adjusts the parameter settings.

4.2 METHODS

Our methodology hinges on three pillars: a synthesizer, implemented via SynthAX [3], gradient-free optimization methods, implemented via the Evosax [117] evolutionary optimization library, and an objective function based on the LAION-CLAP [2] model, which we use to estimate semantic alignment between the synthesized audio and its corresponding text prompt (see Figure 6 for an overview of the pipeline).

4.2.1 Synthesizer

We use a simple synthesizer implementation available in SynthAX, a fast modular synthesizer written in JAX [21]. We specifically use the *Voice* synthesizer architecture, adapted from *torchsynth* [1], which has already been used for programmatic resynthesis of sounds [49]. It consists of 78 parameters for a monophonic keyboard, two low-frequency oscillators (LFOs), six ADSR envelopes, a sine voltage-controlled oscillator (VCO), a square-saw VCO, a noise

generator, voltage-controlled amplifiers (VCAs), a modulation mixer and an audio mixer. All parameters are initialized uniformly, $\theta_i \sim U(0,1)$.

In addition to this architecture, we evaluate the following variants in increasing order of architectural complexity:

- *ShapedNoise*: An 18-parameter synthesizer consisting of a noise generator, and two control elements to shape the noise amplitude over time: an ADSR envelope, and a low-frequency oscillator (LFO). These are combined into a modulation signal through a modulation matrix, which itself has learnable weights for this combination.

- *OneOsc*: A 23-parameter synthesizer consisting of a sine wave voltage-controlled oscillator (VCO), and the same two control elements as above. These elements are combined into two signals through a modulation matrix, one each for frequency and amplitude.

- *NoLFO*: A 29-parameter two-VCO synthesizer, where one is a sine wave oscillator and the other is a square-saw wave oscillator with a "shape" parameter which controls the degree of "square-ness" vs. "saw-ness". This synthesizer has no LFO components, all modulation is conducted by two ADSR envelopes combined into four separate modulation signals (pitch and amplitude controls for each of the two VCOs).

- *NoNoise*: A 51-parameter synthesizer with two VCOs (as before), and a more complex modulation structure. Here, there is a single LFO, but there are additional ADSRs to modulate the frequency and amplitude of this

LFO. The modulated LFO and two ADSR envelopes comprise the inputs to the modulation matrix.

- *Voice+FM*: A 130-parameter synthesizer which adds a frequency modulation (FM) component to the original *Voice* architecture.

For reference, an ADSR envelope is a piecewise control signal consisting of linear or exponential segments: **A**ttack, **D**ecay, and **R**elease, which specify the duration of each envelope segment. The **S**ustain parameter is the level of the control signal after the decay phase. An LFO is an oscillator whose frequencies are typically lower than audible frequencies, i.e. below 20-40 Hz. These are used for periodic control of synthesis parameters.

In all our experiments, the synthesizer has a control rate of 480 Hz and the audio is generated in batches at a sample rate of 48 kHz. This sample rate is much higher than that commonly used for neural audio synthesis systems (often 16 kHz) and therefore admits much more high-frequency content to be generated.

4.2.2 Optimization

During initial experiments, we found the gradients of our differentiable synthesizer to be highly unstable. This instability hindered optimization performance even after attempting mitigation strategies. Recent works in abstract visual synthesis have shown that non-gradient methods can achieve state-of-the-art results without relying on gradient information [42]. Given these findings, we decided to explore non-gradient approaches which are more suitable for

Algorithm 1 Our optimization procedure for producing sounds in *CTAG*. Note: d is the number of parameters of the synthesizer S; for simplicity we omit batches.

Require: Text prompt p
Require: Population/batch size N
Require: Iterations M

Components:
CLAP text embedding model $C_t(p) \rightarrow E^p$
SynthAX synthesizer $S(\Theta) \rightarrow X^a$
CLAP audio embedding model $C_a(X^a) \rightarrow E^{X^a}$
Optimization Strategy: O

Initialize:
Synthesis parameters $\Theta = \{\theta_1, \ldots, \theta_N\}, \theta_i \sim U(0,1)$
Flattened parameters $\Theta_f \in [0,1]^{N \times d} = Flatten(\Theta)$

for $i = 1$ **to** M **do**
$\Theta_{f_{\text{new}}} \leftarrow O_{\text{ask}}(\Theta)$ *Generate candidates*
$\Theta_{\text{new}} \leftarrow \text{Reshape}(\Theta_{f_{\text{new}}})$ *Reshape*
$X^a \leftarrow S(\Theta_{\text{new}})$ *Synthesize audio*
$E^{X^a} \leftarrow C_a(X^a)$ *Get audio embeddings*
$F \leftarrow -E^{X^a} E^{p\,T}$ *Compute fitness*
$O_{\text{tell}}(\Theta_{\text{new}}, F)$ *Update optimizer state*
$\Theta \leftarrow \Theta_{\text{new}}$
end for

$\theta^* = \arg\min_\theta F$ *Select optimal parameters*

our synthesizer's instability and have demonstrated effectiveness for this task. Focusing efforts here allowed us to sidestep gradient issues while leveraging successful techniques from related synthesis domains.

We experimented with several non-gradient optimization algorithms, using implementations from Evosax [117]. Specifically, we examined simple baselines like random search and a simple genetic algorithm [118], well-known methods like CMA-ES [119] and Particle Swarm Optimization [120], and state-of-the-art methods like Learned and Discovered Evolution Strategies [121]. For each

algorithm, we first tuned hyperparameters using Bayesian optimization via the Adaptive Experimentation (AX) platform [122]. We tuned for 50 trials on the ESC-10 dataset, a subset of ESC-50 [123]. Note that the hyperparameter tuning uses no privileged information and can easily be applied downstream on new prompt sets to maximize the performance.

The optimization procedure is specified in Algorithm 1.

4.2.3 Objective Function

We use LAION-CLAP [2] with an HTSAT-based audio encoder [124] and a RoBERTa-based text encoder [125]. We used the *audioset-best* checkpoint for general audio less than 10 seconds long.

The encoders process the audio data X_i^a in batches of size $\mathcal{B}$ where $\mathcal{B}$ corresponds to the optimizer's population size, along with a prompt p. Note that (X_i^a, p) is one particular pair of synthesized audio with input text prompt. We extract the audio embeddings $E_{\mathcal{B}}^a \in \mathbb{R}^{\mathcal{B}\times 512}$ and the text embeddings $E^p \in \mathbb{R}^{1\times 512}$ with the encoders and use them to calculate the similarity score $\in [0, 1]$ between a batch of audio data and a specific prompt.

$$X_i^a = \mathcal{S}(\theta_i) \tag{1}$$

$$\theta^* = \arg\min_{\theta} \; -E_i^{\mathcal{S}(\theta_i)} E^{pT} \tag{2}$$

Equation (1) shows how the synthesizer $\mathcal{S}$ takes parameters θ_i and produces a sound (in practice, this is done batched). Then Equation (2) formulates the optimization problem to optimize the similarity score between each audio in the batch and one given text prompt using their corresponding embeddings.

4.2.4 Evaluation Metrics

Since we propose a novel synthesis task without existing evaluation metrics, we devise a principled evaluation suite that allows us to quantitatively assess our contributions, in addition to qualitatively reviewing synthesized examples.

CLASSIFICATION EXPERIMENTS To determine whether our generated sounds are more abstract than neural synthesis methods, we compared results on pre-trained classifiers with sounds generated from their class labels. Lower scores can indicate a distribution shift from real audio, despite explicitly optimizing for similarity to the label. We complement with human listener ratings.

Without a perfect synthesis engine, any methods to generate sound will introduce a distribution shift from real audio. In our case, there is a deliberate domain shift to abstract audio. We evaluate on two well-known datasets. The first is ESC-50, a 50-class canonical environmental sound classification dataset [123]. The second is a subset of AudioSet [126]; the full ontology of classes is very large (over 500). We consider classes from "sounds of things" given that this category contains the most sub-classes and sub-selected the top 50 classes by number of annotations, removing duplicates or equivalent

classes. We use a pretrained Audio Spectrogram Transformer (AST) model for AudioSet-50, and fine-tune an AST for ESC-50 classification [127]. When evaluating on AudioSet-50, we mask the remaining logits to effectively make it a 50-class classifier.

SYNTHESIS QUALITY A significant benefit of our approach is synthesizing clean audio using signal generators while keeping attributes like sample rate flexible. We find synthesized sounds also often exaggerate aspects of the prompts, resulting in large variations in acoustic properties over time. Evaluating audio quality reference-free is challenging, so we examine acoustic features that correlate with these aspects (such as high-frequency content and spectral variation).

USER STUDY We conduct a listening test with human evaluators. We ask them to classify sounds, rate their confidence, and rate sounds along a scale from realistic portrayal to artistic interpretation. This offers us the most direct signal of our abstraction-related goal. We share details on this study in the next subsection. We compared against the recent neural generation methods *AudioLDM* [16] and *AudioGen* [15].

From our 50-prompt subset of AudioSet [126] classes, we randomly selected 10 for this study. We used text embeddings of the labels with a facility location submodular optimization algorithm from the apricot package [128] to select a modest-sized semantically representative subset. Within each prompt, we randomly sampled two of 10 available *CTAG* sounds. The prompts were: *Truck air brake*, *Water tap*, *Train horn*, *Motorcycle*, *Microwave oven*, *Liquid slosh*, *Chainsaw*,

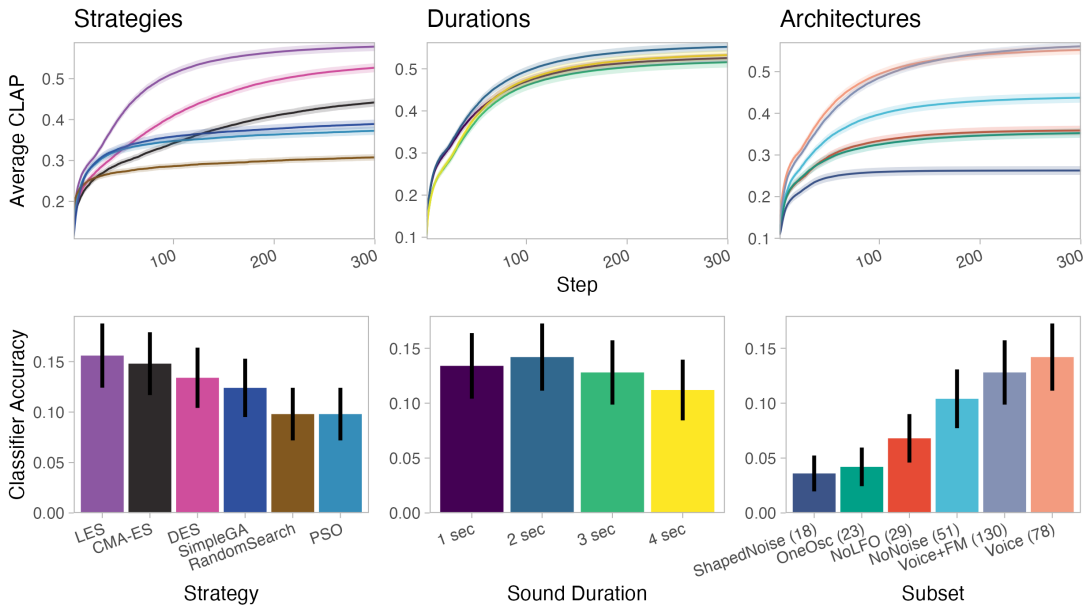

Figure 7: Results from our ablation study; all experiments are conducted with ESC-50. **(Top)** CLAP similarity maximization curves, averaged across 10 iterations for each of the 50 prompts. Colored bands show 95% confidence intervals. **(Bottom)** Classification accuracy, with error bars showing 95% confidence intervals. Top and bottom plots share colors. **(Left)** Performance of different algorithms, with hyperparameters tuned on ESC-10. LES is strongest in both optimization and downstream classification. **(Center)** Different sound durations; we find 2 seconds to be strongest. **(Right)** Impact of synthesizer architecture, finding strongest results from the *Voice* model. Parameter counts are given in parenthesis, such as (78) for *Voice*.

Airplane, *Bicycle bell*, and *Machine gun*. For *AudioLDM* and *AudioGen*, we used their default parameters to generate two sounds per prompt.

This study was determined to be exempt by our institution's IRB. Each participant rated 60 sounds (20 per method) in random order. To examine category-level recognition, participants were asked to select one category given a list of ten options and rate their confidence. To determine whether our generated sounds were perceived as (abstract) artistic interpretations, we posed the question: "Would you associate this sound more with a realistic portrayal or

an artistic interpretation of the label that you selected?" with options on a scale from 1 (realistic portrayal) to 5 (artistic interpretation). We modeled participant responses with mixed-effects logistic and linear regression models and post-hoc contrasts.

4.3 RESULTS

4.3.1 Ablation Studies

Figure 7 shows results from our ablation studies, including, from left to right, (1) optimization algorithms with tuned hyperparameters, (2) sound durations, and (3) synthesis architectures. Overall, we observe that the LES algorithm significantly outperforms our other options within the computation budget of 300 iterations (more than needed for several prompts). This experiment was conducted with 2-second long sounds, which we observe in the *Durations* experiment to yield a higher overall CLAP score and classification accuracy than 1, 3, or 4-second long generations. Finally, we see that the *Voice* architecture yields the best results, offering a balance of flexibility in its parameters and modular structure, as well as ease of optimization. However, we note that expanding to larger architectures like VoiceFM could be useful for future work to explore, with more work on the optimization strategy to obtain the best results.

Based on these results, we conduct all additional experiments discussed with the LES optimizer, 2-second sounds, and the *Voice* architecture. We conducted a full hyperparameter tuning run with 50 trials of all ESC-50 prompts to obtain the final optimization hyperparameters.

4.3.2 Qualitative Results

Examples

Figure 5 shows spectrograms of synthesized sounds corresponding to six text prompts. The "spray" shows bands of noisy bursts, reflecting the short, sharp sound of aerosol being expelled. The "bees buzzing" presents a band of low to high frequencies, encapsulating the vibrant hum of a bee. The "police car siren" is characterized by high-frequency oscillations that sharply rise and fall. The "machine gun" reveals rapid, staccato bursts of energy across a broad frequency range. The "train horn" displays horizontal bands across mid to high frequencies, illustrating the horn's fundamental tone and its partials. Lastly, the "chainsaw" spectrogram is dominated by intense, continuous mid-range frequencies, punctuated by peaks corresponding to the engine's roaring and cutting action.

Interpolation

In sound synthesis, interpretable parameters offer a unique opportunity for deeper insight. Our method provides a fixed set of parameters that possess this property—a salient distinction from contemporary models equipped with

high-dimensional latent spaces. This interpretability extends to interpolation between parameters of distinct sounds, granting auditory access to intermediate acoustical transitions. In Figure 8, we present a systematic series of spectrograms between pairs of prompts: (1) "Spray" to "Machine gun", (2) "Train horn" to "Chainsaw", and (3) "Train wagon" to "Engine revving," with three intermediary steps linearly interpolated. This discernible gradation corroborates the capacity of our parameter space to retain congruence.

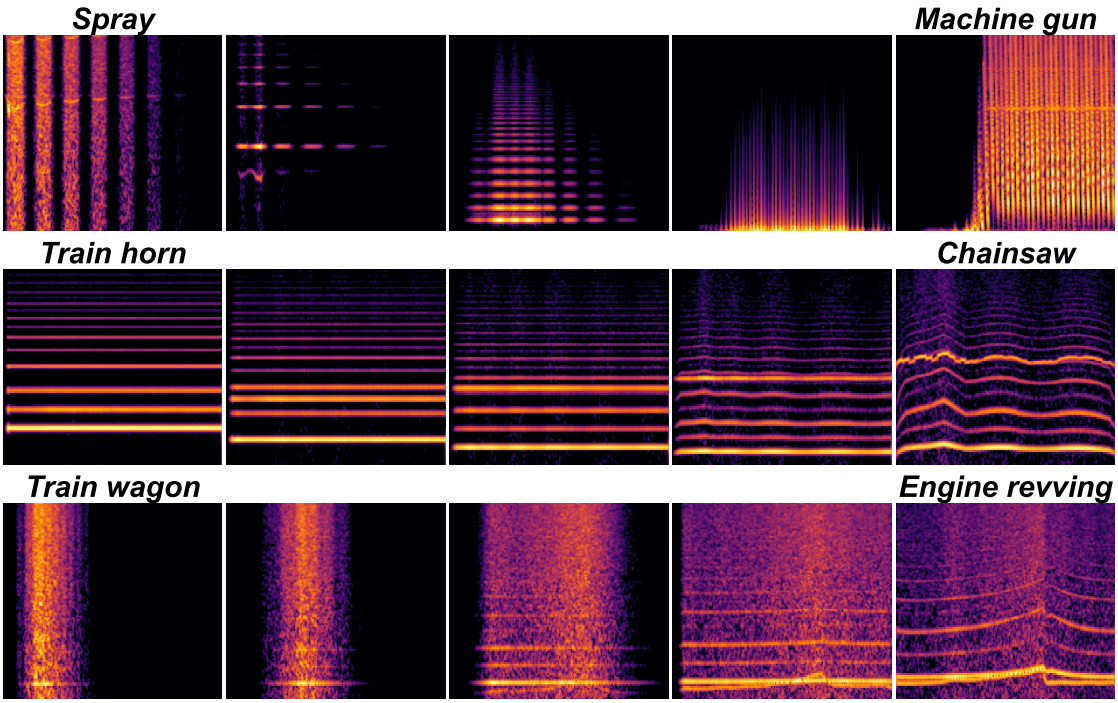

Figure 8: Spectrogram series as the result of linear interpolation of the synthesizer's parameters (1) from "Spray" (left) to "Machine gun" (right), (2) from "Train horn" to "Chainsaw", and (3) from "Train wagon" to "Engine revving". Each spectrogram in the sequence represents a step in the interpolation, highlighting the systematic shift in acoustic properties.

4.3.3 Classification Results

Results are shown in Table 2. On AudioSet-50, our results are higher than *AudioLDM*. On ESC-50, the classifier recognizes *CTAG*'s sounds the least, showcasing the distribution shift from its training on realistic sounds. We experimented with constructing concise and descriptive prompts from each sound class from both ESC-50 and AudioSet-50. We used GPT-4 [129] to automatically produce caption-style prompts. We also tried a simple template (i.e. "Sound of a/an ...") to compare. Table 2 also shows results for these template (*CTAG+T*) and caption-style prompts (*CTAG+C*). Introducing such strategies does not appear to greatly influence classifier identification. However, in a few cases, we observed the elaborated prompts helped to produce qualitatively more accurate results. Overall, *CTAG* sounds are classified correctly significantly higher than chance, and competitively with *AudioLDM*.

4.3.4 Synthesis Quality and Variation

Evaluating the quality of generated examples is challenging for two reasons. First, we lack auditory references to compare against, as we generate from text directly and never use text-audio reference pairs. Most audio quality metrics are reference-based. Second, distance-based metrics such as the Fréchet Audio Distance (FAD) will likely be confounded by realism. *CTAG*'s sounds are high-quality in that they can be generated at high sample rates and are free of noise or artifacts owing to real-world recording environments or neural synthesis.

To evaluate, we use auditory descriptors (implemented using Essentia [130]) that are plausible correlates of these notions of quality, shown in Table 1. Spectral complexity highlights the presence of more peaks, signaling diversity in the timbral components, while flux shows greater variation of timbre over time for *CTAG* compared with other methods. Following these, HFC (high-frequency content), spectral rolloff, and spectral centroid provide signals of "brightness" or high-frequency presence in the sounds. All of these results show our method's ability to introduce high-frequency content into generated sounds, likely in part due to the higher sample rate we use.

	AudioSet-50			**ESC-50**		
	AudioGen	*AudioLDM*	*CTAG*	*AudioGen*	*AudioLDM*	*CTAG*
Complexity	16.50	17.65	18.06	9.60	12.94	17.76
Flux	0.08	0.11	0.18	0.06	0.09	0.15
HFC	53.25	152.06	427.03	34.49	101.32	380.74
Rolloff	2,487.71	1,628.55	7,031.67	2,254.98	1,647.51	6,996.19
Centroid	1,629.95	1,096.16	4,139.99	1,512.55	1,108.42	4,227.08
Comp. Ratio	6.42	7.09	9.51	6.46	7.58	9.57

Table 1: Comparison of spectral descriptors—complexity, flux, HFC, rolloff, centroid—and audio compression ratio, across ESC-50 and AudioSet-50. Results are grouped by the evaluation of three methods: *AudioGen*, *AudioLDM*, and our method, *CTAG*.

We also report compression ratio, under variable bit rate (VBR) MP3 compression (quality = 4). Interestingly, *CTAG* achieves a higher average compression ratio. VBR generally works by applying lower ratios to more perceptually complex input. Whether related to high-frequency content or other factors, this suggests *CTAG* sounds contain more perceptual redundancy or are perceptually "simpler".

Note that none of these measures are validated as perceptual metrics of audio quality, and we do not intend to use them as such. Rather, they help us to quantify the qualitative differences we observe between *CTAG*-synthesized sounds and other text-to-audio generation models' results.

Dataset	**Model**	**Top-1 Acc.**	**Top-5 Acc.**
AudioSet-50	*AudioGen*	51.6	77.4
	AudioLDM	17.4	44.2
	CTAG	26.2	45.2
	CTAG+T	25.2	52.2
	CTAG+C	23.6	51.6
ESC-50	*AudioGen*	54.0	71.8
	AudioLDM	23.0	49.4
	CTAG	16.4	30.4
	CTAG+T	11.4	26.4
	CTAG+C	13.8	31.0

Table 2: Top-1 and Top-5 classification accuracies (%) for pre-trained classifiers with AudioSet-50 and ESC-50. We evaluated both models on results collected using *AudioGen*, *AudioLDM*, and our method with just the class labels (*CTAG*), a simple template (i.e. "Sound of a ...") for each sound (*CTAG+T*) and finally using an LLM for prompt engineering (*CTAG+C*).

4.3.5 User Study

We recruited 10 participants via Prolific at $12/h for a total of $53.33, resulting in a total of 600 observations per outcome variable (i.e. accuracy, confidence, and artistic interpretiveness). Table 3 contains the results, which show that our sounds were identified by listeners substantially more accurately than those from *AudioLDM* (odds ratio = 2.72, 95% CI [1.61, 4.58], $p < .0001$), and only slightly less than *AudioGen* on average (odds ratio = 0.85, 95% CI [0.51, 1.42],

$p = 1$). Interestingly, though the confidence ratings replicate the ordering of the accuracy results, respondents were significantly more confident rating *AudioGen* sounds, and reported similar, lower confidence levels for both *CTAG* and *AudioLDM*. This underscores the abstractness of *CTAG*'s sounds; despite being identified more correctly, they still create uncertainty.

	AudioGen	*AudioLDM*	*CTAG*
Accuracy	59.5	34.0	56.0
Confidence	3.48	2.95	2.99
Artistic Interpretation	2.32	2.90	3.54

Table 3: User study results for sounds from *AudioGen*, *AudioLDM*, and our method, *CTAG*. We report accuracy percentage and confidence (1–5) on label identification, and average rating of the artistic interpretiveness (1–5) of the sound. Overall, *CTAG* retains competitive identifiability while being perceived as more artistic.

Results also show *CTAG* sounds were perceived to be significantly more artistically interpretive than both *AudioGen* (contrast = 1.22, 95% CI [0.93, 1.51], $t(579) = 10.20$, $p < .0001$) and *AudioLDM* (contrast = 0.65, 95% CI [0.36, 0.93], $t(579) = 5.39$, $p < .0001$).

These findings highlight our approach's benefits in capturing artistic interpretation compared to both the existing approaches. All p-values are Bonferroni-adjusted. Full results for post-hoc contrasts are available in the Appendix.

4.3.6 Additional Analyses

In Section 4.6 we provide additional analyses relating to generation time, CLAP scores, prompting strategies for the baseline models, user study results, and a visualization of the parameter space of *CTAG*-generated sounds.

4.4 LIMITATIONS

Our method requires iterating for each prompt from random initialization, but techniques like semantic caching to initialize to similar prompts' parameters, predictive methods for prompt-to-parameter derivation, and a user interface extension for tweaking parameters are all potential extensions to make our method more useful in real-world settings. We also focus on brief, non-mixture sounds as these are what the synthesizer is suited to modeling. Future work could explore strategies to extend the duration and complexity of sounds that can be synthesized this way.

4.5 CONCLUSION

In this work, we proposed a method for text-to-audio generation that offers a fresh perspective on neural audio synthesis by using a virtual modular synthesizer. This approach emphasizes the meaningful abstraction of auditory phenomena, contrary to prevalent methods that prioritize acoustic realism. Our results position this approach as a distinctive tool in the field of audio synthesis, capable of both expanding the toolkit of novices and experts, and stimulating new directions in audio generation research.

4.6 SUPPLEMENTARY ANALYSES

4.6.1 Generation Time

Iter/Popsize	25	50	100
50	5.49 ± 0.154	9.62 ± 0.452	18.43 ± 0.752
100	10.01 ± 0.194	18.05 ± 0.605	33.40 ± 0.331
300	27.61 ± 0.703	49.94 ± 0.424	97.23 ± 0.469

Table 4: Time (in seconds) for different population sizes (columns) and iteration counts (rows).

In Table 4 we illustrate the optimization times, in seconds, for different numbers of iterations (rows) and optimizer population sizes (columns) below, on a modest GPU, i.e. single V100. Note that the necessary number of iterations varies for different prompts, from 50 to 300+ to get optimal results.

4.6.2 CLAP Scores

Model	AudioSet-50	ESC-50
AudioGen	0.249 ± 0.160	0.277 ± 0.180
AudioLDM	0.166 ± 0.128	0.173 ± 0.142
CTAG	**0.573** ± 0.126	**0.585** ± 0.130
Real	–	0.416 ± 0.139

Table 5: Comparison of CLAP scores between *CTAG* and other generative models on AudioSet-50 and ESC-50 datasets

Table 5 shows the CLAP [2] evaluations for each model with AudioSet-50 and ESC-50 prompts, as well as for the actual ESC-50 dataset of real sounds.

CLAP is the objective that we optimize (see Equation (2)) in our synthesis-by-optimization approach, and these results show how *CTAG* trivially achieves a higher score compared to all other models and even the real data. This highlights the ability of our optimization strategy to effectively maximize the CLAP score, and also the importance of finding alternative and distinct evaluation metrics as we showed in Section 4.2.4.

4.6.3 Prompting Strategies for All Tested Models

Dataset	Metric	Model	Sounds	Template	Caption
AudioSet-50	Top-1	*AudioGen*	51.6	**57.0**	48.8
		AudioLDM	17.4	**21.0**	16.6
		CTAG	**26.2**	25.2	23.6
	Top-5	*AudioGen*	77.4	**84.8**	80.8
		AudioLDM	44.2	**49.8**	48.0
		CTAG	45.2	**52.2**	51.6
ESC-50	Top-1	*AudioGen*	54.0	**69.0**	62.0
		AudioLDM	23.0	20.2	**29.4**
		CTAG	**16.4**	11.4	13.8
	Top-5	*AudioGen*	71.8	**85.2**	81.8
		AudioLDM	49.4	47.0	**58.4**
		CTAG	30.4	26.4	**31.0**

Table 6: Performance comparison, with different prompting strategies, of models on AudioSet-50 and ESC-50 datasets

For completeness, Table 6 provides all the results for all different models with templates and captions as we showed for *CTAG* in Section 4.3.3. The performance of *AudioGen* shows a notable boost when using the +T (Template) strategy. However, the impact of these strategies on the other models and

datasets is less consistent, with some cases showing modest improvements and others exhibiting a decrease in performance (e.g., *AudioLDM* ESC-50 +T, *AudioLDM* AudioSet-50 +C). Given the variability in results, it is difficult to make a definitive statement about the effectiveness of these strategies across all baselines. While they may prove beneficial in certain scenarios, their impact appears to be context-dependent.

4.6.4 User Study Statistical Models

We report post-hoc contrasts for the user study results in Tables 7 to 9.

contrast	odds.ratio	SE	asymp.LCL	asymp.UCL	z.ratio	p.value
AudioLDM / AudioGen	0.31	0.07	0.19	0.53	-5.28	<1e-04
CTAG / AudioGen	0.85	0.18	0.51	1.42	-0.75	1
CTAG / AudioLDM	2.72	0.59	1.61	4.58	4.59	<1e-04

Table 7: Post-hoc contrasts from a mixed-effects logistic regression for accuracy.

contrast	estimate	SE	df	lower.CL	upper.CL	t.ratio	p.value
AudioLDM - AudioGen	-0.53	0.12	579	-0.82	-0.24	-4.34	<1e-04
CTAG - AudioGen	-0.48	0.12	579	-0.78	-0.19	-3.97	0.00024
CTAG - AudioLDM	0.04	0.12	579	-0.25	0.34	0.37	1

Table 8: Post-hoc contrasts from a mixed-effects linear regression for confidence ratings.

contrast	estimate	SE	df	lower.CL	upper.CL	t.ratio	p.value
AudioLDM - AudioGen	0.57	0.12	579	0.29	0.86	4.81	<1e-04
CTAG - AudioGen	1.22	0.12	579	0.93	1.51	10.20	<1e-04
CTAG - AudioLDM	0.65	0.12	579	0.36	0.93	5.39	<1e-04

Table 9: Post-hoc contrasts from a mixed-effects linear regression for artistic interpretativeness.

4.6.5 User Study Per-Prompt Accuracy

Figure 9 shows the accuracy of our user study participants at classifying sounds generated with *CTAG*, *AudioGen*, and *AudioLDM*. Reviewing these differences shows that some sounds are overall more difficult to identify, for instance; "Truck air brake". This may be due to the ambiguity in what this can sound like, as it is not as common a sound as "Bicycle bell".

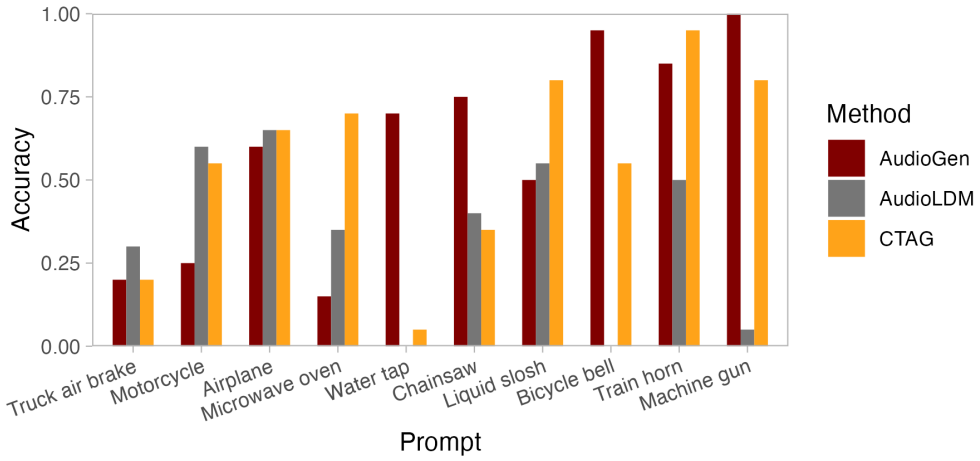

Figure 9: User study classification accuracy per prompt, for *CTAG*, *AudioGen*, and *AudioLDM*.

4.6.6 Dimensionality Reduction

Having access to the parameters of the synthesizer also allows us to project them into a two-dimensional space to explore the relationship between sounds. Leveraging the Uniform Manifold Approximation and Projection (UMAP) [131] algorithm for dimensionality reduction of the synthesizer parameters, Figure 10 shows how the representation delineates clusters for each distinct sound class

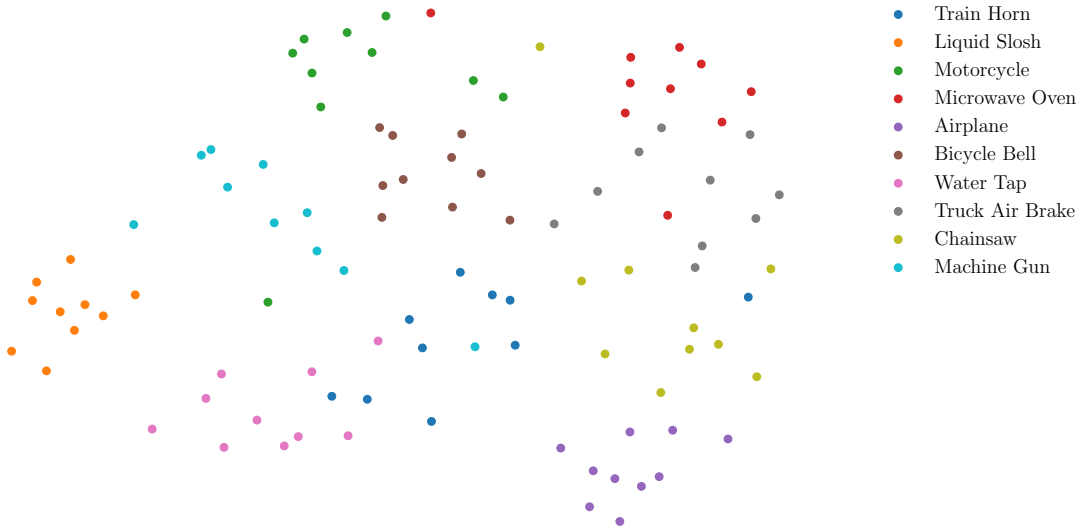

Figure 10: Dimensionality reduction of the *Voice* synthesizer parameters using UMAP applied to 10 sounds from each of the 10 classes from the user study. It distinctly reveals clusters corresponding to individual sounds, and it shows how conceptually similar sounds such as "water tap" and "liquid slosh" are closer in space.

while retaining semantic meaning—sounds with similar acoustic properties cluster together.

4.6.7 Caption Prompt

We used the following instructions to generate caption-like prompts from class labels:

> *"Write a simple one-sentence audio caption that describes objectively each sound itself in a real scenario without making up any extra details about other possible sounds or places. You should define the most common*

action for such an entity when multiple options are available. Avoid using templates such as 'A sound of' or 'The sound of'. Sounds: [List]"

This method results in prompts such as "A basketball bounce resonates with a rhythmic thud as it hits the floor", "An engine idling gives off a steady, low rumble, signifying readiness but inactivity", or "A printer emits a series of mechanical whirs and clicks as it processes a document".

4.6.8 Listener Survey

In this section, we provide information about the survey design we used to collect human ratings.

Survey Flow

- Standard: Introduction (3 Questions)
- Block: Audio (4 Questions)
- Standard: Additional (2 Questions)

Start of Block: Introduction

Q1: We are conducting a survey to assess the quality of a novel method for text-to-audio generation. You will be presented with a series of short sounds, and asked to select the closest category from a given list, the confidence in your prediction, and how artistically designed the sound is compared to a more realistic interpretation.

Q2: I consent to participate. I understand that my participation is voluntary and I may withdraw my consent at any time.

- Yes (1)
- No (2)

Q3: I am at least 18 years old.

- Yes (1)
- No (2)

Q4: Do you have any hearing loss or hearing difficulties?

- Yes (1)
- No (2)

Q5: Are you fluent in English?

- Yes (1)
- No (2)

Q5: What is your Prolific ID? Please note that this response should auto-fill with the correct ID

Start of Block: Audio

We use Qualtrics' Loop & Merge functionality to loop through the sounds.
A: Select the closest category for the following sound: **[Audio Clip]**

- Truck air brake (1)

- Water tap (2)
- Train horn (3)
- Motorcycle (4)
- Microwave oven (5)
- Liquid slosh (6)
- Chainsaw (7)
- Airplane (8)
- Bicycle bell (9)
- Machine gun (10)

B: How confident are you in your selected answer?

- Completely confident (1)
- Fairly confident (2)
- Somewhat confident (3)
- Slightly confident (4)
- Not confident at all (5)

C: Would you associate this sound more with a realistic portrayal or an artistic interpretation of the category that you selected?

- 1 (1) Realistic Portrayal

- 2 (2) •
- 3 (3) •
- 4 (4) •
- 5 (5) Artistic Interpretation

Start of Block: Additional

We have two questions to check that participants were paying attention, which all participants passed.

A1 Please select "Chainsaw" from the options below:

- Truck air brake (1)
- Water tap (2)
- Train horn (3)
- Motorcycle (4)
- Microwave oven (5)
- Liquid slosh (6)
- Chainsaw (7)
- Airplane (8)
- Bicycle bell (9)
- Machine gun (10)

A2: All of the sounds you heard during this survey were the same.

- Yes (1)
- No (2)

Completion Message: Thank you for taking part in this study. Please click the button below to be redirected back to Prolific and register your submission.

5 CONTRASTIVE LEARNING FROM SYNTHETIC AUDIO DOPPELGÄNGERS

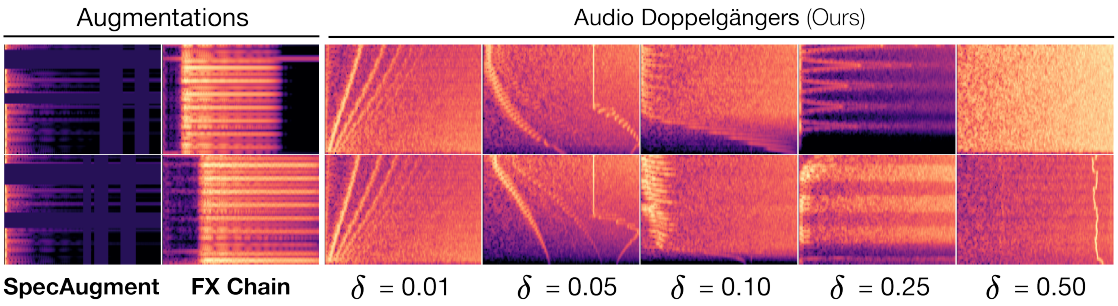

Figure 11: **(Left)** Standard data augmentation techniques for contrastive learning applied to audio spectrograms **(Right)** *Audio Doppelgängers*, our approach synthesizing sounds that are controllably different using perturbed synthesis parameters, shown for different factors δ. These sounds can vary in causally controllable ways beyond what data augmentations can achieve.

5.1 INTRODUCTION

"Noises have generally been thought of as indistinct, but this is not true."

— **Pierre Schaeffer** [132]

The success of modern machine learning algorithms for tasks like audio understanding often hinges on both the quality and quantity of available data. Self-supervised learning methods, like contrastive learning, have even been able to leverage unlabeled data, enabling more human-like learning from patterns

without needing explicit supervision. However, human perceptual processing is remarkably robust beyond this: for example, the human auditory system can easily recognize sounds across a wide range of variations, such as changes in pitch, timbre, or background noise. Moreover, humans can quickly learn to recognize novel sounds that they encounter in their environment. Replicating this ability to learn from a diverse array of sounds—or "noises," as we might call them—could significantly enhance the efficiency, scalability, and adaptability of machine learning models.

Contrastive learning, which operates by recognizing similarities in the data among negative distractors, often relies on augmentations: transformations of input data that preserve content semantics. This method has been influential in audio representation learning, with specific implementations ranging from spectral masking to temporal jitter to cropping and other methods [133, 95, 134, 97, 94, 135, 136]. Data augmentations, though demonstrably useful, operate at the level of the observed data, not the underlying data-generating process as would be observed in real-world variation. They statistically alter data without directly manipulating the causal mechanisms that produced it, resulting in high correlation between augmented samples, as well as limited control and interpretability.

In our work, we propose using a synthesizer to overcome this barrier, in addition to providing the scalability required for modern pretraining regimes through virtually unlimited data synthesis. A synthesizer can be understood as a system where parameters (relating to psychophysical attributes like pitch, timbre, and loudness) causally influence the generated sound. Modifying these parameters allows us to intervene in the data-generating process in a

controllable way to generate positive pairs that vary in terms of their underlying synthesis parameters.

We formulate an approach[1] in which we randomly synthesize sounds, and then slightly perturb their parameters to generate positive pairs. We call these *audio doppelgängers* (examples in Figure 11); they share a resemblance but are in fact distinct enough to learn from the variation between them. In a way, this approach uses an artificial data source effectively consisting of random synthetic noises but more "natural" augmentations akin to variation in similar sounds; as Pierre Schaeffer put it, noises are not indistinct. Through a comprehensive set of experiments, we show that models trained this way can yield strong performance on a wide range of downstream tasks, competitive with real audio datasets.

Overall, this work contributes:

1. An approach to synthesizing paired audio examples with a continuously controllable degree of dissimilarity, specified by a simple and interpretable hyperparameter δ.

2. The first study, to our knowledge, of synthetic data methods for audio representation learning.

3. Comprehensive experiments in which we train and compare over 20 model variants across 8 downstream tasks to provide evidence that training with our approach can yield strong results on a wide range of audio processing tasks.

[1] doppelgangers.media.mit.edu

4. An analysis of how these synthetic datasets differ from realistic audio datasets in terms of their auditory features, and how this might contribute to learning effective representations.

5.2 METHODS

5.2.1 Data Generation

Our data generation pipeline uses virtual modular synthesizers implemented by SynthAX [3] in JAX. By default, we use the *Voice* synthesizer architecture [1], which has been shown to generate perceptually diverse sounds. This synthesizer has 78 parameters: a monophonic keyboard; two low-frequency oscillators and six ADSR temporal envelopes to generate control signals; sine and square-saw oscillators and a noise generator; and additional amplitude- and modulation-related components. In our experiments, we investigate two further architectures: *VoiceFM* has 130 parameters and includes a frequency modulation (FM) operator, and *ParametricSynth* has 2 sine and 2 square-saw oscillators, 1 sine FM and 1 square-saw FM operator, 340 in total. Varying the architecture allows us to investigate whether architectural complexity could affect the quality of representations learned. We generate 1-second sounds by default, for compatibility with most encoders (e.g. VGGish [137]). However, this practice can be extended to longer sounds.

SYNTHESIS PERTURBATION FACTOR (δ) A key contribution of our work is synthesizing paired positive samples that sound alike, but are dissimilar due to their synthesis parameters and not only post-hoc effects (e.g. augmentations). This draws on the canonical definition from contrastive learning of positives that are sampled from the same *latent class* [138].

For a single positive pair, we first sample a parameter vector uniformly randomly $\theta \in [0,1]^{m_S} \sim \mathcal{U}(0,1)$ from the normalized synthesis parameter space, where $m_S \in \{78, 130, 340\}$ is the number of control parameters in the given synthesizer. Then, we independently sample two isotropic Gaussian noise vectors $\mathbf{z_1}, \mathbf{z_2} \sim \mathcal{N}(0, \mathbf{I}_{m_S \times m_S})$. We define a parameter δ that scales this noise, and then produce two perturbed parameter vectors $\theta_i = \theta + \delta \mathbf{z_i}\ \forall_i \in \{1,2\}$. From these, we clip values back into $[0,1]$ to synthesize two corresponding sounds which serve as positives in the contrastive learning setup.

In principle, δ controls the distance between the positive pairs and therefore the hardness of the contrastive learning task. Practically, we expect there to be a sweet-spot for δ, considering prior work on mutual information and redundancy in contrastive learning problems [139, 140] as with very high δ, the parameter vectors may become dominated by noise, resulting in difficulty effectively aligning their representations. Given this, we extensively study the effect of δ on downstream results.

5.2.2 Real Data

To compare to real audio data, we use sounds from VGGSound [4], a well-known dataset taken from YouTube videos (we only use audio). We use a random sample of 100,000 10-second files and select a random 1-second segment from each file at each iteration to augment. This allows us to fairly compare to our synthetic sounds by keeping duration constant, while still sampling from a variety of real sounds by randomizing the 1-second segments. Though VGGSound has labels included, we do not use them in training these models to keep the self-supervised constraint.

5.2.3 Preprocessing, Data Augmentations, and Audio Encoder

In our experiments, we use VGGish frontend representations [137]. We resample audio to 16kHz and obtain mel spectrograms with 64 mel bands and 96 time steps. We use a chain of effects as augmentations (implemented in torch-audiomentations[2]): a high-pass filter (cutoff frequency range 20–800Hz), a low-pass filter (1.2–8kHz), pitch shift (-2 to 2 semitones), time shift (-25% to 25%, rollover enabled), and finally reverberation for which we sample randomly from a set of impulse responses. All augmentations are applied with probability 0.5. We found that this yielded far stronger results than SpecAugment [141], and so we use this as a comparison point in all our experiments. More details on the augmentation are given in Section 5.6.4. We also test temporal jitter,

[2] https://github.com/asteroid-team/torch-audiomentations

wherein different 1-second segments are sampled from within the same source clip and treated as positives [95, 134]. Our audio encoder is a ResNet18 [142], where we replace the initial layer with a 1-channel convolution to account for the effectively 1-channel spectrogram.

5.2.4 Contrastive Learning

We train for 200 epochs, generating (or sampling) 100,000 sounds per epoch, with a 90%-10% train-validation split. We use a batch size of 768 per GPU with two V100s. The training uses the alignment and uniformity objectives [143] used in prior work on learning with synthetic data [92]. We adopt the default parameters for these: $\text{unif}_t = 2$, $\text{align}_\alpha = 2$, and equal weights $\lambda_1 = \lambda_2 = 1$ for both terms. Following this work, we use stochastic gradient descent for optimization, with a maximum learning rate of 0.72 (calculated as $0.12 \times \frac{\text{total batch size}}{256}$) and weight decay 10^{-6}. The learning rate follows a multi-step schedule with $\gamma = 0.1$, and milestones at 77.5%, 85%, and 92.5% of the total learning epochs. Detailed steps are provided in Algorithm 2. Training with our synthetic data takes approx. 1-2 hours, as the data is generated on the fly in batches, whereas using on-disk datasets with effect chain augmentations can extend training time up to 6-8+ hours.

Algorithm 2 Our contrastive learning procedure with *audio doppelgängers*. In the training loop, we drop the batch index i for simplicity.

Require: Batch size k
Require: Perturbation factor δ
Require: Virtual synthesizer S with $m_S \in \{78, 130, 340\}$ parameters
Require: Embedding model M with embedding size m_M (512 in our case)
Require: Total number of training batches N_{batches}
Require: $\ell_{\text{unif}}(\mathbf{X} \in [0,1]^{k \times m_M}) \leftarrow \log\left[\frac{1}{k^2}\exp\left(-t\sum_{j=1}^{k}\sum_{l=1}^{k}\|\mathbf{X}[j]-\mathbf{X}[l]\|_2^2\right)\right]$
where $t = 2$

for $i = 1$ **to** N_{batches} **do**
  $\mathbf{\Theta} \in [0,1]^{k \times m_S} \sim \mathcal{U}(0,1)$ {Random batch of parameters}
  $\mathbf{Z}_1, \mathbf{Z}_2 \in \mathbb{R}^{k \times m_S} \sim \mathcal{N}(0, \mathbf{I})$ {Isotropic Gaussian perturbation noise}
  $\hat{\mathbf{\Theta}}_1 \leftarrow \max(0, \min(\mathbf{\Theta} + \delta\mathbf{Z}_1, 1))$ {Clipped perturbed parameters}
  $\hat{\mathbf{\Theta}}_2 \leftarrow \max(0, \min(\mathbf{\Theta} + \delta\mathbf{Z}_2, 1))$
  $\mathbf{A}_1 \leftarrow S(\hat{\mathbf{\Theta}}_1)$, $\mathbf{A}_2 \leftarrow S(\hat{\mathbf{\Theta}}_2)$ {Synthesize audio from parameters}
  $\mathbf{E}_1 \leftarrow M(\mathbf{A_1})$, $\mathbf{E}_2 \leftarrow M(\mathbf{A_2})$ {Embedding from model}
  $\mathcal{L}_{align} \leftarrow \frac{1}{k}\sum_{j=1}^{k}\|\mathbf{E}_1[j] - \mathbf{E}_2[j]\|_2^{\alpha}$ where $\alpha = 2$ {Alignment cost}
  $\mathcal{L}_{uniform} \leftarrow \frac{1}{2}[\ell_{\text{unif}}(\mathbf{E}_1) + \ell_{\text{unif}}(\mathbf{E}_2)]$ {Uniformity cost}
  $\mathcal{L}_{total} \leftarrow \lambda_1\mathcal{L}_{align} + \lambda_2\mathcal{L}_{uniform}$ {By default, we set $\lambda_1 = \lambda_2 = 1$}
  Update model M using $\mathcal{L}_{total}$
end for

5.2.5 Evaluation Tasks

To obtain a broad picture of the quality of our learned representations, we conduct experiments on a range of audio classification tasks selected from the HEAR [144] and ARCH [145] benchmarks. We are primarily interested in tasks that focus on real-world sounds (for example, vs. speech or music audio). We use a modified version of the HEAR evaluation setup to conduct linear probing experiments (instead of using an MLP probe) for a more direct signal

of representation quality. We use the Adam optimizer to train this linear probe for the benchmark-specified number of epochs, with the default learning rate of 0.001 and a batch size of 32.

We evaluate on tasks covering a wide range of capabilities including sound classification tasks like ESC-50 [6], FSD-50k [11], and UrbanSound8K [5], vocal affect tasks with and without speech like VIVAE [9] and CREMA-D [8], musical pitch recognition via NSynth Pitch (5h) [10], vocal sound imitation recognition using Vocal Imitations [12], and LibriCount [7] for a "cocktail party" style speaker count estimation task.

5.3 RESULTS

5.3.1 Benchmark Results

In Table 10, we show results across 8 tasks. The top section features external baselines from the HEAR [144] leaderboard and ARCH [145] benchmark results, first the strongest overall and then only self-supervised. It also includes results from MS-CLAP [146] linear probing experiments, GURA [147] (strongest overall model on HEAR), and finally the original ResNet18 trained on VGGSound (supervised) [4]. Note that HEAR leaderboard results may use MLP probes, whereas ours are linear. We add additional internal baselines, including random weights, synthetic data trained without δ but with augmentations, and variants of ResNet18 we trained on VGGSound (with augmentations, and alternately

with temporal jitter). Finally, we include a selection of our results; the best overall score we achieve using our synthetic approach (first row), followed by the best-performing model trained on data from each of the synthesizer architectures (including *Voice* with augmentations). In Section 5.6.3, we provide a full set of results from all model variants: all synthetic datasets for all values of δ, and further baselines less performant than those we present here.

Overall, our best scores uniformly outperform training on VGGSound with augmentations, and outperform training with temporal jitter (the strongest internal baseline) in 6/8 cases. In some cases, these results are also competitive with strong baselines, such as beating the supervised ResNet18 result on 3/8 tasks, CLAP on 2/5, and GURA on 1/6. Additionally, adding further augmentations to our *audio doppelgänger*-based training does not seem to hold significant benefits, despite being highly beneficial when training with synthetic sounds with no δ, suggesting the δ-based perturbations are already sufficiently strong. All this is accomplished without these models seeing any real sounds during pretraining. Finally, *Voice* with $\delta = 0.25$ is the strongest synthetic-trained model overall, being the top performer on 5/8 tasks, but we note that there is some inter-task variability in the best synthesizer and delta.

5.3.2 Characterizing the Data Distribution

Here, we focus on understanding the distribution of synthetic sounds and how they differ from natural sound properties. We primarily use our alternate training set, VGGSound [4], for these measures. Unless specified otherwise, we

Data/Model	ESC	US8K	VIV	NSyn	C-D	FSD	VI	LCount
External Baselines								
HEAR/ARCH Top [144, 145]	96.65	79.09	44.28	87.80	75.21	65.48	22.69	78.53
HEAR/ARCH SSL	80.50	79.09	44.28	52.40	75.21	50.88	18.48	78.53
MS-CLAP Linear [146]	89.95	82.29	–	–	23.15	50.24	–	54.51
GURA (HEAR) [147]	74.35	–	–	38.20	75.21	41.32	18.48	68.34
VGGSound Sup. [4]	87.45	77.57	39.38	43.80	54.36	43.76	14.06	56.10
Internal Baselines								
Random Init.	22.45	55.03	33.81	36.20	38.91	9.03	2.43	44.91
Voice (Ours, No-δ, Aug.)	48.65	59.46	36.31	32.80	46.32	16.88	7.12	47.64
VGGSound SSL (Aug.)	48.85	61.91	32.67	39.60	47.86	19.63	6.03	53.46
VGGSound SSL (Jitter)	52.95	63.82	38.12	14.20	**50.03**	24.02	3.43	**69.77**
Audio Doppelgängers (Ours)								
Best Synthetic	**58.90**	**66.71**	**39.45**	**44.40**	**48.43**	**24.12**	**9.15**	**58.60**
Voice ($\delta = 0.25$)	**58.90**	**66.71**	**39.45**	32.20	48.24	**24.12**	**9.15**	52.95
Voice ($\delta = 0.25$, Aug.)	58.75	65.01	34.81	**44.40**	46.17	21.76	8.54	50.70
VoiceFM ($\delta = 0.25$)	57.20	65.11	38.48	35.20	48.43	22.15	6.96	54.00
Parametric ($\delta = 0.25$)	50.55	62.83	37.91	37.60	46.77	18.68	5.70	54.72

Table 10: Evaluation results on a suite of tasks including (from left to right) ESC-50 [6], UrbanSound8k [5], VIVAE [9], NSynth Pitch 5h [10], CREMA-D [8], FSD50k [11], Vocal Imitation [12], and LibriCount [7]. For internal baselines, we only bold tasks where the baseline beats the best synthetically trained result. Results for all synthetic variants are in Section 5.6.3.

use a randomly sampled (for VGGSound) or generated (for synthetic) set of 1000 sounds for each given dataset used for these characterizations. Our goal is to help understand what properties of the synthetic data make it useful for representation learning, given its strong performance.

Embedding Similarity

First, we look at the distribution of synthetic sound pairs and establish that δ meaningfully controls (a proxy for) perceptual or semantic dissimilarity. We operationalize this using LAION-CLAP embeddings [2], since they are trained on a large variety of sounds with semantic descriptions associated.

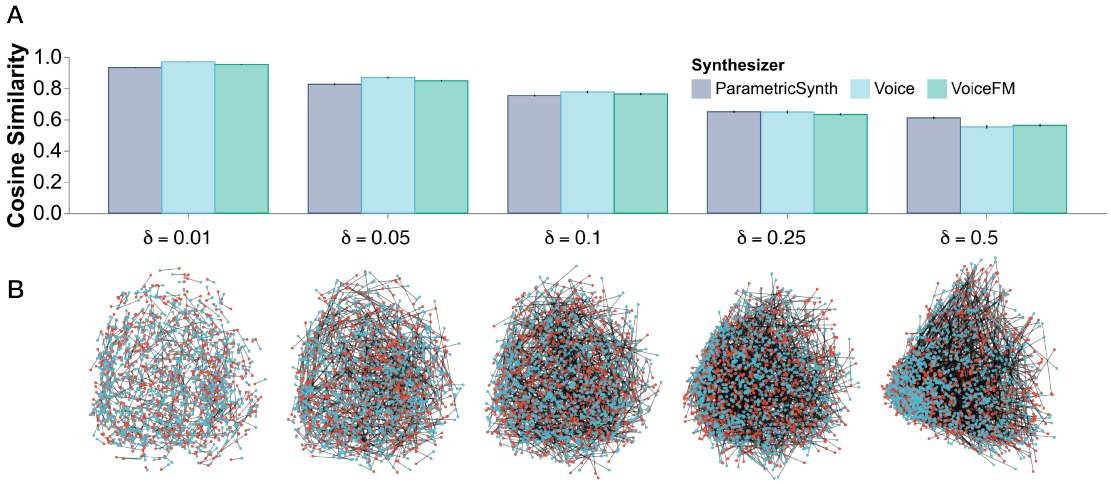

Figure 12: **(A: Top)** Average CLAP [2] embedding cosine similarity between positive pairs for different architectures and different values of δ. **(B: Bottom)** PCA of CLAP embeddings for sounds generated with the *Voice* architecture, with line segments showing distances between paired examples. Red and blue points are paired positive instances. Across both plots, as δ increases, the positive pairs systematically become more perceptually dissimilar (via the CLAP embedding proxy).

Figure 12A shows how the average cosine similarity decreases monotonically with increasing δ for all 3 synthesizers. Figure 12B provides an alternate view of the space of positive pairs, compared to negatives (other samples). Here, we plot the first two principal components of the CLAP embeddings along with the path length for each positive pair of synthesized samples from a *Voice* synthesizer. As δ increases, the path lengths increase and overlap more resulting in less clear separation of positive pairs from negatives. We view this as a signal that we can effectively control the hardness of the contrastive task using δ, the perturbation factor.

Similarities and Differences from Real Data

Next, we compare the synthetic data distribution to VGGSound [4] data. Figure 13A compares a selection of features' distributions between several dataset variants. For synthetic datasets, we have *Voice*, *VoiceFM*, and *ParametricSynth* variants. For real datasets, we have VGGSound. We first compare to randomly sampled 1-second chunks. Here, the synthetic sounds match several feature distributions well, such as Inharmonicity [148], Odd-to-Even Harmonic Ratio [149], Pitch Salience [150], and, to a lesser extent, Spectral Flatness [148]. However, the synthetic sounds have higher Spectral Flux [151] and Complexity [152]. Note that *ParametricSynth* also has lower pitch salience. We believe this is due to its larger mixture of sound generators which reduce salience of particular pitches.

Based on these results, we hypothesize that one potential reason the synthetic sounds could be useful for training is the informativeness of the samples. The larger amount of spectral change and higher complexity in terms of peaks could expose the model to more different kinds of sounds rapidly. To try to match these attributes, we produce mixtures of VGGSound, since mixed sounds may have more peaks and variation than individual samples. In VGGSound-Mix 5s, we take 5 arbitrary seconds from each sound and layer them into a 1-second sample. In VGGSound-Mix 10s, we do the same with all 10 seconds available. We show (Figure 13) that these get closer to the synthetic distributions on these features, without deviating on other features. These data distributions allow us to assess whether the benefits of synthetic data are largely driven by the change and informativeness of the signals. In Section 5.6.3, we present results from models trained on these mixture distributions, and obtained mixed results,

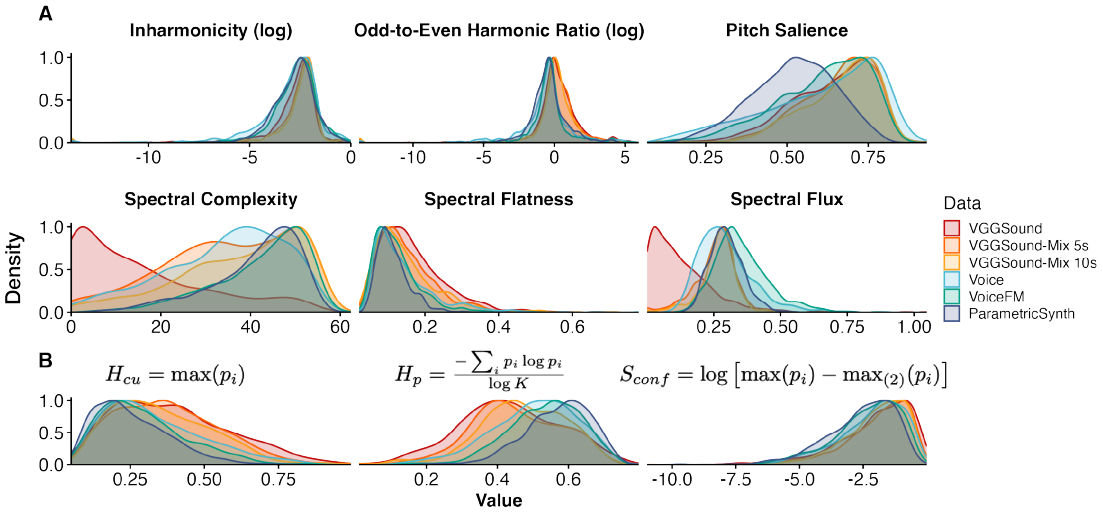

Figure 13: Comparisons of synthetic and real sound data (VGGSound [4]) on **(A: Top)** spectral features and **(B: Bottom)** causal uncertainty. Spectral features of synthetic sounds partially replicate real sounds, but exhibit differences in complexity and flux. Synthetic sounds are also more causally ambiguous, indicating a distribution shift. Using dense mixtures of real sounds partially closes these gaps, suggesting the synthetic sounds are different in part due to their density of auditory information.

suggesting other factors of the synthesized sounds may also be important beyond this.

Causal Uncertainty

We also consider causal uncertainty [153, 154, 155], a factor that we intuitively expect to be different for the synthetic sounds. Helmholtz famously discussed perception in terms of unconscious causal inference from sensory input [156], but the synthetic sounds have no physical causes and do not come from well-understood categories. In Figure 13B, we plot 3 proxies for causal uncertainty derived from probabilities of an AST classifier [157] trained on AudioSet. We use the formulation from prior work of H_{cu}, the maximum predicted probabil-

ity [154, 155]. We also propose two simple metrics to corroborate this: H_p the (normalized) entropy of the output probability distribution, and a confidence score S_{conf}, the difference in probability between the most and second-most probable classes (log-scaled for the plot). Across all, the synthetic sounds are more causally uncertain than the real sounds. However, as with the spectral feature distributions, using mixtures of VGGSound [4] clips moves the real distribution slightly closer to the synthetic distribution per H_{cu} and H_p. We speculate that exposure to more causally uncertain sounds might be subtly helpful for representation learning; for example, they may contain diverse features that aid generalization to more ambiguous sounds present in downstream tasks. We characterize this as another important distributional difference between the synthetic sounds and realistic sounds from datasets such as VGGSound.

Similarity to Target Distributions

Another lens we can use to understand the effectiveness of training on synthetic data is in terms of the distribution of sounds in the target downstream tasks. A common metric to compare sound distributions is the Fréchet Audio Distance (FAD) [13]. For simplicity, we use the canonical formulation based on VGGish embeddings, though there are some limitations of this [158, 159], and we use either the validation sets or first multi-fold splits of the target task audio. Table 11 shows that for ESC-50 [6], VGGSound is closer in distribution to the target sounds, likely due to ESC-50's focus on environmental sounds. For all other tasks, however, the synthetic sounds achieve a lower FAD, suggesting they may better capture task-relevant features for these tasks' sounds. This finding echoes a study of MMDs in torchsynth [1], where the *Voice* architecture shows

higher-than-expected similarity to FSD50k sounds. We hypothesize that the synthetic training allows the model to see a wide variety of spectral behavior rapidly, in a way that supports an array of tasks.

Dataset	ESC-50	FSD50k	LibriCount	NSynth	CREMA-D	Vocal Imitation
Voice	17.39	**13.37**	**16.67**	**12.83**	**18.55**	**11.64**
VoiceFM	18.48	15.91	17.67	14.49	21.24	13.66
ParametricSynth	18.75	19.44	21.04	17.42	25.33	17.32
VGGSound	**6.71**	25.33	29.09	27.67	33.83	27.75
VGGSound-Mix 5s	8.81	26.17	30.02	28.70	34.35	29.05
VGGSound-Mix 10s	9.30	26.09	30.15	28.88	34.06	29.16

Table 11: FAD [13] scores between different synthetic/real datasets and target downstream task data distributions, computed using either validation sets or the first fold (for multi-fold datasets). For 5/6 tasks, *Voice* achieves the lowest FAD despite containing synthetic sounds. On ESC-50, however, the VGGSound distribution appears to be closest.

5.3.3 Ablations and Sensitivity Analysis

In Figure 14, we study the effect of the perturbation factor δ on downstream task performance across all tasks for the strongest (*Voice*) architecture with and without additional (FX chain based) augmentations. Overall, $\delta = 0.25$ appears to give the best result across tasks, with a few notable exceptions like NSynth without augmentations, and LibriCount overall.

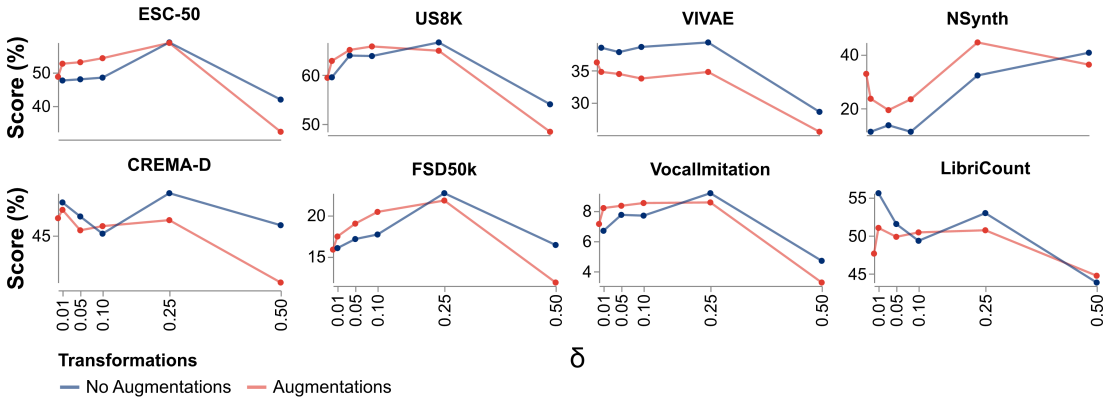

Figure 14: Scores with the *Voice* architecture and different values of δ for evaluation tasks in Table 10 with and without augmentations. $\delta = 0.25$ tends to give the best results overall.

5.4 LIMITATIONS

We opted for small-scale, established architectures like ResNets due to computational constraints and the desire for generalizable results. Scaling up to larger architectures like AST [157] could further validate our approach in high-compute environments (our attempts to train such models were infeasible). Additionally, we focused on direct comparisons between synthetic and real data, excluding hybrid approaches that might offer valuable insights, since the design space of hybrid approaches (e.g. mixtures or fine-tuning) is large and warrants a deeper exploration.

Our simple isometric Gaussian noise perturbation proved effective, but it relies on a simplifying independence assumption (parameters are correlated within components and component chains) and doesn't account for parameter semantics (e.g. octaves in pitch). Future work could explore fine-grained, semantically-aware perturbations operationalized via the covariance structure

of the perturbations. Scheduling or adapting the perturbation factor during training could offer more challenging examples. It could also be learned, which would require stable end-to-end auto-differentiation (challenging depending on the synthesizer architecture). Our evaluation focused on standard classification benchmarks, but incorporating aspects like representation disentanglement could offer a richer view. Finally, while we explored various synthesis options, the possibilities are vast, and future work could investigate variable architectures for enhanced data augmentation.

On a broader note, we believe it's important to examine synthetic data-generating procedures for possible biases in the way that we do with datasets. Though we think this procedure can mitigate some of the biases in collectible real datasets, different synthesizer architectures, values of δ, and other decisions might inadvertently produce performance gaps for different tasks, applications, and downstream populations of use. We evaluated on a wide range of tasks in part to explore this possibility, but further evaluations would be helpful to assess these impacts.

5.5 CONCLUSION

Further improvements in auditory understanding depend greatly on the data underlying new models. In this work, we examined the value of synthetic data for learning representations of sound. We presented a method that perturbs the parameters of random synthetic sounds to generate *audio doppelgängers*, distinct

yet similar sounds that provide a strong signal for contrastive learning. Through a comprehensive set of experiments, we showed how this approach can yield strong results on a wide range of tasks. We will release our code and models to enable the community to experiment with synthetic data sources for audio understanding, and hope this approach will help expand the machine learning toolkit for audio processing.

5.6 SUPPLEMENTARY ANALYSES

5.6.1 Comparison of Different Architectures Across Tasks

In Figure 15 we show the relative performance of models trained with data from different synthesizer architectures with $\delta = 0.25$. These results illustrate that, though *Voice*-generated sounds appear strongest overall, there is some task specialization of these different synthesis approaches. For example, on LibriCount and NSynth, *Voice* is the lowest performer here.

5.6.2 Effects of Increasing Perturbation Factor δ on Training

We seek to understand how increasing δ impacts the training dynamics. Figure 16 shows the impact of different δ on the final validation value of the alignment and uniformity costs respectively. Alignment cost increases monotonically with δ, which shows the increased difficulty of aligning increasingly

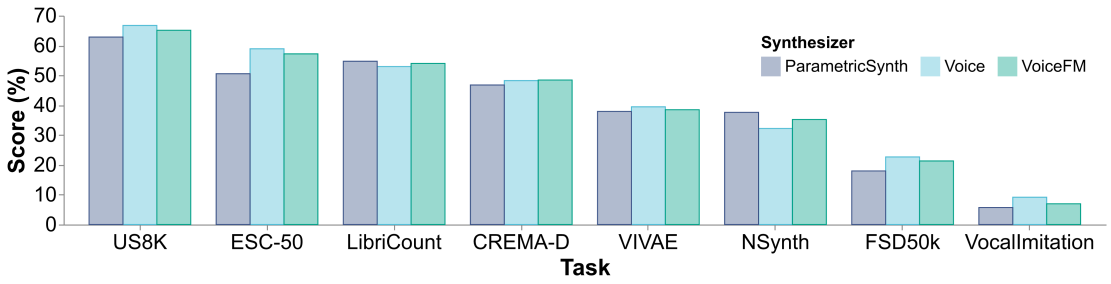

Figure 15: Scores with a fixed $\delta = 0.25$ and different synthesizer architectures for a suite of tasks including (from left to right) UrbanSound8k [5], ESC-50 [6], LibriCount [7], CREMA-D [8], VIVAE [9], NSynth Pitch 5h [10], FSD50k [11], and Vocal Imitation [12]

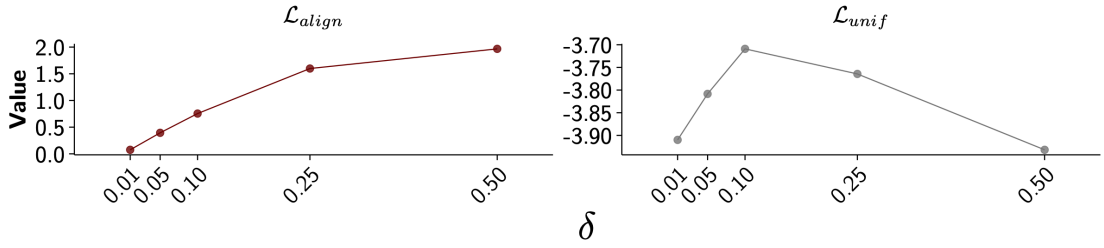

Figure 16: Final validation scores showing the effect of δ on $\mathcal{L}_{align}$ and $\mathcal{L}_{unif}$. $\mathcal{L}_{align}$ increases monotonically with δ, since the difficulty of aligning more distinct samples goes up. $\mathcal{L}_{unif}$, on the other hand, shows an inverse-U-shaped relationship with δ.

distant pairs. Uniformity, interestingly, has an inverted-U-shaped relationship with δ. This may be because as the model struggles to align positives with moderate noise driven variation, it incurs a cost in uniformity in order to do so (e.g. creating clusters). With large δ, the amount of noise present is significant, alignment is difficult, and the representations may be able to be more spread out.

5.6.3 Results for all Variants

We give results for all synthetic model variants below, in Table 12.

Data/Model	ESC	US8K	VIV	NSyn	C-D	FSD	VI	LCount
External Baselines								
HEAR/ARCH Top [144, 145]	96.65	79.09	44.28	87.80	75.21	65.48	22.69	78.53
HEAR/ARCH SSL	80.50	79.09	44.28	52.40	75.21	50.88	18.48	78.53
MS-CLAP Linear [146, 160]	89.95	82.29	–	–	23.15	50.24	–	54.51
GURA (HEAR) [147]	74.35	–	–	38.20	75.21	41.32	18.48	68.34
VGGSound Sup. [4]	87.45	77.57	39.38	43.80	54.36	43.76	14.06	56.10
Internal Baselines								
Random Init.	22.45	55.03	33.81	36.20	38.91	9.03	2.43	44.91
Voice (Ours, No-δ, Aug.)	48.65	59.46	36.31	32.80	46.32	16.88	7.12	47.64
VGGSound SSL (Aug.)	48.85	61.91	32.67	39.60	47.86	19.63	6.03	53.46
VGGSound SSL (Jitter)	52.95	63.82	38.12	14.20	**50.03**	24.02	3.43	**69.77**
VGGSound-Mix 5s	43.95	59.69	33.31	40.80	46.10	14.71	5.95	52.57
VGGSound-Mix 10s	42.95	57.40	32.03	40.20	46.57	15.77	6.43	51.07
Audio Doppelgängers (Ours)								
Best Synthetic	**58.90**	**66.71**	**39.45**	**44.40**	**48.43**	**24.12**	**9.15**	**58.60**
Voice ($\delta = 0.01$)	47.55	59.56	38.62	11.40	47.53	17.15	6.67	55.56
Voice ($\delta = 0.05$)	47.90	64.02	37.93	13.80	46.45	17.77	7.72	51.52
Voice ($\delta = 0.10$)	48.40	63.92	38.74	11.40	45.13	18.40	7.67	49.32
Voice ($\delta = 0.25$)	**58.90**	**66.71**	**39.45**	32.20	**48.24**	**24.12**	**9.15**	52.95
Voice ($\delta = 0.50$)	41.85	54.03	28.54	40.60	45.78	17.14	4.69	43.85
VoiceFM ($\delta = 0.01$)	42.40	59.89	36.58	9.20	44.31	15.34	5.15	57.13
VoiceFM ($\delta = 0.05$)	42.90	62.96	36.54	14.20	44.93	15.64	5.79	50.61
VoiceFM ($\delta = 0.10$)	44.80	62.03	35.73	14.80	43.99	15.67	5.60	50.56
VoiceFM ($\delta = 0.25$)	57.20	65.11	38.48	35.20	48.43	22.15	6.96	54.00
VoiceFM ($\delta = 0.50$)	43.50	60.98	39.04	12.20	44.17	15.25	6.06	51.07
Parametric ($\delta = 0.01$)	39.50	58.95	36.87	12.20	42.16	13.92	4.53	**58.60**
Parametric ($\delta = 0.05$)	40.15	57.22	35.11	14.60	42.65	12.87	4.78	55.37
Parametric ($\delta = 0.10$)	42.50	59.65	34.12	14.20	43.01	13.41	4.97	53.43
Parametric ($\delta = 0.25$)	50.55	62.83	37.91	37.60	46.77	18.68	5.70	54.72
Parametric ($\delta = 0.50$)	41.15	56.86	35.41	10.40	41.73	12.76	4.48	54.27
Voice ($\delta = 0.01$, Aug.)	52.55	62.92	34.82	23.60	46.96	18.18	8.17	51.01
Voice ($\delta = 0.05$, Aug.)	53.00	65.17	34.49	19.40	45.39	19.79	8.32	49.84
Voice ($\delta = 0.10$, Aug.)	54.20	65.89	33.78	23.40	45.71	20.38	8.50	50.42
Voice ($\delta = 0.25$, Aug.)	58.75	65.01	34.81	**44.40**	46.17	21.76	8.54	50.70
Voice ($\delta = 0.50$, Aug.)	32.25	48.40	25.41	36.20	41.38	11.82	3.26	44.74

Table 12: Complete results for all model variants.

5.6.4 Additional Details on Training

Augmentation Batching

Due to practical considerations in batching and memory management, augmentations are applied differently for real and synthetic data. In real data, augmentations are applied per-example within distributed data-loading workers. Synthetic data is batch-generated within the main process to avoid concurrency issues between JAX's multithreading and PyTorch's data loading. Individually augmenting examples in this synthetic data environment is prohibitively slow. As a solution, we mini-batched augmentations with a default size ≤ 100. This allows us to memory-efficiently leverage GPU processing and introduces variation within each training batch. While per-example augmentations might further enhance performance of synthetic data with augmentations, we believe our current approach is a conservative yet effective option and expect minimal impact.

6 FUTURE WORK

The research in this thesis shows how versatile small virtual synthesizers can be, achieving results comparable to state-of-the-art large black-box models in some tasks. However, there remain multiple exciting opportunities to extend this work.

We developed SynthAX [3] to be the fastest synthesizer available in a framework that would allow for programmatic manipulation of its parameters. However, designing and developing a user interface for easy manipulation would expand its accessibility in creative domains. For instance, an option would be to make it compatible with common audio plugin formats. Moreover, expanding functionality with diverse modules, architectures, and integrated effects processing capabilities would enable the generation of more diverse sounds. All these improvements would directly impact *CTAG* [19]. Simplifying the process of transforming ideas into sounds via text descriptions, while allowing manipulation and interpolation within one single interface, would ultimately empower sound designers and musicians to explore new sonic possibilities.

To further enhance the *audio doppelgängers* [20], future work could scale to larger architectures like AST [127], which might significantly improve our results. Additionally, investigating hybrid approaches that combine the advan-

tages of synthetic and real data might offer a more comprehensive and effective learning strategy. A deeper exploration of the specific features within the data that contribute to its learnability could unveil valuable insights, and studying the learning process from different modules or parameters in isolation, such as focusing on pitch, could reveal nuanced relationships and interactions within the audio data, leading to potentially improved performance on various tasks.

The work in this thesis mainly leverages the controllability of synthesizers, but interpretability remains one of the most important features. Future research could explore this aspect further, particularly by studying the relationship between sound and affective responses. While this connection has been investigated for decades, it still isn't fully understood. Employing interpretable synthesis methods could allow examining causal relationships, and ultimately, enable the synthesis of sounds conditioned on specific affective responses.

7 CONCLUSION

This thesis explores the potential of virtual synthesizers as lightweight, interpretable, and controllable alternatives to complex neural network-based generative audio models. Through the development of SynthAX [3], a fast modular synthesizer built for accelerators, we have demonstrated the capability to generate vast amounts of audio data efficiently.

We leveraged this synthesizer to develop *CTAG* [19], a text-to-parameter method that allows for the creation of semantically aligned sounds from text prompts using only 78 physically-motivated variables. This approach not only achieves high-quality results comparable to state-of-the-art models with billions of parameters but also provides a more abstract and conceptual representation of sounds.

Finally, we show how this same synthesizer can be used in a novel approach for contrastive learning using what we call *audio doppelgängers* [20]—synthetic positive pairs generated by randomly perturbing the parameters of our synthesizer. This method produces robust audio representations competitive with those trained on real data, while offering richer contrastive information through causally manipulated variations in timbre, pitch, and temporal envelopes.

This work highlights the surprising power and versatility of virtual synthesizers as powerful tools in the field of audio machine learning, even with a

relatively small number of parameters. Our work bridges the gap between traditional synthesizer technology and modern computational paradigms, offering a complementary tool to neural network-based approaches in audio synthesis and representation learning.

BIBLIOGRAPHY

[1] Joseph Turian, Jordie Shier, George Tzanetakis, Kirk McNally, and Max Henry. One billion audio sounds from gpu-enabled modular synthesis. In *2021 24th International Conference on Digital Audio Effects (DAFx)*, pages 222–229. IEEE, 2021.

[2] Yusong Wu, Ke Chen, Tianyu Zhang, Yuchen Hui, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[3] Manuel Cherep and Nikhil Singh. Synthax: A fast modular synthesizer in jax. In *Audio Engineering Society Convention 155*. Audio Engineering Society, 2023.

[4] Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. Vggsound: A large-scale audio-visual dataset. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 721–725. IEEE, 2020.

[5] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1041–1044, 2014.

[6] Karol J Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018, 2015.

[7] Fabian-Robert Stöter, Soumitro Chakrabarty, Emanuël Habets, and Bernd Edler. Libricount, a dataset for speaker count estimation, 2018.

[8] Houwei Cao, David G Cooper, Michael K Keutmann, Ruben C Gur, Ani Nenkova, and Ragini Verma. Crema-d: Crowd-sourced emotional multimodal actors dataset. *IEEE transactions on affective computing*, 5(4):377–390, 2014.

[9] Natalie Holz, Pauline Larrouy-Maestri, and David Poeppel. The variably intense vocalizations of affect and emotion (vivae) corpus prompts new perspective on nonspeech perception. *Emotion*, 22(1):213, 2022.

[10] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. Neural audio synthesis of musical notes with wavenet autoencoders. In *International Conference on Machine Learning*, pages 1068–1077. PMLR, 2017.

[11] Eduardo Fonseca, Xavier Favory, Jordi Pons, Frederic Font, and Xavier Serra. Fsd50k: an open dataset of human-labeled sound events. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:829–852, 2021.

[12] Bongjun Kim, Madhav Ghei, Bryan Pardo, and Zhiyao Duan. Vocal imitation set: a dataset of vocally imitated sound events using the audioset ontology. In *DCASE*, pages 148–152, 2018.

[13] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fr\'echet audio distance: A metric for evaluating music enhancement algorithms. *arXiv preprint arXiv:1812.08466*, 2018.

[14] Yonglong Tian, Lijie Fan, Kaifeng Chen, Dina Katabi, Dilip Krishnan, and Phillip Isola. Learning vision from models rivals learning vision from data. *arXiv preprint arXiv:2312.17742*, 2023.

[15] Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. Audiogen: Textually guided audio generation. *arXiv preprint arXiv:2209.15352*, 2022.

[16] Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. *arXiv preprint arXiv:2301.12503*, 2023.

[17] Richard D Ashley. A knowledge-based approach to assistance in timbral design. In *12th International Computer Music Conference, ICMC 1986*, 1986.

[18] Eric Lipmann. I love quincy. Film, 1984. France/UK/USA.

[19] Manuel Cherep, Nikhil Singh, and Jessica Shand. Creative text-to-audio generation via synthesizer programming. *arXiv preprint arXiv:2406.00294*, 2024.

[20] Manuel Cherep and Nikhil Singh. Contrastive learning from synthetic audio doppelgangers. *arXiv preprint arXiv:2406.05923*, 2024.

[21] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: composable transformations of python+ numpy programs. 2018.

[22] Max V Mathews, Joan E Miller, F Richard Moore, John R Pierce, and Jean-Claude Risset. *The technology of computer music*, volume 5. MIT press Cambridge, MA, 1969.

[23] Trevor Pinch and Frank Trocco. *Analog days: The invention and impact of the Moog synthesizer*. Harvard University Press, 2004.

[24] Paul Théberge. *Any sound you can imagine: Making music/consuming technology*. Wesleyan University Press, 1997.

[25] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

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

[27] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis. In *International Conference on Machine Learning*, pages 2410–2419. PMLR, 2018.

[28] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. *arXiv preprint arXiv:1802.04208*, 2018.

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

[30] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. Ddsp: Differentiable digital signal processing. *arXiv preprint arXiv:2001.04643*, 2020.

[31] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin,

Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[32] Ho-Hsiang Wu, Prem Seetharaman, Kundan Kumar, and Juan Pablo Bello. Wav2clip: Learning robust audio representations from clip. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4563–4567. IEEE, 2022.

[33] Katherine Crowson, Stella Biderman, Daniel Kornis, Dashiell Stander, Eric Hallahan, Louis Castricato, and Edward Raff. Vqgan-clip: Open domain image generation and editing with natural language guidance. In *European Conference on Computer Vision*, pages 88–105. Springer, 2022.

[34] Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, and Huaming Wang. Clap: Learning audio concepts from natural language supervision. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[35] Zalán Borsos, Raphaël Marinier, Damien Vincent, Eugene Kharitonov, Olivier Pietquin, Matt Sharifi, Dominik Roblek, Olivier Teboul, David Grangier, Marco Tagliasacchi, et al. Audiolm: a language modeling approach to audio generation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2023.

[36] Xubo Liu, Zhongkai Zhu, Haohe Liu, Yi Yuan, Meng Cui, Qiushi Huang, Jinhua Liang, Yin Cao, Qiuqiang Kong, Mark D Plumbley, et al. Wavjourney: Compositional audio creation with large language models. *arXiv preprint arXiv:2307.14335*, 2023.

[37] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. *arXiv preprint arXiv:2301.12661*, 2023.

[38] Jiawei Huang, Yi Ren, Rongjie Huang, Dongchao Yang, Zhenhui Ye, Chen Zhang, Jinglin Liu, Xiang Yin, Zejun Ma, and Zhou Zhao. Make-an-audio 2: Temporal-enhanced text-to-audio generation. *arXiv preprint arXiv:2305.18474*, 2023.

[39] Yael Vinker, Ehsan Pajouheshgar, Jessica Y Bo, Roman Christian Bachmann, Amit Haim Bermano, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. Clipasso: Semantically-aware object sketching. *ACM Transactions on Graphics (TOG)*, 41(4):1–11, 2022.

[40] Yiren Song. Cliptexture: Text-driven texture synthesis. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 5468–5476, 2022.

[41] Yiren Song, Xuning Shao, Kang Chen, Weidong Zhang, Zhongliang Jing, and Minzhe Li. Clipvg: Text-guided image manipulation using differentiable vector graphics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 2312–2320, 2023.

[42] Yingtao Tian and David Ha. Modern evolution strategies for creativity: Fitting concrete images and abstract concepts. In *International conference on computational intelligence in music, sound, art and design (part of evostar)*, pages 275–291. Springer, 2022.

[43] Nikhil Singh. The sound sketchpad: Expressively combining large and diverse audio collections. In *26th International Conference on Intelligent User Interfaces*, pages 297–301, 2021.

[44] Davide Rocchesso, Guillaume Lemaitre, Patrick Susini, Sten Ternström, and Patrick Boussard. Sketching sound with voice and gesture. *interactions*, 22(1):38–41, 2015.

[45] Halley Young, Maxwell Du, and Osbert Bastani. Neurosymbolic deep generative models for sequence data with relational constraints. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 37254–37266. Curran Associates, Inc., 2022.

[46] Philippe Esling, Axel Chemla-Romeu-Santos, and Adrien Bitton. Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces. In *International Society for Music Information Retrieval Conference*, 2018.

[47] Jordie Shier. *The synthesizer programming problem: improving the usability of sound synthesizers*. PhD thesis, 2021.

[48] 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 Emerging Topics in Computational Intelligence*, 2(2):150–159, 2018.

[49] Masato Hagiwara, Maddie Cusimano, and Jen-Yu Liu. Modeling animal vocalizations through synthesizers. *arXiv preprint arXiv:2210.10857*, 2022.

[50] Philippe Esling, Naotake Masuda, and Axel Chemla-Romeu-Santos. Flowsynth: Simplifying complex audio generation through explorable latent spaces with normalizing flows. In *International Joint Conference on Artificial Intelligence*, 2020.

[51] Xubo Liu, Egor Lakomkin, Konstantinos Vougioukas, Pingchuan Ma, Honglie Chen, Ruiming Xie, Morrie Doulaty, Niko Moritz, Jachym Kolar, Stavros Petridis, et al. Synthvsr: Scaling up visual speech recognition with synthetic supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18806–18815, 2023.

[52] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[53] Varun Kumar, Ashutosh Choudhary, and Eunah Cho. Data augmentation using pre-trained transformer models. *arXiv preprint arXiv:2003.02245*, 2020.

[54] Yu Meng, Jiaxin Huang, Yu Zhang, and Jiawei Han. Generating training data with language models: Towards zero-shot language understanding. *Advances in Neural Information Processing Systems*, 35:462–477, 2022.

[55] Allan Tucker, Zhenchen Wang, Ylenia Rotalinti, and Puja Myles. Generating high-fidelity synthetic patient data for assessing machine learning healthcare software. *NPJ digital medicine*, 3(1):1–13, 2020.

[56] August DuMont Schütte, Jürgen Hetzel, Sergios Gatidis, Tobias Hepp, Benedikt Dietz, Stefan Bauer, and Patrick Schwab. Overcoming barriers to data sharing with medical image generation: a comprehensive evaluation. *NPJ digital medicine*, 4(1):141, 2021.

[57] Shuhan Tan, Yujun Shen, and Bolei Zhou. Improving the fairness of deep generative models without retraining. *arXiv preprint arXiv:2012.04842*, 2020.

[58] Vikram V Ramaswamy, Sunnie SY Kim, and Olga Russakovsky. Fair attribute classification through latent space de-biasing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9301–9310, 2021.

[59] Shayne Longpre, Robert Mahari, Ariel Lee, Campbell Lund, Hamidah Oderinwale, William Brannon, Nayan Saxena, Naana Obeng-Marnu, Tobin South, Cole Hunter, et al. Consent in crisis: The rapid decline of the ai data commons. *arXiv preprint arXiv:2407.14933*, 2024.

[60] Andrew Rosenberg, Yu Zhang, Bhuvana Ramabhadran, Ye Jia, Pedro Moreno, Yonghui Wu, and Zelin Wu. Speech recognition with augmented synthesized speech. In *2019 IEEE automatic speech recognition and understanding workshop (ASRU)*, pages 996–1002. IEEE, 2019.

[61] Nick Rossenbach, Albert Zeyer, Ralf Schlüter, and Hermann Ney. Generating synthetic audio data for attention-based speech recognition systems. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7069–7073. IEEE, 2020.

[62] Aleksandr Laptev, Roman Korostik, Aleksey Svischev, Andrei Andrusenko, Ivan Medennikov, and Sergey Rybin. You do not need more data: Improving end-to-end speech recognition by text-to-speech data augmentation. In *2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 439–444. IEEE, 2020.

[63] Amin Fazel, Wei Yang, Yulan Liu, Roberto Barra-Chicote, Yixiong Meng, Roland Maas, and Jasha Droppo. Synthasr: Unlocking synthetic data for speech recognition. *arXiv preprint arXiv:2106.07803*, 2021.

[64] Ting-Yao Hu, Mohammadreza Armandpour, Ashish Shrivastava, Jen-Hao Rick Chang, Hema Koppula, and Oncel Tuzel. Synt++: Utilizing imperfect synthetic data to improve speech recognition. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7682–7686. IEEE, 2022.

[65] Daria Soboleva, Ondrej Skopek, Márius Šajgalík, Victor Cărbune, Felix Weissenberger, Julia Proskurnia, Bogdan Prisacari, Daniel Valcarce, Justin Lu, Rohit Prabhavalkar, et al. Replacing human audio with synthetic audio for on-device unspoken punctuation prediction. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7653–7657. IEEE, 2021.

[66] Martijn Bartelds, Nay San, Bradley McDonnell, Dan Jurafsky, and Martijn Wieling. Making more of little data: Improving low-resource automatic speech recognition using data augmentation. *arXiv preprint arXiv:2305.10951*, 2023.

[67] Zhehuai Chen, Andrew Rosenberg, Yu Zhang, Gary Wang, Bhuvana Ramabhadran, and Pedro J Moreno. Improving speech recognition using gan-based speech synthesis and contrastive unspoken text selection. In *Interspeech*, pages 556–560, 2020.

[68] Xianrui Zheng, Yulan Liu, Deniz Gunceler, and Daniel Willett. Using synthetic audio to improve the recognition of out-of-vocabulary words in end-to-end asr systems. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5674–5678. IEEE, 2021.

[69] Nicolas Jonason and Bob LT Sturm. Timbreclip: Connecting timbre to text and images. *arXiv preprint arXiv:2211.11225*, 2022.

[70] Zilin Wang, Peng Liu, Jun Chen, Sipan Li, Jinfeng Bai, Gang He, Zhiyong Wu, and Helen Meng. A synthetic corpus generation method for neural vocoder training. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[71] Yuhua Chen, Wen Li, Xiaoran Chen, and Luc Van Gool. Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1841–1850, 2019.

[72] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016.

[73] Gul Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 109–117, 2017.

[74] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2013.

[75] Shakhnarovich, Viola, and Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 750–757. IEEE, 2003.

[76] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016.

[77] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[78] Zhongzheng Ren and Yong Jae Lee. Cross-domain self-supervised multi-task feature learning using synthetic imagery. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 762–771, 2018.

[79] Victor Besnier, Himalaya Jain, Andrei Bursuc, Matthieu Cord, and Patrick Pérez. This dataset does not exist: training models from generated images. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2020.

[80] Suman Ravuri and Oriol Vinyals. Classification accuracy score for conditional generative models. *Advances in neural information processing systems*, 32, 2019.

[81] Ali Jahanian, Xavier Puig, Yonglong Tian, and Phillip Isola. Generative models as a data source for multiview representation learning. *arXiv preprint arXiv:2106.05258*, 2021.

[82] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. Datasetgan: Efficient labeled data factory with minimal human effort. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10145–10155, 2021.

[83] Nontawat Tritrong, Pitchaporn Rewatbowornwong, and Supasorn Suwajanakorn. Repurposing gans for one-shot semantic part segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4475–4485, 2021.

[84] Daiqing Li, Junlin Yang, Karsten Kreis, Antonio Torralba, and Sanja Fidler. Semantic segmentation with generative models: Semi-supervised learning and strong out-of-domain generalization. In *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition, pages 8300–8311, 2021.

[85] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2107–2116, 2017.

[86] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. Pmlr, 2018.

[87] Yonglong Tian, Lijie Fan, Phillip Isola, Huiwen Chang, and Dilip Krishnan. Stablerep: Synthetic images from text-to-image models make strong visual representation learners. *Advances in Neural Information Processing Systems*, 36, 2024.

[88] Brandon Trabucco, Kyle Doherty, Max Gurinas, and Ruslan Salakhutdinov. Effective data augmentation with diffusion models. *arXiv preprint arXiv:2302.07944*, 2023.

[89] Ceyuan Yang, Yujun Shen, Yinghao Xu, and Bolei Zhou. Data-efficient instance generation from instance discrimination. *Advances in Neural Information Processing Systems*, 34:9378–9390, 2021.

[90] Jongheon Jeong and Jinwoo Shin. Training gans with stronger augmentations via contrastive discriminator. *arXiv preprint arXiv:2103.09742*, 2021.

[91] Hirokatsu Kataoka, Kazushige Okayasu, Asato Matsumoto, Eisuke Yamagata, Ryosuke Yamada, Nakamasa Inoue, Akio Nakamura, and Yutaka Satoh. Pre-training without natural images. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[92] Manel Baradad Jurjo, Jonas Wulff, Tongzhou Wang, Phillip Isola, and Antonio Torralba. Learning to see by looking at noise. *Advances in Neural Information Processing Systems*, 34:2556–2569, 2021.

[93] Manel Baradad, Richard Chen, Jonas Wulff, Tongzhou Wang, Rogerio Feris, Antonio Torralba, and Phillip Isola. Procedural image programs for representation learning. *Advances in Neural Information Processing Systems*, 35:6450–6462, 2022.

[94] Haider Al-Tahan and Yalda Mohsenzadeh. Clar: Contrastive learning of auditory representations. In *International Conference on Artificial Intelligence and Statistics*, pages 2530–2538. PMLR, 2021.

[95] Aaqib Saeed, David Grangier, and Neil Zeghidour. Contrastive learning of general-purpose audio representations. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3875–3879. IEEE, 2021.

[96] Mirco Ravanelli and Yoshua Bengio. Learning speaker representations with mutual information. In *Interspeech*, pages 1153–1157, 2019.

[97] Luyu Wang and Aaron van den Oord. Multi-format contrastive learning of audio representations. *arXiv preprint arXiv:2103.06508*, 2021.

[98] Eduardo Fonseca, Diego Ortego, Kevin McGuinness, Noel E O'Connor, and Xavier Serra. Unsupervised contrastive learning of sound event representations. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 371–375. IEEE, 2021.

[99] Craig Bruce. http://www.csbruce.com/quotes/craig/, 1990. Accessed: 2024-07-27.

[100] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2023.

[101] Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020.

[102] Robert Tjarko Lange. evosax: Jax-based evolution strategies. *arXiv preprint arXiv:2212.04180*, 2022.

[103] Yujin Tang, Yingtao Tian, and David Ha. Evojax: Hardware-accelerated neuroevolution. *arXiv preprint arXiv:2202.05008*, 2022.

[104] Bryan Lim, Maxime Allard, Luca Grillotti, and Antoine Cully. Accelerated quality-diversity for robotics through massive parallelism. *arXiv preprint arXiv:2202.01258*, 2022.

[105] Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino. Byol for audio: Exploring pre-trained general-purpose audio representations. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:137–151, 2022.

[106] Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino. Masked modeling duo: Learning representations by encouraging both networks to model the input. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[107] Ricardo Antonio García. *Automatic generation of sound synthesis techniques*. PhD thesis, Massachusetts Institute of Technology, 2001.

[108] Allan Seago, Simon Holland, and Paul Mulholland. A critical analysis of synthesizer user interfaces for timbre. 2004.

[109] Naotake Masuda and Daisuke Saito. Synthesizer sound matching with differentiable dsp. In *ISMIR*, pages 428–434, 2021.

[110] Harri Renney, Benedict Gaster, and Thomas J Mitchell. Survival of the synthesis—gpu accelerating evolutionary sound matching. *Concurrency and Computation: Practice and Experience*, 34(10):e6824, 2022.

[111] Naotake Masuda and Daisuke Saito. Quality-diversity for synthesizer sound matching. *Journal of Information Processing*, 31:220–228, 2023.

[112] Philippe Esling, Naotake Masuda, and Axel Chemla-Romeu-Santos. Flowsynth: simplifying complex audio generation through explorable latent spaces with normalizing flows. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 5273–5275, 2021.

[113] Sebastian Löbbers, Louise Thorpe, and György Fazekas. Sketchsynth: Cross-modal control of sound synthesis. In *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*, pages 164–179. Springer, 2023.

[114] Hugo Scurto, Bavo Van Kerrebroeck, Baptiste Caramiaux, and Frédéric Bevilacqua. Designing deep reinforcement learning for human parameter

exploration. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 28(1):1–35, 2021.

[115] Jordie Shier, George Tzanetakis, and Kirk McNally. Spiegelib: An automatic synthesizer programming library. In *Audio Engineering Society Convention 148*. Audio Engineering Society, 2020.

[116] Hannah Nemer. 'Pop 'n Pour': This Electronic Music Pioneer Created the Sound of Coke's Beloved Bubbles. http://www.coca-colacompany.com/stories/meet-suzanne-ciani-the-legendary-creator-of-cokes-pop-n-pour, 2017. Accessed: 2023-08-15.

[117] Robert Tjarko Lange. evosax: Jax-based evolution strategies. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pages 659–662, 2023.

[118] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017.

[119] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[120] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[121] Robert Lange, Tom Schaul, Yutian Chen, Chris Lu, Tom Zahavy, Valentin Dalibard, and Sebastian Flennerhag. Discovering attention-based genetic algorithms via meta-black-box optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 929–937, 2023.

[122] Eytan Bakshy, Lili Dworkin, Brian Karrer, Konstantin Kashin, Benjamin Letham, Ashwin Murthy, and Shaun Singh. Ae: A domain-agnostic platform for adaptive experimentation. In *Conference on neural information processing systems*, pages 1–8, 2018.

[123] Karol J. Piczak. Esc: Dataset for environmental sound classification. *Proceedings of the 23rd ACM international conference on Multimedia*, 2015.

[124] Ke Chen, Xingjian Du, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Hts-at: A hierarchical token-semantic audio transformer for sound classification and detection. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 646–650. IEEE, 2022.

[125] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[126] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780, 2017.

[127] Yuan Gong, Yu-An Chung, and James R. Glass. Ast: Audio spectrogram transformer. *ArXiv*, abs/2104.01778, 2021.

[128] Jacob Schreiber, Jeffrey Bilmes, and William Stafford Noble. apricot: Submodular selection for data summarization in python. *The Journal of Machine Learning Research*, 21(1):6474–6479, 2020.

[129] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[130] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, José Ricardo Zapata, and Xavier Serra. Essentia: An audio analysis library for music information retrieval. In *International Society for Music Information Retrieval Conference*, 2013.

[131] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[132] Tim Hodgkinson. *Recommended Records Quarterly*, 2(1), 1987. Accessed: 2024-07-27.

[133] Qingqing Huang, Aren Jansen, Joonseok Lee, Ravi Ganti, Judith Yue Li, and Daniel PW Ellis. Mulan: A joint embedding of music audio and natural language. *arXiv preprint arXiv:2208.12415*, 2022.

[134] Janne Spijkervet and John Ashley Burgoyne. Contrastive learning of musical representations. *arXiv preprint arXiv:2103.09410*, 2021.

[135] Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino. Byol for audio: Self-supervised learning for general-purpose audio representation. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[136] Pranay Manocha, Zeyu Jin, Richard Zhang, and Adam Finkelstein. Cdpam: Contrastive learning for perceptual audio similarity. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 196–200. IEEE, 2021.

[137] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. Cnn architectures for large-scale audio classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 131–135. IEEE, 2017.

[138] Nikunj Saunshi, Orestis Plevrakis, Sanjeev Arora, Mikhail Khodak, and Hrishikesh Khandeparkar. A theoretical analysis of contrastive unsupervised representation learning. In *International Conference on Machine Learning*, pages 5628–5637. PMLR, 2019.

[139] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *Advances in neural information processing systems*, 33:6827–6839, 2020.

[140] Christopher Tosh, Akshay Krishnamurthy, and Daniel Hsu. Contrastive learning, multi-view redundancy, and linear models. In *Algorithmic Learning Theory*, pages 1179–1206. PMLR, 2021.

[141] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019.

[142] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[143] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International conference on machine learning*, pages 9929–9939. PMLR, 2020.

[144] Joseph Turian, Jordie Shier, Humair Raj Khan, Bhiksha Raj, Björn W Schuller, Christian J Steinmetz, Colin Malloy, George Tzanetakis, Gissel Velarde, Kirk McNally, et al. Hear: Holistic evaluation of audio representations. In *NeurIPS 2021 Competitions and Demonstrations Track*, pages 125–145. PMLR, 2022.

[145] Moreno La Quatra, Alkis Koudounas, Lorenzo Vaiani, Elena Baralis, Paolo Garza, Luca Cagliero, and Sabato Marco Siniscalchi. Benchmarking representations for speech, music, and acoustic events. In *2024 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*, 2024.

[146] Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, and Huaming Wang. Clap learning audio concepts from natural language supervision. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[147] Tung-Yu Wu, Tsu-Yuan Hsu, Chen-An Li, Tzu-Han Lin, and Hung-yi Lee. The efficacy of self-supervised speech models for audio representations. In *HEAR: Holistic Evaluation of Audio Representations*, pages 90–110. PMLR, 2022.

[148] Geoffroy Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. *CUIDADO Ist Project Report*, 54(0):1–25, 2004.

[149] Keith D Martin and Youngmoo E Kim. 2pmu9. musical instrument identification: A pattern-recognition approach. In *Presented at the 136th meeting of the Acoustical Society of America*, 1998.

[150] Julien Ricard. Towards computational morphological description of sound. *DEA pre-thesis research work, Universitat Pompeu Fabra, Barcelona*, 2004.

[151] George Tzanetakis and Perry Cook. Multifeature audio segmentation for browsing and annotation. In *Proceedings of the 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. WASPAA'99 (Cat. No. 99TH8452)*, pages 103–106. IEEE, 1999.

[152] Cyril Laurier, Owen Meyers, Joan Serra, Martin Blech, Perfecto Herrera, and Xavier Serra. Indexing music by mood: design and integration of an automatic content-based annotator. *Multimedia Tools and Applications*, 48:161–184, 2010.

[153] JA Ballas and MJ Sliwinski. Causal uncertainty in the identification of environmental sounds. *Georgetown University, Washington, DC*, 1986.

[154] Ishwarya Ananthabhotla, David B Ramsay, and Joseph A Paradiso. Hcu400: An annotated dataset for exploring aural phenomenology through causal uncertainty. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 920–924. IEEE, 2019.

[155] Tal Boger, Ishwarya Ananthabhotla, and Joseph Paradiso. Manipulating causal uncertainty in sound objects. In *Proceedings of the 16th International Audio Mostly Conference*, pages 9–15, 2021.

[156] Hermann von Helmholtz. Concerning the perceptions in general. 1867.

[157] Yuan Gong, Yu-An Chung, and James Glass. Ast: Audio spectrogram transformer. *arXiv preprint arXiv:2104.01778*, 2021.

[158] Azalea Gui, Hannes Gamper, Sebastian Braun, and Dimitra Emmanouilidou. Adapting frechet audio distance for generative music evaluation. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1331–1335. IEEE, 2024.

[159] Modan Tailleur, Junwon Lee, Mathieu Lagrange, Keunwoo Choi, Laurie M Heller, Keisuke Imoto, and Yuki Okamoto. Correlation of fr\'echet audio distance with human perception of environmental audio is embedding dependant. *arXiv preprint arXiv:2403.17508*, 2024.

[160] Benjamin Elizalde, Soham Deshmukh, and Huaming Wang. Natural language supervision for general-purpose audio representations, 2023.