Evaluating Sound Similarity Metrics for Differentiable, Iterative Sound-Matching

Amir Salimi, Abram Hindle, Osmar R. Zaïane

***Abstract*—Manual sound design with a synthesizer is inherently iterative: an artist compares the synthesized output to a mental target, adjusts parameters, and repeats until satisfied. Iterative sound-matching automates this workflow by continually programming a synthesizer under the guidance of a loss function (or similarity measure) towards a target sound. Prior comparisons of loss functions have typically favored one metric over another, but only within narrow settings: limited synthesis methods, few loss types, often without blind listening tests. This leaves open the question of whether a universally optimal loss exists, or the choice of loss remains a creative decision conditioned on the synthesis method and the sound designer's preference. We propose differentiable iterative sound-matching as the natural extension of the available literature, since it combines the manual approach to sound design with modern advances in machine learning. To analyze the variability of loss function performance across synthesizers, we implemented a mix of four novel and established differentiable loss functions, and paired them with differentiable subtractive, additive, and AM synthesizers. For each of the sixteen synthesizer–loss combinations, we ran 300 randomized sound-matching trials. Performance was measured using parameter differences, spectrogram-distance metrics, and manually assigned listening scores. We observed a moderate level of consistency among the three performance measures. Our post hoc analysis shows that the loss function performance is highly dependent on the synthesizer. These findings underscore the value of expanding the scope of sound-matching experiments, and developing new similarity metrics tailored to specific synthesis techniques, rather than pursuing one-size-fits-all solutions.**

***Index Terms*—Audio synthesis, differentiable digital signal processing, music information retrieval, sound-matching**

I. Introduction

A digital audio synthesizer is any software used for the creation and manipulation of audio. The possibilities afforded by these synthesizers are infinite, leading to their widespread adoption by artists and sound designers [1, 2, 3]. A typical synthesizer has a number of parameterizable functions which affect the sound output in various ways, and manual sound-design often involves modifying the parameters until a conceptualized target sound is reached. The automation of this approach has commonly been referred to as "sound-matching", with reduction of tinkering time and creation of "interesting" sounds as the main motivators [4, 5, 6, 7, 8, 9, 10, 11, 12, 13]. The main requirements of sound-matching are a target sound, a similarity metric (or loss function), and a heuristic to find parameters for a synthesizer that replicate *all* or *some* of the characteristics of the target sound as best as possible [6, 10, 13]. However, despite the seeming simplicity of the problem and decades of research, sound-matching is not yet a practical solution for sound-designers. Here we are motivated to understand why this is the case, and what future research should prioritize in order to improve the field.

We provide a review of past works and identify major perennial issues in the field. In particular, we are concerned with the issues of **Loss Selection** and **Synthesis Selection**. The former refers to the search for an optimal loss function, lack of novel algorithms, and the—perhaps needless—focus on outperforming state of the art (SOTA), while the latter refers to the lack of diversity in synthesis methods. As a consequence of these problems, we find that not much is known about the interaction between different methods of synthesis and different loss functions. Is there a universally best performing loss function for audio? Is there a need for further development of bespoke loss functions?

The main hypothesis here is that *the performance of a similarity measure (or loss function) is influenced by other factors in the environment, particularly the method of synthesis*. Testing this hypothesis requires a variety of sound-matching experiments that measure whether a single loss function proves most effective. While there have been many works and experiments comparing the accuracy of loss functions [14, 5, 9, 15, 16, 12, 5, 17], we note that claims regarding the effectiveness of one function versus another have often been made in limited contexts that may not generalize to other settings. Beyond this central hypothesis, we also investigate three additional research questions: (Q1) To what extent do automatic evaluation metrics agree with manual listening tests? (Q2) Can loss functions based on Dynamic Time Warping (DTW) and Scale-Invariant Mean Squared Error (SIMSE) provide advantages over SOTA loss functions, and under what conditions? (Q3) Is iterative differentiable optimization a viable strategy for design of sound-matching experiments?

We adopt a lesser used *iterative* and *differentiable* approach to defining sound-matching experiments. The iterative approach better mimics the manual process of recursive listening and parameter adjustments towards sound design, while a differentiable environment allows access to the loss function gradients that can be used to better understand the nature of the problem. We define four differentiable synthesizers (each showcasing a fundamental method of synthesis used in modern synthesizers) and pair them with four different loss functions (two established methods, one utilizing DTW, and one utilizing SIMSE). We evaluate the final similarity with two different automatic methods, as well as manual listening tests conducted by two of the authors. We apply statistical ranking to the evaluation scores to compare outcomes across synthesizer–loss combinations and measure the similarity of different evaluation measures.

Contributions. The contributions of this paper include (1) an evaluation of multiple differentiable losses across multiple synthesis methods, (2) the introduction and justification of

loss functions utilizing DTW and SIMSE (3) a discussion on the utility and agreement between manual and automatic evaluation metrics (4) a nomenclature of different sound-matching approaches and unsolved issues in the field.

II. Background and Related Work

Sound-matching sits at the intersection of digital signal processing, audio representation, and optimization. Here we first formalize the sound-matching task, then review core synthesis methods, similarity measures, and optimization approaches. We conclude with a survey of related work and a discussion of gaps in the field.

A. Formalization of Sound-Matching

Following prior works [14, 16], we define sound-matching in terms of a parametric synthesizer, a target sound, a representation function, and a similarity measure. The key components are:

- $g(\theta)$: A parametric audio synthesizer g with parameters θ.
- x_θ: The synthesizer output, $x_\theta = g(\theta)$.
- x_0: The target sound to be replicated or imitated.
- $\phi(\cdot)$: A representation (feature extraction) function mapping signals into a comparison space.
- L: A loss function that measures distance between x_θ and x_0, typically via $L(\theta, x_0) = d(\phi(x_\theta), \phi(x_0))$ for some metric d.

This formalization highlights that sound-matching is not a single well-defined optimization problem, but rather depends on design choices for g, ϕ, and L. Moreover, the "correct" solution may not exist in a strict sense: depending on the artistic goal, multiple parameter settings may yield acceptable or even preferable outputs. In the following sections we will discuss the various components of sound-matching, the subjectivity of "correct" solutions, and important issues in the field.

Figure 1 shows a general model for iterative sound-matching. To begin the process, the parameters are arbitrarily initialized (usually random generation), the similarity of the target and output is measured, and the parameter is *optimized* with the goal of increasing the similarity, or alternatively, reducing the loss. This process repeats until termination.

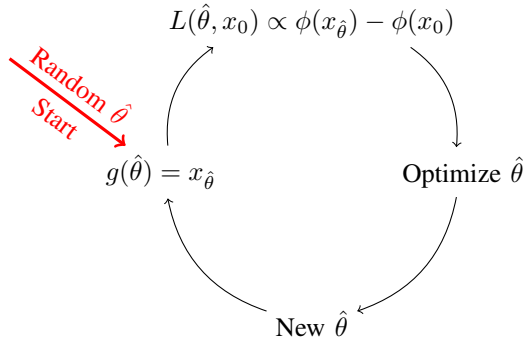


Fig. 1: Iterative approach to sound design.

B. Digital Signal Processing and Synthesis

A digital audio synthesizer generates or processes audio by chaining digital signal processing (DSP) functions. Each function is parameterized, and the set of parameters defining a chain of DSP functions is called a synthesizer program.

The simplest DSP function is a sinusoidal oscillator:

$$x[n] = \sin\left(2\pi f \frac{n}{SR}\right),$$

where f is frequency in hertz, SR is the sampling rate, and n is the discrete time index. At a sampling rate of SR samples per second, a 1 Hz oscillator completes one cycle per second. Such computations form the foundation of digital audio synthesis [1].

Since the advent of DSP in the 1960s [3], a wide variety of parametric functions have been proposed, including oscillators, filters, equalizers, and envelopes [1, 2]. Sound design involves modifying these parameters until a desired output is reached [18, 19].

Early sound-matching research focused heavily on frequency and amplitude modulation (FM/AM) synthesis [6, 10, 14], which is simple to implement yet expressive [20]. Other synthesis methods studied in *isolation* include additive and subtractive synthesis [9, 13, 7] and physical modeling [21, 22]. By *isolation*, we mean settings where the effect of individual parameters on the output sound remains tractable. For example, we exclude studies using commercial Virtual Studio Technology (VST) which can obscure the interactions between modules, losses, and outputs.

Finally, recent years have seen a growing interest in differentiable DSP (DDSP) [9], which integrates gradient-based optimization [23, 24] with DSP building blocks. Implementing complex DSP functions in a differentiable manner remains challenging, and robust differentiable audio similarity measures are still under active investigation [12, 14, 15].

C. Sound Representation and Loss Functions

A digital sound (or an audio signal) is a series of numbers [25, 26]. To compare two digital sounds, the two corresponding series are passed to a function that measures their similarity. Two signals can sound identical to our ears, without having any values in common [27]. This necessitates the use of proxy representations (or feature extractors) when comparing sounds automatically. Similarity between the target sound and the synthesizer output is then measured by some form of subtraction and summation of the proxy representations.

In sound-matching, particularly in a deep learning (DL) context [23], the similarity function can also be called a *loss* function, where the emphasis is on measurement and reduction of the distance between target and output. It is important to note that there is a close relationship between the loss function L and the sound representation function ϕ. L is the result of a distance measure d applied to the features extracted by ϕ.

$$L(\theta, x_0) = d(\phi(x_\theta), \phi(x_0))$$

A proxy representation is the output of the function ϕ, which can be thought of as a feature extraction function that maps

the sounds x_0 and x_θ to their respective representations. The proportionality or distance metric d has typically been the L1 or L2 distance [5, 28], with L1 being calculated as the mean of the absolute difference between every point in the proxy representation:

$$L(\theta, x_0) = \|\phi(x_0) - \phi(x_\theta)\|_1$$

Here we discuss four methods of audio representation and the corresponding loss functions, including technical justifications for our novel methods.

1) Parameter Loss: A common measure of similarity in sound-matching is the distance between synthesizer parameter sets, referred to as "P-Loss" [16]. Typically, for the implementation of P-Loss the parameter sets are treated as vectors in space, and L1 or L2 distance is applied. There are two major limitations to this approach: First, the target and output sound must be made by the same synthesizer; otherwise the parameter sets cannot be compared (see Section II-D). Second, the relationship between synthesizer parameters and the audio output is not linear [11, 16, 8].

2) Fourier Spectrograms: Fourier-based transformations such as short-time Fourier transforms (STFT), Mel-spectrograms, and Mel-frequency cepstral coefficients have been viewed as the de facto and state-of-the-art representation of audio [29, 10, 30], however, there are many issues associated with their use in sound-matching [5, 14, 16, 15]. Fourier transformations allow for the conversion of a signal from the time-domain to the frequency domain. Audio spectrograms can be generated by segmentation of a piece of audio into overlapping windows followed by the application of Fourier transforms to each window. They are costly to compute, but provide a better temporal view of changes in frequency content [31, 26]. There are different types of spectrograms that have a basis in Fourier transformations, but the most notable and commonly used is the STFT. What we call *Fourier-based Spectrograms* are variations on the STFT approach. For example, Mel-Spectrograms *bin* frequencies on a near-logarithmic scale to better match human perception of frequencies [31]. Multi-scale spectrograms (MSS) used in recent works are a simple weighted average of multiple spectrograms with different parameters such as window size, number of frequency bins, and hop size [9, 14]; this may provide some improvements at a higher computational cost [5, 9].

A common limitation of spectrogram-based losses is their sensitivity to global gain: two signals with identical spectral shape but different overall amplitude can yield large errors under L1/L2 norms. Here we propose and test Scale-Invariant Mean Squared Error (SIMSE) as an alternative to L1/L2 norms in spectrogram comparisons. SIMSE normalizes the amplitudes before comparison and emphasizes proportional differences in spectral envelopes [32]. Perceptually, listeners are often tolerant of loudness changes while remaining sensitive to timbral shape, therefore this property could be advantageous for subtractive synthesis, where filter cutoffs reshape spectral balance without predictable changes in total energy. Although SIMSE has been applied in other contexts such as audio reconstruction, to our knowledge its use as a differentiable spectrogram loss in iterative sound-matching is novel.

3) Joint-Time Frequency Transformation: Recent works have focused on the limitations of parameter and spectral loss functions in sound-matching [14, 15], seeking to create more effective general solutions for the comparison of audio. Noting the weaknesses of comparing STFT spectrograms (such as alignment and loudness distances), Vahidi *et al.* proposed differentiable Joint-Time Frequency Scattering (JTFS) [33] as an alternative to spectrogram loss in sound-matching, and showed improved performance in sound-matching with differentiable chirplet synthesizers [14]. JTFS is the result of the application of a 2D wavelet transformation to the time-frequency representation of a signal [33] and has been reported as more sensitive to *mesostructural* features such as melody, syncopation, and textural contrast [14].

4) Dynamic Envelope Warping: Dynamic Time Warping (DTW) is a method for measuring similarity between multi-dimensional time-series [34, 31, 35]. Given any two time-series $X = \{x_1, x_2, ..., x_m\}$ and $Y = \{y_1, y_2, ..., y_n\}$, we have indices $i \in \{1...m\}$ and $j \in \{1...n\}$ defining X and Y. When the series are *warped*, these indices change to expand or contract different portions of the series. To borrow the notation given by Muller [31], warped indices are a sequence $p = (p_1, ..., p_L)$, where $p_\ell = (m_\ell, n_\ell) \in [1 : m] \times [1 : n]$ for $\ell \in [1 : L]$, meaning that the indices for X and Y are reorganized under special conditions. In classical DTW, these conditions are *monotonicity*, *boundary matching*, and *single step-size*. DTW measures the distance between the time-series *after* alignments, typically using Euclidean distance, such that the distance between a time-series and shifted versions of itself would be 0, regardless of shift amount [36]. Additional rules can be imposed to keep alignments locally constrained [37, 38].

DTW provides robustness to local temporal shifts, making it well suited for comparing modulated signals where the perceptual similarity lies in envelope dynamics rather than precise onset alignment. For example, two tremolo signals with slightly different phase are perceptually similar but would appear distant under spectrogram L1. A possible use case to consider is the application of DTW to amplitude envelopes, which may be able to match two sounds with similar rates of loudness modulation regardless of alignments. To our knowledge, DTW applied in this way has not been used as a loss in sound-matching (whether differentiable and iterative or not).

D. In-Domain Versus Out-of-Domain

The choice of domain depends on whether we want to use the same synthesizer for the target and output sounds, known as the *in-domain* scenario, or have target sounds that came from sources other than the synthesizer, known as the *out-of-domain* scenario. To paraphrase the description given by Masuda *et al.* [12], if g, the synthesizer of choice, can accurately replicate the target sound x_0, or put differently, if x_0 itself is an output of g, then the sound-matching task is *in-domain*. If x_0 is not an output of g, then the sound-matching task is *out-of-domain*. In-domain tasks in general are

simpler, and often there is a guarantee that there is a correct answer to the sound-matching problem, particularly if the goal is accurate replication of the sound. If the target sound is out-of-domain, replication is not guaranteed, and the goal becomes the *imitation* of some aspects of sound.

Regardless of the domain, the generation goal can be *replication* or *imitation* of the target sound. In replication, the goal is to make an identical copy of the target sound. Imitation is an artistic pursuit and harder to define, since the goal is to make new sounds that only retain a subset of the target's sonic features. While closely related to the in-domain versus out-of-domain problem, the choice of replication versus imitation is more dependent on how the loss and representation functions are defined.

E. Supervised Versus Direct Optimization

The choice of *heuristics* is yet another important attribute of sound-matching. The goal of sound-matching is to find the optimal parameters θ^* that minimize the loss between the synthesizer output and the target sound.

$$\theta^* = \arg\min_{\theta} L(\theta, x_0)$$

The heuristics (i.e., how θ^* is approximated) used in past works can be broadly split into the two categories of *direct optimization* and *supervised* (or inference) methods. Direct optimization refers to the iterative generation of a sound output, measurement of similarity between target and output, and application of updates to the parameters to maximize similarity (or minimize loss) [6, 10, 30, 14]; while supervised methods use large datasets of synthesizer sounds and corresponding parameters to learn the generation objective, commonly with the use of DNNs [9, 7, 30, 8]. These models often make their parameter predictions in a single step (or 1-shot).

Genetic algorithms (GA) [39] have been the earliest and most common heuristic for direct sound-matching [6, 10, 30]. These algorithms start with arbitrary parameter sets that can be treated as an evolving population where the genomes are the parameter values. The most fit members of the group are the parameters which perform the best in the loss function, and create a new generation of parameters via mutation (random change in subset of parameters) and crossovers (combination of parameter sets); this process repeats until the goal or a maximum number of generations is reached. Rather than using random mutations, differentiable approaches allow goal-oriented updates to the synthesizer parameters (the goal being the minimization of loss), but with some drawbacks: other than requiring more computation power, differentiable functions require careful implementation of operations that are continuous and numerically stable; this makes the implementation of signal processing functions quite difficult, contributing to the scarcity of works in this domain.

F. Historical Framing of Sound-Matching

Table I summarizes relevant literature. Following the discussion of important sound-matching components, a historical analysis of past work can be given.

Perhaps the earliest foundational study is Justice [42], who analytically decomposed and recreated sounds using a simple FM synthesizer (see Section II-B). Subsequent work retained this FM structure but explored new heuristics for parameter search. For example, Horner *et al.* [6] applied genetic algorithms (GAs) to resynthesize sounds with one modulator and one carrier, measuring similarity via the McAulay–Quatieri method [43].

Later studies introduced more complex synthesis models, such as wavetables [44] and physical modeling [21]. Mitchell and Creasy [10] highlighted the difficulty of disentangling synthesizer limitations from optimization inefficiency. They proposed a *contrived methodology* in which the best heuristic for in-domain FM resynthesis should also generalize to out-of-domain targets (e.g., muted trumpet tones [45]). However, limited testing produced contradictory results, and they concluded that changes to the synthesizer, loss, or sound domain effectively redefine the search space [10].

Recent years have seen more works in sound-matching using supervised machine learning techniques. In 2018, Yee-King *et al.* rendered 60,000 audio-parameter pairs from the *Dexed* VST synthesizer [1], and showed that NNs trained on this dataset can outperform GA and hill-climber (HC) [46] methods in rendering speed, with slight improvements in MFCC error (used as an objective performance test). The speed improvement appears trivial, considering the iterative nature of GAs when compared to offline training of supervised models. For GAs and HC optimizer, MFCCs were used as a measure of performance as well as a loss function. For training the networks, P-Loss was used, since differentiable MFCCs were not possible in their pipeline. Importantly, informal listening tests found the results unsatisfactory, likely due to the synthesizer's 155-parameter complexity.

Masuda *et al.* also applied supervised learning [12]. Their work highlights the issue of non-linearity in parameter-to-synthesizer outputs and out-of-domain search. This work uses a differentiable subtractive synthesizer (two oscillators and an LP filter) alongside a NN model pre-trained with P-Loss on an in-domain dataset of randomly selected parameters. After training, the model was fine-tuned using 20,000 out-of-domain sounds from the NSynth dataset [47] and multi-scale spectrogram loss [9]. This approach proved more effective—i.e., achieved lower multi-level spectral difference in out-of-domain tests—than baseline models, which were either not exposed to out-of-domain sounds or trained exclusively with P-Loss. Subjective hearing tests were conducted, showing a preference for the fine-tuned model [12]. Masuda *et al.* later extended this work with semi-supervised learning, highlighting significant gaps in in-domain and out-of-domain performance [13].

Rather than focusing on a particular implementation, Shier *et al.* presented Spieglib, a library for implementation of sound-matching pipelines [11]. This library provides different choices for DNNs, GAs, synthesizers, and feature extractors. Shier *et al.* presented an experiment with a similar setup to Yee-King *et al.* [30], however, they found a genetic algorithm as the best-performing.

[1] https://asb2m10.github.io/dexed/

TABLE I: Summary of select works in sound-matching. The **Gen.** column shows whether the generations are done in 1-shot (generally, output by a neural network) or a search is conducted with the goal of iteratively getting closer to the target sound.

Work	Synthesis	Loss	Heuristics	Goal	Domain	Year	Gen.
Horner *et al.* [6]	Non-differentiable FM	STFT (McAulay-Quatieri)	GA	Replicate	In	1991	Iter
Mitchell and Creasy [10]	Non-differentiable FM	Spectral Relative	Evolutionary	Replicate	In (Contrived)	2007	Iter
Yee-King *et al.* [30]	Non-differentiable VST	P-Loss (for Supervised), Spectral (MFCC)	Supervised & Evolutionary	Replicate	In	2018	Both
Esling *et al.* [8]	Non-differentiable VST	Spec MSS/SC P-Loss	Supervised & Modeling	Replicate	In* & Out	2019	1-shot
Shier *et al.* [11]	Custom, non-diff	FFT & Spectral	Supervised & Direct	Replicate	In	2020	Both
Salimi *et al.* [7]	DSP	STFT and Envelope	Supervised	Imitate	Out	2020	1-shot
Masuda *et al.* [12]	DDSP	P-Loss initially, fine-tune with Spec.	Supervised	Both	In & Out	2021	1-shot
Han *et al.* [16]	NN-encoder $\rightarrow$ physical model	PNP (approx. of STFT)	Supervised	Replicate	In	2023	1-shot
Vahidi *et al.* [14]	Differentiable FM	JTFS/MSS	Gradient desc.	Replicate	In	2023	Iter
Barkan *et al.* [40]	Diff. Synthesizer Proxy	P-Loss + L1 Spec (weighted)	Supervised (init.), Semi-Supervised	Replicate	In	2023	Both
Uzrad *et al.* [15]	DDSP	Synth. Chain and P-Loss	Supervised	Replicate	In & Out	2024	1-shot
Cherep *et al.* [41]	DDSP	CLAP	Evolutionary	Imitate	Out	2024	Iter

Differentiable loss functions that use spectrogram differences can be computationally expensive. To mitigate this, Han *et al.* [16] introduced "perceptual-neural-physical loss" (PNP). PNP is an approximation of loss functions; specifically, a loss function that uses the L2 norm of the difference between the features of two sounds, or $||\phi(t) - \phi(x)||_2^2$, where ϕ could be a spectrogram or JTFS function. PNP loss functions are fast and differentiable, but require training and parameter estimation. A Riemannian metric M needs to be calculated for the minimization of locally linear approximation of the "true" spectral loss function [16].

$$\|\varphi(x_0) - \varphi(x_\theta)\|_2^2 = \langle \tilde{\theta} - \theta \,|\, M(\theta) \,|\, \tilde{\theta} - \theta \rangle + O(\|\tilde{\theta} - \theta\|_3^2).$$

This metric is calculated alongside the neural parameter estimator. Once trained, PNP enabled the fast differentiable optimization of computationally expensive loss functions such as JTFS with FM and physical models [16, 22].

A neural approximation for another part of the supervised sound-matching chain—this time the synthesizer—was proposed by Barkan *et al.* [40]. As they noted, without a differentiable synthesizer, "model-based" (or what we call supervised) approaches cannot directly compare x_θ to x_0, often opting for P-Loss, which may not correctly map the parameters of sound to the output audio [8, 16, 13]. This approach used a model for mapping sounds to parameters, another for approximating the synthesizer, and a loss function which combines P-Loss with STFT differences. This "Inversynth II" (IS2) approach yielded significant improvements to previous works which did not use the STFT approximations for loss [8, 48]. Barkan *et al.* then attempted to improve the IS2 model with Inference-Time Fine-tuning (ITF). For ITF, the synthesizer approximation is frozen and the encoder is iteratively fine-tuned for a particular sample. However, ITF often degraded performance, possibly due to proxy–synthesizer mismatch or overfitting.

Uzrad *et al.* [15] took another unique approach to sound-matching: using a differentiable *synthesis chain* of DSP generators and effects along with a loss function that combines P-Loss with a *signal-chain loss*. The synthesizer is a customizable chain of effects, which feeds one output as input to the next step of the chain; signal-chain loss compares the differences in parameter and output at every output step in the chain [15]. Possible chain functionalities are FM/AM, low-frequency oscillators (LFOs), filters, and envelopes. Like the results shown by Masuda *et al.* [12], better out-of-domain results were achieved when pre-trained on in-domain data and fine-tuned using out-of-domain NSynth data [47].

Audio embeddings can also be used as a similarity metric. For example, Cherep *et al.* used latent representations from the CLAP model [49] along with a differentiable synthesizer [50] to create creative interpretations of sound effects. In their approach, a desired sound-effect is described in text and embedded using CLAP, followed by iterative updates from a gradient-free optimizer [51] to the synthesizer's parameters to minimize embedding differences. Based on manual hearing tests, this approach did not produce the "correct" sounds more often than prior methods [52]. However, it did yield better scores for "artistic-interpretation" (what we referred to as "imitation"). It should be noted that audio embedding models are often trained using simpler (often Fourier-based) loss functions, therefore improvements to the fundamental loss functions may also lead to better embeddings.

G. What Is Lacking In the Field

Having looked at the current literature, we identify four major areas of weakness in sound-matching. We target the first two issues in this work, while the latter two are left for future work.

1) **Loss Selection**: It remains unclear whether there exists a universally best-performing loss function, or whether performance depends on factors such as the sound domain, synthesizer architecture, and desired output characteristics. Existing studies typically evaluate losses only with simple synthesis setups [14], leaving open questions about their generality.
2) **Synthesis Selection**: There is a lack of diversity in the synthesis methods used in sound-matching. The effects of using different isolated DSP functions in iterative sound-matching and how these interact with loss functions remains largely untested.
3) **Loss Landscape Navigation**: Loss function landscapes are not easy to navigate, as there can be many local minima, or large flat areas [5, 14]. In differentiable settings, this can cause gradient descent updates to not reach the global minima.
4) **Out-of-Domain Generation**: Sound-matching with out-of-domain sounds is an under-explored area, yet a necessary one for practical applications for sound designers.

H. What Approach Should We Take?

We argue that isolated benchmarking is a necessary precursor to building robust, generalizable sound-matching systems. To address the problems of **Loss Selection** and **Synthesis Selection**, we adopt a methodology with three key characteristics that have not previously appeared together.

1) Use simple differentiable synthesizers built from common DSP functions (oscillators, filters, envelopes) to address the **Synthesis Selection** problem. Restriction to small-scale, low-parameter models avoids the confounding effects of neural proxies or embedding models, enabling direct analysis of how synthesis method interacts with loss functions.
2) Evaluate experiments under multiple loss functions to address the **Loss Selection** problem. Prior work has typically tested losses in isolation, making direct comparisons of loss functions difficult [14, 16, 15]. Particularly rare are comparisons which are done in the context of different methods of synthesis.
3) Optimize synthesizer parameters directly with gradient descent, without neural approximations. This complements prior proxy-based methods by providing the missing "low-level experiments" that clarify fundamental interactions between DSP functions and similarity measures.

Together, these design choices supply the controlled experimental evidence missing from the literature, clarifying how losses and synthesis methods interact in iterative sound-matching, and offering insights likely to generalize to more complex domains.

III. Experimental Setup and Results

At a glance, our methodology is to conduct controlled, low-level differentiable experiments without the use of proxy networks. We pair four differentiable losses with four differentiable DSP-based synthesizers: BP-Noise (band-pass noise with LP/HP filters), Add-SineSaw (additive sine and saw oscillators), Noise-AM (noise modulated by an LFO), and SineSaw-AM (sine-modulated saw oscillator). We optimize parameters directly via gradients in order to isolate loss–synthesizer interactions. This controlled setup systematically evaluates the performance of iterative sound-matching pipelines, where "performance" is defined as the similarity of the synthesized output to a target. Similarity is assessed with P-Loss, MSS, and—most importantly—manual listening tests. This design allows us to directly test our central hypothesis: that the effectiveness of a loss function depends on the synthesis method, and that no universal "best" similarity measure exists.

Here we discuss the implementation of the four loss functions (Section III-A) across four differentiable synthesizers (Section III-B). For each loss–synthesizer pair, we run a large number of trials and evaluate their outcomes automatically and manually (Section III-C). "Experiment" in this context refers to one complete iterative sound-matching run, from random parameter initialization through iterative gradient-based optimization until termination. Scores from these experiments are then aggregated and compared to determine best-performing losses and synth–loss pairings.

A. Loss Function Implementation Details

1) STFT Losses: Due to their ubiquity and lower cost of gradient calculation, we use STFT as the basis of two loss functions. L1 or L2 distances are the most common methods of comparing spectrograms [5, 28]. However, these measures can be overly sensitive to global gain or minor misalignments, potentially overstating perceptual differences. To address this, we also test Scale-Invariant Mean Squared Error (SIMSE) as an alternative to L1 for comparing STFT spectrograms.

We define the `L1_Spec` and `SIMSE_Spec` functions as the application of L1 and SIMSE to the STFT spectrograms. The STFT spectrogram function uses 512 FFT bins, window size of 600 samples, and hop length (how many samples the window shifts) of 100 samples. The L1 and SIMSE implementations are differentiable.

2) JTFS Loss: The `JTFS` loss function is the application of L1 difference to the JTFS representations of two sounds [14]. The code used for the JTFS transformation is the differentiable implementation of a 1-dimensional JTFS function provided by Andreux *et al.* [53].

3) Soft-DTW Loss: We use the soft-DTW function, which is differentiable and—depending on its parameters—not shift-invariant [54, 55, 56]. The loss function `DTW_Envelope` is the application of the soft-DTW function to the envelope of the two sounds being compared [1]. This loss uses the similarity of amplitude modulation patterns, which are perceptually important in many sound-design scenarios. The envelope is calculated by creating the STFT spectrogram of a sound (the

Listing 1: **BP-Noise**

```
import("stdfaust.lib");
lp_cut = hslider("lp_cut",900,50,1000,1);
hp_cut = hslider("hp_cut",100,1,120,1);
process = no.noise:fi.lowpass(3,lp_cut):fi.
    highpass(10,hp_cut);
```

Listing 2: **Add-SineSaw**

```
import("stdfaust.lib");
saw_freq = hslider("saw_freq",800,20,1000,1)
    ;
sine_freq = hslider("sine_freq"
    ,300,20,1000,1);
sineOsc(f) = +(f/ma.SR) ~ ma.frac:*(2*ma.PI)
     : sin;
sawOsc(f) = +(f/ma.SR) ~ ma.frac;
process = sineOsc(sine_freq)+sawOsc(saw_freq
    );
```

same process used in Section II-C2) and summing the values at each timestep.

B. The Synthesizers

The synthesizer programs are meant to be simple examples that test the building blocks of digital sound synthesis. Subtractive, additive, and FM/AM synthesis are three of the most common techniques in sound design [25]. In subtractive synthesis, frequencies are removed from a sound via digital filters. In additive synthesis, complex sounds are created via the linear combination of simpler sounds [1, 57]. As discussed previously, FM/AM synthesis refers to the general technique of modulating the frequency or amplitude of a waveform (the carrier) by another waveform (the modulator).

For each program, we provide the Faust code [58], which can be run in the online IDE [2]. Faust is a functional language for audio synthesis that can succinctly define signal processing chains. Following Braun's methodology [59], programs are first defined in Faust, then converted (or *transpiled*) to differentiable JAX functions using the DawDreamer library[3].

[2]https://faustide.grame.fr/
[3]https://github.com/DBraun/DawDreamer

Listing 3: **Noise-AM**

```
import("stdfaust.lib");
amp = hslider("amp",0.5,0,5,0.01);
modulator = hslider("modulator"
    ,0.5,0,4,0.01);
sineOsc(f) = +(f/ma.SR) ~ ma.frac:*(2*ma.PI)
     : sin;
process = no.noise*sineOsc(modulator)*amp;
```

Listing 4: **SineSaw-AM**

```
import("stdfaust.lib");
carrier = hslider("carrier",100,20,1000,1);
amp = hslider("amp",6,1,20,1);
sineOsc(f) = +(f/ma.SR) ~ ma.frac:*(2*ma.PI)
     : sin;
sawOsc(f) = +(f/ma.SR) ~ ma.frac;
process = sineOsc(amp)*sawOsc(carrier);
```

*1) **BP-Noise**:* **BP-Noise** is a bare-bones example of subtractive synthesis using digital filters [57]. Despite the ubiquity of subtractive synthesis in practical sound design, it has rarely been tested in differentiable sound-matching. In this program, a noise signal is fed through a band-pass (BP) filter. This removes the frequencies outside the low-pass (LP) and high-pass (HP) cutoffs. The noise generator produces all frequencies with random variation over time. The LP filter removes frequencies over its threshold frequency, and the HP removes frequencies lower than its threshold [57]. The search parameters are the cutoff thresholds for the LP and HP filters. Listing 1 shows the Faust code for **BP-Noise**.

*2) **Add-SineSaw**:* **Add-SineSaw** is an additive program that combines a saw and a sine wave function. Like subtractive synthesis, additive synthesis is a common approach to sound design that has not been extensively tested as a benchmark in sound-matching. The search parameters here are the frequency of the sine and saw oscillators. Listing 2 is the Faust code for **Add-SineSaw**.

*3) **Noise-AM**:* **Noise-AM** involves amplitude modulation of a noise generator by an LFO with *modulator* as its frequency and a global *amp* value that does not change over time and applies to the entire signal. The search parameters for this program are the LFO frequency and the amp value. The amp value acts as a global volume control, and since the sounds are normalized before analysis, it is likely inconsequential to the performance. This program is meant to be a stepping stone to **SineSaw-AM**, which utilizes proper AM synthesis. Listing 3 is the Faust code for **Noise-AM**.

*4) **SineSaw-AM**:* Relative to additive and subtractive synthesis, AM/FM synthesis are less common methods of sound design. Due to their frequent use in previous works, we also tested an AM synthesizer. The synthesizer program is the multiplication of a low frequency sine oscillator with frequency parameter *amp*, and a saw oscillator with frequency parameter *carrier*. Listing 4 is the Faust code for **SineSaw-AM**.

C. Evaluation Methods

The two automatic evaluation methods used here are P-Loss and MSS, as described in Section II-C. For P-Loss, the parameters are normalized between 0-1 based on the valid ranges defined in the Faust program, and the L1 distance between the normalized parameters is calculated. MSS is computed using a hop size of 100 samples, and FFT window lengths of $(512, 1024, 2048, 4096)$.

Automatic sound evaluation methods are often unreliable, as their alignment with the subjective judgments of human listeners is uncertain. Therefore, listening tests are conducted by randomly sampling 40 experiment results for each program and loss function pair. Two of the authors then assign blinded similarity scores to the outputs and targets using a 5-point Likert scale (1 = no similarity, 5 = near identical) [60]. Using Spearman's rank correlation [61, 62], we found a very strong rate of correlation between the human listeners (see Section III-G), a good indicator that the scores assigned by manual listeners can be treated as the ground truth. We then

combine the ranks assigned by both authors, giving us 80 ranks per program and loss function pair.

D. Training Loop and Gradient Calculations

Given a differentiable loss and synthesizer, the iterative sound-matching procedure is as follows:

1) Initialize θ^* and $\hat{\theta}$, i.e., random generation of target and initial parameters uniformly over a predefined range
2) Generating the output of the synthesizer with $\hat{\theta}$ (the length of the output is set to 1 second with sample-rate of 44100 Hz)
3) Calculating the loss between the target and output
4) Applying gradient updates to the synthesizer parameters
5) Repeating the second step with the updated parameters $\hat{\theta}$, until maximum number of iterations has been reached

For updating the synthesizer parameters (or weights), we use RMSProp, which operates similar to stochastic gradient descent (SGD) [23], with the caveat that the gradients of each weight are scaled by the root-mean-square of past gradients of that weight. Based on some initial test runs, we used an arbitrary fixed learning rate of 0.045 for all experiments (learning rates used in Vahidi *et al.* [14] is unknown). The maximum number of iterations is set to 200, where, based on our observations, the parameters have either irrecoverably diverged outside the acceptable ranges, or are stuck at a local minimum. Additionally, the gradients are large, and calculated backwards sequentially through the signal processing chain. This backward calculation resembles the *exploding gradients* problem in recurrent neural networks [63]. Gradient clipping [23] is preemptively used to ensure that the ℓ_2 norm of all gradients does not exceed the threshold of 1.

E. Ranking Loss Functions

Figure 2 is a visualized summary of how the loss functions are ranked. We have four loss functions and four synthesizer programs. The maximum number of iterations set to 200, and 300 experiments for each loss/synthesizer combination. Each synthesizer program is paired with four loss functions and 300 experiments are conducted and evaluated automatically with P-Loss and MSS, and manually with Likert scores. With this approach, for each program and loss function pair, there are three distributions that can be used to rank the loss functions. Two distributions with 300 automatically assigned similarities (P-Loss and MSS) and one distribution with 80 Likert scores (combining the 40 ranks assigned by each author).

For consistency and statistical robustness, the distributions are upsampled to 1000 values using bootstrapping [64]. Bootstrapping gives an estimation of the distribution for the mean performance of each experiment. k (set to 1000) samples of n values (set to 100% of the values) are taken with replacement from the empirical distribution (list of 300 performance values for MSS and P-Loss or 80 for manual rankings). The mean performance is calculated for each of the k samples. The k estimates of the mean performance give us a bootstrapped distribution for each loss function.

A post-hoc test is conducted in two stages in order to determine whether there is a difference in performance between loss functions for each program, and if so, which are the best performers. The first stage determines whether there is a difference in group means, with the null hypothesis being that all groups have similar mean ranks. The second stage ranks the loss functions from best to worst using the non-parametric Scott-Knott test (NPSK) [65, 66]. The Kruskal-Wallis test is used for the first stage [67]; this test pools and ranks all evaluation measures for a program, then tests whether the differences in mean rank for all loss function groups is zero. The Kruskal-Wallis test calculates an H statistic that is compared to a chi-square distribution with k-1 degrees of freedom (k is the number of groups, or 4, and degrees of freedom is 3). Using the significance level of 0.05, if the H statistic is greater than the critical value of the chi-square distribution then the null hypothesis is rejected, meaning that at least one group has a mean rank significantly above others [67].

TABLE II: Do loss functions have different median performance? Kruskal-Wallis results by program and bootstrapped evaluation results.

Program	Eval. Method	H-Stat.	P-value	Reject
BP-Noise	MSS	81.19	1.70×10^{-17}	Yes
BP-Noise	P-Loss	163.60	3.07×10^{-35}	Yes
BP-Noise	Manual	9.45	2.39×10^{-2}	Yes
Add-SineSaw	MSS	308.97	1.14×10^{-66}	Yes
Add-SineSaw	P-Loss	348.42	3.28×10^{-75}	Yes
Add-SineSaw	Manual	9.45	2.39×10^{-2}	Yes
Noise-AM	MSS	1.35	0.7171	No
Noise-AM	P-Loss	366.76	3.50×10^{-79}	Yes
Noise-AM	Manual	32.71	3.70×10^{-7}	Yes
SineSaw-AM	MSS	564.65	4.65×10^{-122}	Yes
SineSaw-AM	P-Loss	229.19	2.07×10^{-49}	Yes
SineSaw-AM	Manual	207.58	9.69×10^{-45}	Yes

F. Best Performers For Each Program

Table II shows the Kruskal-Wallis results for every program and evaluation method, we see significant differences between loss function performance measures (i.e., the bootstrapped distributions of the evaluation for each loss function) in 11 of the 12 cases, with the exception of MSS for the **Noise-AM** program.

Based on the bootstrapped distribution of the scores, the NPSK algorithm ranks the loss functions from 1 (best) to a maximum of 4 (worst). In cases where the distributions are similar, multiple loss functions can be clustered into the same rank.

In Figure 3, we use color-coded violin plots to visualize the best performers for each program. The corresponding colors for each rank are **1** **2** **3** **4**.

*1) **BP-Noise**:* For this synthesizer program, the spectrogram-based models performed the best. This makes intuitive sense, as the visual effects of a band-pass filter on white noise are readily apparent in a spectrogram. As shown in Figure 3a, manual hearing tests and MSS selected `SIMSE_Spec` as the best performer and `L1_Spec` as the second-best performer, while P-Loss gave the reverse order.

*2) **Add-SineSaw**:* As shown in Figure 3b, JTFS was the best performing loss function in all evaluation methods. Moreover, all evaluation methods produced identical results.

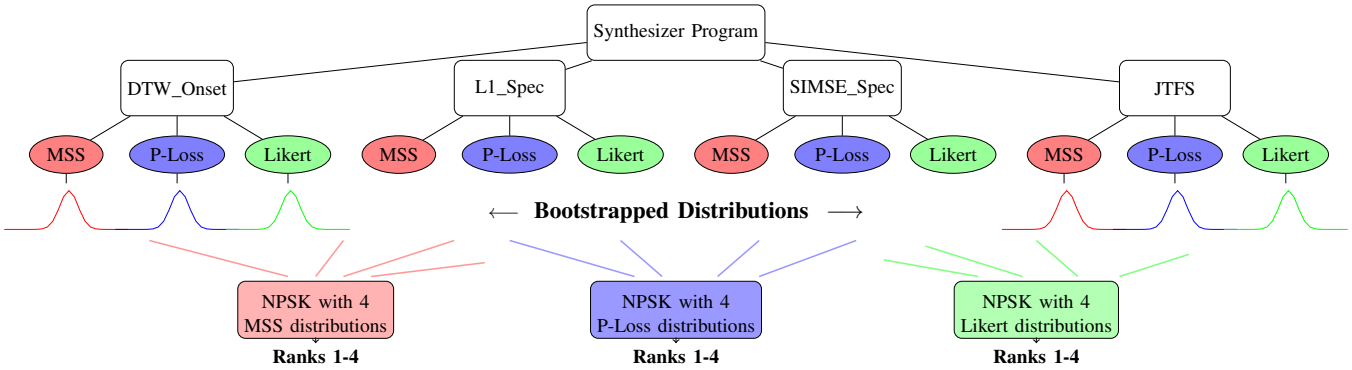


Fig. 2: For each synthesizer program, we assign ranks to the four loss functions (`DTW_Envelope`, `L1_Spec`, `SIMSE_Spec`, `JTFS`) in three different ways, using MSS, P-Loss, or manually assigned Likert scores.

*3) **Noise-AM**:* P-Loss and manual hearing tests yield identical rankings, and MSS results vary for ranks 2 to 4. It is not surprising that MSS results differ from the hearing tests, as we saw in Table II, MSS showed no significant differences between groups. As shown in Figure 3c, all models selected `DTW_Envelope` as the best performer. This may be due to `DTW_Envelope`'s focus on periodic changes in loudness.

*4) **SineSaw-AM**:* `DTW_Envelope` is again the best performer here. As shown in Figure 3d, all methods of analysis gave identical rankings to all programs.

G. Consistency In Rankings

1) Manual Ranks: Correlation between the manually assigned scores is measured using "Spearman's rank correlation", which provides both a correlation coefficient (ρ) ranging from –1 (perfect negative correlation) to 1 (perfect positive correlation), and a p-value testing the null hypothesis of no correlation [61, 62]. Across all programs, the correlation was very strong ($\rho = 0.86$, $p < 10^{-180}$). Per-program correlations were also very strong: $\rho = 0.71$ for **BP-Noise** ($p < 10^{-25}$), $\rho = 0.64$ for **Add-SineSaw** ($p < 10^{-19}$), $\rho = 0.84$ for **Noise-AM** ($p < 10^{-43}$), and $\rho = 0.85$ for **SineSaw-AM** ($p < 10^{-44}$).

Table III shows the SNPK rankings of bootstrapped evaluation results for each program. Both MSS and P-Loss gave a different rank from the hearing results in 3 out of 16 cases, which shows consistency between automatic and manual hearing tests, at least for the simple programs used in this work. We also observe that top ranks were consistent across performance evaluation methodologies, with the exception of P-Loss in **BP-Noise**, which narrowly picks a different spectrogram-based loss function.

The experiment results provide answers regarding the main hypothesis of this work as well as the secondary questions. In the following sections, we will discuss our key findings, caveats, and make recommendations for future research.

IV. Discussion

A. Key Findings

The most important takeaway regarding our main hypothesis is that the loss function that yields the best sound-matching outcomes varies depending on the synthesizer program. This is likely due to the interaction between how the parameters of the synthesizer influence the sound, and the core sonic features used by the loss function.

Q1: To what extent do automatic evaluation metrics agree with manual listening tests?
We see somewhat consistent results between the rankings assigned by manual hearing tests (which we take as the ground truth), and automatic measures of P-Loss and MSS. With the exception of P-Loss in **BP-Noise** narrowly selecting a different spectrogram-based loss, all measures of performance selected the same top performer. However, given the simplicity of our programs and the occasional deviations, we cannot conclude that these automatic measures are substitutes for human hearing tests. As many previous works have done (see Table I), the use of MSS and/or P-Loss as general loss functions does seem appropriate in general cases, however, the use of listening tests or custom made loss functions in specific contexts remains necessary.

Q2: Are DTW and SIMSE effective measures of loss? If so, in what contexts?
`DTW_Envelope` consistently outperformed other measures in amplitude-modulated synthesis, where envelope alignment is critical. `SIMSE_Spec` performed best in subtractive synthesis, where scale-invariance captures noise-filtering effects more robustly than L1-based measures. These results suggest that the advantages of loss functions are context-dependent, underscoring the need for further exploration of creative differentiable similarity measures.

Q3: Can iterative differentiable optimization be applied effectively with synthesizers using classical DSP functions?
The approach of designing DSP functions in Faust and transpiling to differentiable Faust code was convenient, yielded meaningful outputs, and enabled comparative evaluation of losses. We were able to run our 1200 experiments within 72 hours on a laptop without a GPU[4].

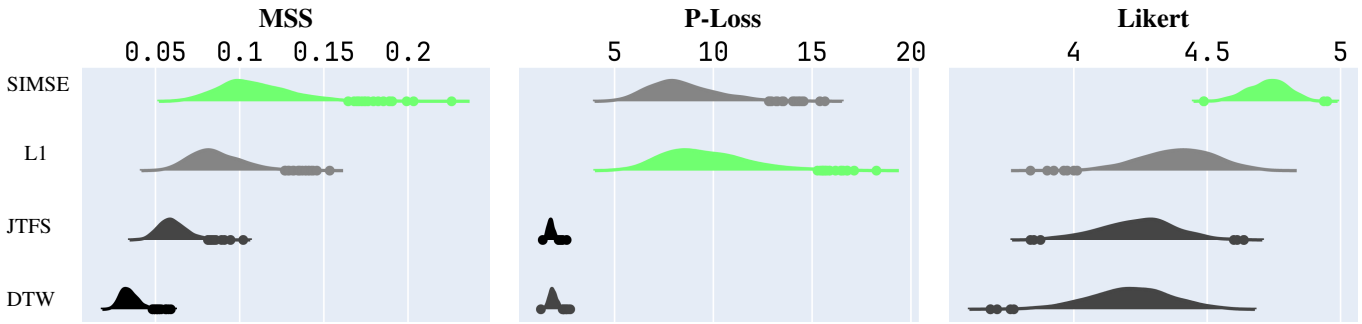
B. Practical Recommendations

Our findings not only provide answers to the research questions proposed in the introduction, but also serve as

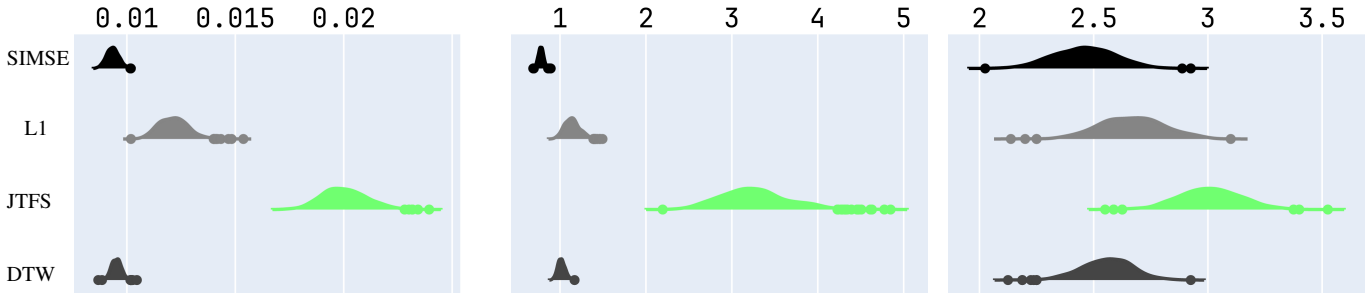
[4]Lenovo ThinkPad T480, i5-8350U CPU at 1.70GHz, 32 GB RAM.

TABLE III: Ranks for each synthesis method (rows) under the three evaluation metrics (columns), across four targets (**BP-Noise**, **Add-SineSaw**, **Noise-AM**, **SineSaw-AM**). Values marked with an asterisk are different from the Likert score hearing ranks.

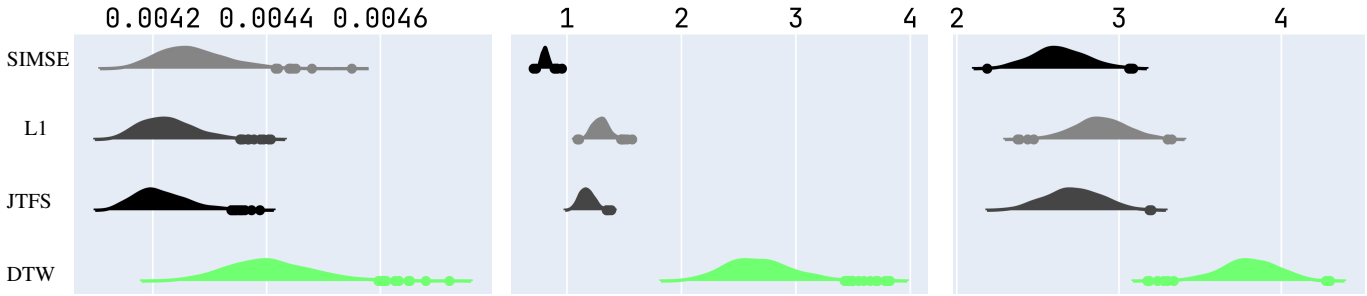
Function	BP-Noise			Add-SineSaw			Noise-AM			SineSaw-AM		
	MSS	P-LOSS	Hearing	MSS	P-LOSS	Hearing	MSS	P-LOSS	Hearing	MSS	P-LOSS	Hearing
SIMSE	1	***2**	1	4	4	4	***3**	4	4	2	2	2
L1	2	***1**	2	2	2	2	2	2	2	3	3	3
JTFS	3	***4**	3	1	1	1	***4**	3	3	4	4	4
DTW	***4**	3	3	3	3	3	1	1	1	1	1	1



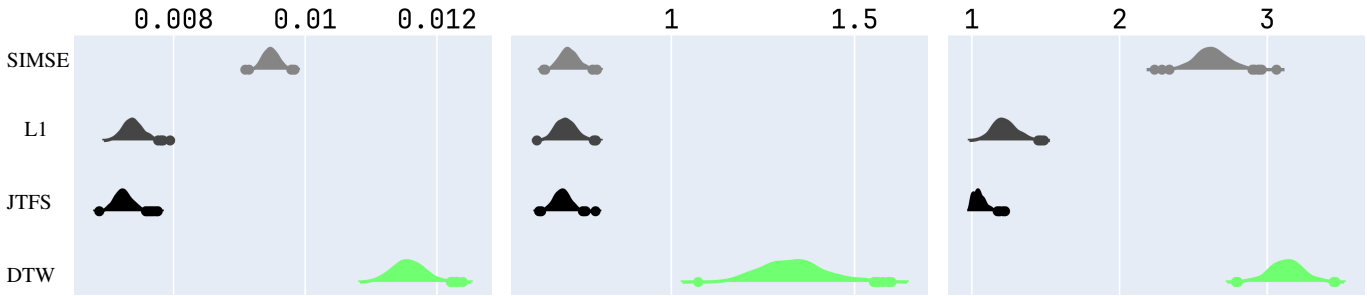
(a) **BP-Noise** bootstrapped distributions and ranks given by NPSK.



(b) **Add-SineSaw** bootstrapped distributions and ranks given by NPSK.



(c) **Noise-AM** bootstrapped distributions and ranks given by NPSK.



(d) **SineSaw-AM** bootstrapped distributions and ranks given by NPSK.

Fig. 3: Distributions and ranks of the loss functions based on three different performance measures. From left to right, the performance measures are: MSS, P-Loss, and Likert. Higher values indicate better performance. Rank colors are **1** **2** **3** **4**.

guidelines for practitioners in selecting appropriate similarity measures and synthesis methods for future experiments.

- In lieu of listening tests, MSS or P-Loss are generally useful for large-scale benchmarking. However, we encourage confirmation of such results with manual listening, particularly if specificity is important.
- For amplitude-modulated synthesis, DTW-based losses such as `DTW_Envelope` are recommended over spectrogram differences and JTFS.
- For subtractive/noise-filtered synthesis, spectrogram based losses are most effective. Based on our results, SIMSE appears to be the better method of spectrogram comparisons relative to L1, but the amplitude agnosticism of SIMSE likely plays a role in its poor performance in additive synthesis.
- Iterative differentiable optimization via the Faust-to-JAX pipeline is a viable strategy for defining various differentiable DSP functions, requiring only modest hardware resources and providing a more natural approach to synthesizer definition.
- In our amplitude modulation experiments, `JTFS` did not outperform other measures, contradicting prior claims of its superiority for mesostructures [14]. Thus, our results call into question its effectiveness as a general mesostructure measure, and highlight a need for further research into its utility.

C. Illustrating Loss Landscapes

To complement our quantitative analysis, we provide examples of loss landscapes that illustrate why specific losses succeed or fail in certain synthesis contexts. These are illustrative aids rather than conclusive evidence, and highlight the importance of **Loss Landscape Navigation** discussed earlier in Section II-G.

We show the landscapes of the loss functions with simplified versions of **BP-Noise** (HP-Noise) and **Noise-AM** (S-Noise-AM), each reduced to a single parameter for one-dimensional visualization. HP-Noise uses only a high-pass filter with cutoff between 100–20,000 Hz, while S-Noise-AM varies only the modulation rate between 0.1–20 Hz. To simplify comparisons, loss values and parameter ranges are respectively normalized to 0–1 and -1–1. Figure 4 shows the HP-Noise results. `L1_Spec`, `SIMSE_Spec`, and `JTFS` exhibit clear minima at the correct parameter (red dashed line), whereas `DTW_Envelope` remains relatively flat, making gradient-based optimization more difficult.

Figure 5 shows the S-Noise-AM results. Here, `DTW_Envelope` produces the smoothest and most informative landscape, while spectrogram losses are minimized near the target but lack consistent correlation with parameter distance. `JTFS` remains largely flat. This aligns with `DTW_Envelope`'s superior performance in amplitude-modulated synthesis observed in our experiments.

D. Applicability to More Complex Synthesizers

The experiments in this study were conducted with differentiable synthesizers chosen to isolate fundamental synthesis principles, each with two parameters. While this level of simplicity facilitates controlled comparisons, it does not fully capture the richness of real-world synthesizers, which can have hundreds of parameters with various routes. As a result, the extent of the generalizability of our findings does require further research.

Nonetheless, we believe that the insights here can extend naturally to more complex domains. Depending on the feature of interest, the observed dependence of loss performance on synthesis method is likely to remain stable even with added complexities such as layers of modulation, filtering, or nonlinearity.

`DTW_Envelope` measures very specific features of audio, and its use in the recommended settings—where low frequency amplitude modulation matching is required—would likely yield the desired results regardless of synthesizer complexity. Likewise, the use of `SIMSE_Spec` or `L1_Spec` for matching filter cut-offs is likely to be successful regardless of the underlying sound.

V. Summary, Weaknesses, and Conclusion

Summary: Sound-matching is an umbrella term for the algorithmic programming of audio synthesizers, often with the goal of assisting sound designers. Here we provided a history of sound-matching, major issues in the field, and discussed the importance of "differentiable iterative sound-matching" as a natural extension of current literature.

The main hypothesis tested here is whether the performance of differentiable loss functions is influenced by the synthesis techniques used for sound-matching. We conducted systematic iterative sound-matching experiments by combining four different loss functions with four different sound synthesis programs. We ranked the performance of the iterative sound-matching pipelines for every loss function and program, and observed that the success of the pipeline (that is, how closely the output sound matches the target sound) is program dependent. In other words, different synthesizer programs work best with different loss functions. Notably, we see that our novel use of DTW and SIMSE based differentiable loss functions (`DTW_Envelope` and `SIMSE_Spec`) can outperform what are regarded as the SOTA loss functions in 3 of 4 cases, although their success is highly synthesizer dependent.

P-Loss and MSS have frequently been used as automatic performance measures, yet their "preference" has rarely been compared to human rankings. We observed that automatic performance measures and manual listening tests were generally in agreement, despite this, manual verification of sound-matching results remains a necessity in non-general experiments due to occasional divergences between automatic and manual tests.

Weaknesses: While we cannot prove that a universally best similarity measure does not exist, we can advocate for more creativity in the field. Compared to previous work, we presented a more cohesive approach to iterative sound-matching which utilizes a variety of loss functions and synthesis methods. However, there are many other methods of synthesis and sound-similarity that can be combined in practically infinite

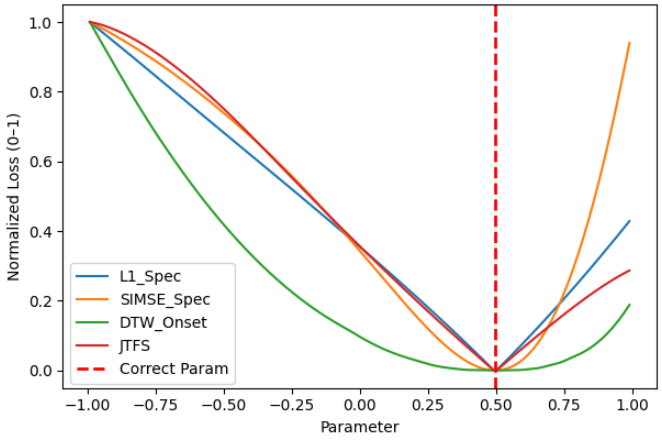


Fig. 4: Loss landscapes for **BP-Noise** with only a high-pass filter parameter. `L1_Spec`, `SIMSE_Spec`, and `JTFS` show clear global minima near the correct parameter, while `DTW_Envelope` remains flat around the target.

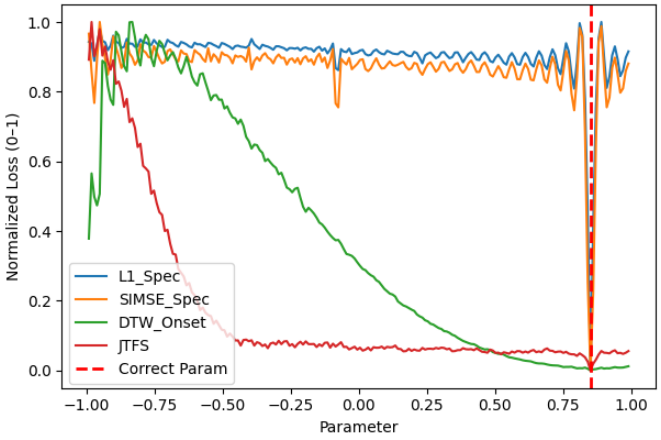


Fig. 5: Loss landscapes for a simplified **Noise-AM** synthesizer with only amplitude modulation parameter. `DTW_Envelope` exhibits the smoothest and most informative landscape, explaining its superior performance in amplitude-modulated synthesis.

ways. Due to this large search space, we set arbitrary parameters for the various signal processing functions. We used bare-bones versions of STFT and JTFS with fixed parameters. We did not test complex synthesizers using parallel and sequential DSP functions. Arbitrary hyperparameters such as learning rate and max number of iterations were selected for the DL pipeline, and only the RMSProp optimizer was tested.

Like the majority of previous works, this work utilizes in-domain sounds; that is, the target sound is made by the synthesizer, and the parameters are already known. This simplifies the issues of measuring sound-similarity, but it is not a realistic scenario for practical sound-matching. This problem is left for future work, discussed in the next section.

Future Work: The problem of periodic loss-landscapes, noted by many previous works, remains unaddressed [5, 14, 15, 17]. Perhaps this problem emerges due to the periodic nature of sound, which requires better loss landscape navigation methods. An optimizer that is more aware of fluctuations in the gradients would perhaps lead to better solutions than simple gradient descent. Viewing the sound-matching problem as "the navigation of an agent from an arbitrary point in a gradient field to a target" closely resembles many classical problems in the field of reinforcement learning (RL) [68]. Naturally, the application of RL and other heuristic search techniques to the problem of iterative sound-matching would be an important contribution.

Contemporary works often involve the application of domain specific and computationally expensive loss functions [16, 15], use of large neural networks [69, 70], or complex ensemble methods [71]. Such models are useful but intractable; furthermore, training them requires the definition of simpler loss functions, which emphasizes the need for further development of differentiable and expressive loss function implementations.

Like nearly all previous work in sound-matching, the main measure of success here was the accurate *replication* of sounds (or synthesizer parameters) rather than *imitation* of sounds outside of the synthesizer's domain. Replication of arbitrary features of sounds is an important component of sound-design, though it is much harder to define or measure. Future work could explore loss functions which only measure certain characteristics of sound (such as `DTW_Envelope`), and whether they can pave the way for better imitation in sound-matching.

References

[1] R. G. Lyons, *Understanding digital signal processing, 3/E*. Pearson Education India, 1997.

[2] M. Russ, *Sound synthesis and sampling*. Routledge, 2012.

[3] D. Stranneby, *Digital signal processing and applications*. Elsevier, 2004.

[4] G. Krekovic, "Insights in habits and attitudes regarding programming sound synthesizers: A quantitative study," in *Proceedings of the 16th Sound and Music Computing Conference*, 2019.

[5] J. Turian and M. Henry, "I'm sorry for your loss: Spectrally-based audio distances are bad at pitch," *arXiv preprint arXiv:2012.04572*, 2020.

[6] A. Horner, J. Beauchamp, and L. Haken, "Machine tongues xvi: Genetic algorithms and their application to FM matching synthesis," *Computer Music Journal*, vol. 17, no. 4, pp. 17–29, 1993.

[7] A. Salimi and A. Hindle, "Make your own audience: Virtual listeners can filter generated drum programs," in *Proceedings of the 2020 AI Music Creativity Conference*, 2020, pp. 1–8.

[8] P. Esling, N. Masuda, A. Bardet, R. Despres, and A. Chemla-Romeu-Santos, "Flow synthesizer: Universal audio synthesizer control with normalizing flows," *Applied Sciences*, vol. 10, no. 1, p. 302, 2019.

[9] J. Engel, L. H. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in

International Conference on Learning Representations, 2020. [Online]. Available: https://openreview.net/forum?id=B1x1ma4tDr

[10] T. J. Mitchell and D. P. Creasey, "Evolutionary sound matching: A test methodology and comparative study," in *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*. IEEE, 2007, pp. 229–234.

[11] J. Shier, G. Tzanetakis, and K. McNally, "Spiegelib: An automatic synthesizer programming library," in *148th Audio Engineering Society Convention*, May 2020.

[12] N. Masuda and D. Saito, "Synthesizer sound matching with differentiable DSP," in *ISMIR*, 2021, pp. 428–434.

[13] ——, "Improving semi-supervised differentiable synthesizer sound matching for practical applications," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 863–875, 2023.

[14] C. Vahidi, H. Han, C. Wang, M. Lagrange, G. Fazekas, and V. Lostanlen, "Mesostructures: Beyond spectrogram loss in differentiable time–frequency analysis," *Journal of the Audio Engineering Society*, vol. 71, pp. 577–585, September 2023.

[15] N. Uzrad, O. Barkan, A. Elharar, S. Shvartzman, M. Laufer, L. Wolf, and N. Koenigstein, "Diffmoog: A differentiable modular synthesizer for sound matching," *arXiv preprint arXiv:2401.12570*, 2024.

[16] H. Han, V. Lostanlen, and M. Lagrange, "Perceptual–neural–physical sound matching," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[17] F. Bruford, F. Blang, and S. Nercessian, "Synthesizer sound matching using audio spectrogram transformers," in *Proceedings of the 27th International Conference on Digital Audio Effects*, 2024.

[18] C. Roads, *The Computer Music Tutorial*. MIT press, 1996.

[19] T. Pinch and F. Trocco, *Analog days: The invention and impact of the Moog synthesizer*. Harvard University Press, 2004.

[20] J. M. Chowning, "The synthesis of complex audio spectra by means of frequency modulation," *Journal of the audio engineering society*, vol. 21, no. 7, pp. 526–534, 1973.

[21] J. Riionheimo and V. Välimäki, "Parameter estimation of a plucked string synthesis model using a genetic algorithm with perceptual fitness calculation," *EURASIP Journal on Advances in Signal Processing*, vol. 2003, pp. 1–15, 2003.

[22] H. Han, V. Lostanlen, and M. Lagrange, "Learning to solve inverse problems for perceptual sound matching," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 2605–2615, 2024.

[23] I. Goodfellow, "Deep learning," 2016.

[24] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[25] J. O. I. Smith, "Viewpoints on the history of digital synthesis," in *Proceedings of the International Computer Music Conference*. International Computer Music Association, 1991.

[26] ——, *Mathematics of the Discrete Fourier Transform (DFT): With Audio Applications*. Julius Smith, 2007.

[27] B. C. Moore, *An introduction to the psychology of hearing*. Brill, 2012.

[28] G. Richard, V. Lostanlen, Y.-H. Yang, and M. Müller, "Model-based deep learning for music information research: Leveraging diverse knowledge sources to enhance explainability, controllability, and resource efficiency [special issue on model-based and data-driven audio signal processing]," *IEEE Signal Processing Magazine*, vol. 41, no. 6, pp. 51–59, 2025.

[29] J. W. Beauchamp and A. Horner, "Error metrics for predicting discrimination of original and spectrally altered musical instrument sounds," *The Journal of the Acoustical Society of America*, vol. 114, no. 4_Supplement, pp. 2325–2325, 2003.

[30] M. J. Yee-King, L. Fedden, and M. d'Inverno, "Automatic programming of VST sound synthesizers using deep networks and other techniques," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 150–159, 2018.

[31] M. Müller, *Dynamic Time Warping*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, ch. 4, pp. 69–84. [Online]. Available: https://doi.org/10.1007/978-3-540-74048-3_4

[32] J. T. Barron and J. Malik, "Shape, illumination, and reflectance from shading," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 8, pp. 1670–1687, 2014.

[33] J. Andén, V. Lostanlen, and S. Mallat, "Joint time-frequency scattering for audio classification," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2015, pp. 1–6.

[34] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., 1993.

[35] T. Giorgino, "Computing and visualizing dynamic time warping alignments in R: the DTW package," *Journal of Statistical Software*, vol. 31, pp. 1–24, 2009.

[36] R. Tavenard, "An introduction to dynamic time warping," https://rtavenar.github.io/blog/dtw.html, 2021.

[37] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 23, no. 1, pp. 67–72, 1975.

[38] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE transactions on acoustics, speech, and signal processing*, vol. 26, no. 1, pp. 43–49, 1978.

[39] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–73, 1992.

[40] O. Barkan, S. Shvartzman, N. Uzrad, M. Laufer, A. Elharar, and N. Koenigstein, "Inversynth ii: Sound matching via self-supervised synthesizer-proxy and inference-time finetuning," in *24th International Society for Music Information Retrieval Conference (ISMIR 2023)*. ISMIR, 2023, 24th International Society for Music Information Retrieval Conference (ISMIR 2023) ; Conference date: 05-11-2023 Through 09-11-2023.

[41] M. Cherep, N. Singh, and J. Shand, "Creative text-to-audio generation via synthesizer programming," in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24. JMLR.org, 2024.

[42] J. Justice, "Analytic signal processing in music computation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 27, no. 6, pp. 670–684, 1979.

[43] R. McAulay and T. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 4, pp. 744–754, 1986.

[44] A. Horner, "Auto-programmable FM and wavetable synthesizers," *Contemporary Music Review*, vol. 22, no. 3, pp. 21–29, 2003.

[45] F. Opolko and J. Wapnick, "Mcgill university master samples (mums)," 1987, Faculty of Music, McGill Univ., Montreal, QC, Canada CD-ROM.

[46] J. Hoffmann, "A heuristic for domain independent planning and its use in an enforced hill-climbing algorithm," in *International Symposium on Methodologies for Intelligent Systems*. Springer, 2000, pp. 216–227.

[47] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with wavenet autoencoders," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1068–1077.

[48] O. Barkan, D. Tsiris, O. Katz, and N. Koenigstein, "Inversynth: Deep estimation of synthesizer parameter configurations from audio signals," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 12, pp. 2385–2396, 2019.

[49] Y. Wu, K. Chen, T. Zhang, Y. Hui, T. Berg-Kirkpatrick, and S. 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)*. IEEE, 2023, pp. 1–5.

[50] M. Cherep and N. Singh, "Synthax: A fast modular synthesizer in jax," in *Audio Engineering Society Convention 155*, May 2023. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=22261

[51] R. T. Lange, "evosax: Jax-based evolution strategies," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 2023, pp. 659–662.

[52] F. Kreuk, G. Synnaeve, A. Polyak, U. Singer, A. Défossez, J. Copet, D. Parikh, Y. Taigman, and Y. Adi, "Audiogen: Textually guided audio generation," in *The Eleventh International Conference on Learning Representations*, 2023.

[53] M. Andreux, T. Angles, G. Exarchakis, R. Leonarduzzi, G. Rochette, L. Thiry, J. Zarka, S. Mallat, J. Andén, E. Belilovsky *et al.*, "Kymatio: Scattering transforms in python," *Journal of Machine Learning Research*, vol. 21, no. 60, pp. 1–6, 2020.

[54] M. Cuturi and M. Blondel, "Soft-DTW: A differentiable loss function for time-series," in *International Conference on Machine Learning*. PMLR, 2017, pp. 894–903.

[55] H. Janati, M. Cuturi, and A. Gramfort, "Spatio-temporal alignments: Optimal transport through space and time," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2020, pp. 1695–1704.

[56] R. Tavenard, "Differentiability of dtw and the case of soft-dtw," https://rtavenar.github.io/blog/softdtw.html, 2021.

[57] J. O. I. Smith, *Introduction to Digital Filters: With Audio Applications*. Julius Smith, 2007, vol. 2.

[58] Y. Orlarey, D. Fober, and S. Letz, "Faust: An efficient functional approach to DSP programming," *New Computational Paradigms for Computer Music*, pp. 65–96, 2009.

[59] D. Braun, "DAC-JAX: A JAX implementation of the descript audio codec," *arXiv preprint arXiv:2405.11554*, 2024.

[60] A. T. Jebb, V. Ng, and L. Tay, "A review of key likert scale development advances: 1995–2019," *Frontiers in Psychology*, vol. 12, p. 637547, 2021. [Online]. Available: https://doi.org/10.3389/fpsyg.2021.637547

[61] C. Spearman, "The proof and measurement of association between two things," *The American Journal of Psychology*, vol. 100, no. 3/4, pp. 441–471, 1987.

[62] A. Rebekić, Z. Lončarić, S. Petrović, and S. Marić, "Pearson's or spearman's correlation coefficient-which one to use?" *Poljoprivreda*, vol. 21, no. 2, pp. 47–54, 2015.

[63] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[64] R. J. Tibshirani and B. Efron, "An introduction to the bootstrap," *Monographs on Statistics and Applied Probability*, vol. 57, no. 1, pp. 1–436, 1993.

[65] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "An empirical comparison of model validation techniques for defect prediction models," *IEEE Transactions on Software Engineering (TSE)*, vol. 43, no. 1, 2017.

[66] C. Tantithamthavorn, S. McIntosh, A. Hassan, and K. Matsumoto, "The impact of automated parameter optimization on defect prediction models," *IEEE Transactions on Software Engineering*, vol. 45, no. 7, pp. 683–711, 2019.

[67] W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," *Journal of the American statistical Association*, vol. 47, no. 260, pp. 583–621, 1952.

[68] R. S. Sutton, "Reinforcement learning: An introduction," *A Bradford Book*, 2018.

[69] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, "CNN architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 131–135.

[70] A. L. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, "Look, listen, and learn more: Design choices for deep audio embeddings," in *ICASSP 2019-2019 IEEE Inter-*

national Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 3852–3856.

[71] J. Turian, J. Shier, H. R. Khan, B. Raj, B. W. Schuller, C. J. Steinmetz, C. Malloy, G. Tzanetakis, G. Velarde, K. McNally *et al.*, "Hear: Holistic evaluation of audio representations," in *NeurIPS 2021 Competitions and Demonstrations Track*. PMLR, 2022, pp. 125–145.