Improving Semi-Supervised Differentiable Synthesizer Sound Matching for Practical Applications

Naotake Masuda and Daisuke Saito, *Member, IEEE*

***Abstract*—While synthesizers have become commonplace in music production, many users find it difficult to control the parameters of a synthesizer to create a sound as they intended. In order to assist the user, the *sound matching* task aims to estimate synthesis parameters that produce a sound that is as close as possible to the query sound. Recently, neural networks have been employed for this task. These neural networks are trained on paired data of synthesis parameters and the corresponding output sound, optimizing a loss of synthesis parameters. However, query by the user usually consists of real-world sounds, different from the synthesizer output sounds used as training data. In a previous work, the authors presented a sound matching method where the synthesizer is implemented using differentiable DSP. The estimator network could then be trained by directly optimizing the spectral similarity between the original sound and the output sound. Furthermore, the network could be trained on real-world sounds whose ground-truth synthesis parameters are unavailable. This method was shown to improve the match quality in both objective and subjective measures. In this work, we experiment with different synthesizer configurations and extend this approach to a more practical synthesizer with effect modules and envelope generators. We propose a novel training strategy where the network is fully trained using both parameter loss and spectral loss. We show that models trained using this strategy is able to utilize the chorus effect effectively while models that switch completely to spectral loss underutilizes the chorus effect.**

***Index Terms*—Audio synthesis, synthesizer, music information retrieval, semi-supervised learning, neural network.**

I. Introduction

SYNTHESIZERS are widely used in modern music production and sound design, owing to their ability to create unique and diverse timbre. Sound designers can create sounds by directly interacting with the parameters of the synthesis algorithm through the interface of the synthesizer. Despite the prevalence of synthesizers, most novice music producers find it difficult to design a sound with the synthesizer. This is because it is often unclear what adjustments to the synthesis parameters are needed to get the desired sound. Thus, many producers rely on *presets*, preconfigured parameter settings for the synthesizer crafted by sound designers. For such producers, the sonic palette of the synthesizer is limited by the availability of presets, and the full potential of the synthesizer is out of reach.

There have been numerous works on utilizing machine learning methods to facilitate the use of conventional synthesizers. A common approach is *sound matching*, estimation of synthesis parameters that best replicate the target sound. Techniques such as genetic algorithm, linear regression, and neural network (NN) have been applied to various synthesizer architectures. Among these methods, NNs have been shown to return the best matches in a reasonable amount of computing time [1]. In such cases, NNs are trained to predict the synthesis parameters from the features of the target audio as a regression/classification problem, minimizing the error of estimated parameters against ground-truth parameters.

However, this parameter loss may not be the ideal loss for the sound matching task. In fact, it has been found that the model with the best performance in terms of parameter loss does not perform the best in terms of spectral features of the actual output sound [2]. Since we are interested in the auditory similarity to the target sound, it may be better to optimize the network using a loss function directly related to the output audio. Unfortunately, conventional synthesizers do not allow for backpropagation of the gradients, which prevents the optimization of such a loss function.

Another problem with previous NN-based methods is that models are trained only on sounds created by the same synthesizer used for creating matches (we will refer to these sounds as *in-domain* sounds). Sounds that we want to imitate using a synthesizer using a sound matching system usually consist of real-world sounds from various sources (*out-of-domain* sounds). Since out-of-domain sounds are not labeled with the ground-truth parameter values that create the best match, only in-domain sounds can be used for training.

Earlier, the authors presented a novel approach to the problem of synthesizer sound matching by implementing a conventional synthesizer using differentiable DSP [3]. Differentiable DSP [4] allows for backpropagation of the gradients, meaning that we can optimize not only the parameter loss but also the estimation network using a loss function based on the spectral features of synthesized audio. Since the network can be optimized without the parameter loss, unlabeled out-of-domain sounds can be used

Manuscript received 7 June 2022; revised 8 November 2022; accepted 26 December 2022. Date of publication 16 January 2023; date of current version 25 January 2023. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Hema A Murthy. *(Corresponding author: Naotake Masuda.)*

The authors are with the Department of Electrical Engineering and Information Systems, Graduate School of Engineering, The University of Tokyo, Tokyo 113-8656, Japan (e-mail: n_masuda@gavo.t.u-tokyo.ac.jp; dsk_saito@gavo.t.u-tokyo.ac.jp).

Digital Object Identifier 10.1109/TASLP.2023.3237161

as training data. The use of out-of-domain sounds was shown to be instrumental in improving the perceptual quality of the estimated match.

In this work, we extend the capabilities of our synthesizer with differentiable versions of ADSR envelopes and effect modules (chorus and reverb). While differentiable implementations of the reverb effect has been explored previously [4], to our knowledge, this work is the first to estimate the parameters of the chorus effect and ADSR envelope in a differentiable manner. The ADSR envelope is present in most conventional synthesizers, since they are crucial for the interpretability and controllability of the parameter dynamics. Furthermore, we find that models trained on spectral loss tend to underutilize the chorus effect, because the spectral loss does not provide proper gradients to adjust the chorus delay parameter. Thus, we propose a novel training scheme where the network is trained on in-domain sounds and out-of-domain sounds alternately, and both spectral loss and parameter loss is used until the end of training. Finally, we investigate in detail the relationship between synthesizer architecture and the difficulty of parameter estimation. We experiment with a model conditioned on the fundamental frequency estimated by an external network to separate the problem of frequency estimation. We also examine the effects of making oscillators unordered in terms of frequency.

In Section II, we cover the related works for this work including previous works in synthesizer sound matching. In Section III, we explain the architecture of the proposed method including the differentiable synthesizer. To analyze the behavior of this differentiable synthesizer, we examine the values and analytic gradients of spectral loss on a synthetic benchmark in Section IV. From these results, we design the sound matching experiments. The experiment setup is explained in Section V and the results are shown in Section VI. We discuss our findings in Section VII and end with a conclusion in Section VIII.

II. Related Works

A. Synthesizer Sound Matching

Synthesizer sound matching refers to the estimation of synthesizer parameters that can closely replicate the target sound. This can be seen as an inverse problem of sound synthesis. One important thing to address is the motivation for such a task. Why must we go through the effort of recreating a sound with a synthesizer? One reason is that a sound matching algorithm can assist the user of a synthesizer during sound design. For example, a user can query the system with a sound that is similar to but not exactly according to their intended sound. Then, the estimated parameters can be tweaked further using the synthesizer interface to better fit the user's needs. In this case, the sound matching results serve as a starting point for sound design. Another reason is playability. By recreating a sound using the synthesizer, its pitch, duration and velocity can be changed flexibly. A sampler may serve similar purposes, but requires time-stretching of audio, which may alter the timbral qualities in undesired ways.

A simple way of achieving sound matching would be to view it as a sound retrieval problem from a database of synthesizer presets [5]. This requires that each preset in the database is rendered into audio and coupled with the corresponding preset. Then, a query-by-example sound retrieval algorithm [6] can be used to find the entry with the audio that is closest to the query sound. A limitation of this approach is that a large database of presets that sufficiently cover the capabilities of a synthesizer is not available for most synthesizers, as the distribution of presets is limited and often not free.

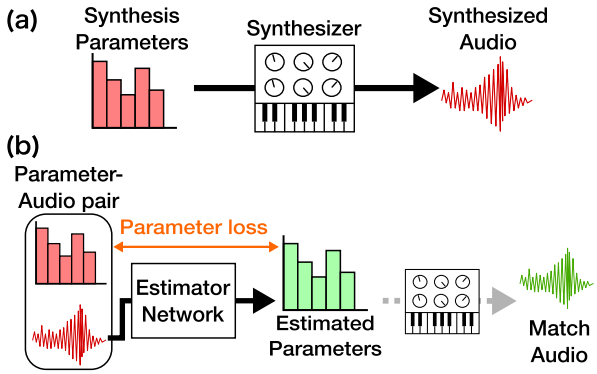


Fig. 1. *NN-based synthesizer sound matching.* (a) A synthesizer renders audio according to synthesis parameters. (b) An estimator network is trained to estimate the parameters from the sound, optimizing the parameter loss.

Earlier work in sound matching used genetic algorithm (GA) to estimate the FM synthesis parameters to match a target spectrum frame [7]. The fitness function is defined as how close the produced spectrum is to the target spectrum. Some work have expanded upon this with different fitness functions [8] and synthesis algorithm [9]. As these work deal with matching spectra of a static tone, they may be better described as *tone matching*.

Subsequent works in GA-based sound matching have focused on the more complex task of matching the entire sound whose spectra vary over time. The fitness function is typically defined as the distance in terms of time-varying audio features such as spectrogram or mel-frequency cepstrum coefficients (MFCC). Many synthesizers such as physical modeling synthesizer [10], FM synthesizer [1], [11], and combination of many synthesis algorithms [12] have been used with GA-based sound matching. The main disadvantage of GA is that it takes anywhere from 10 minutes to several hours to match a single sound, since fitness of each individual can only be evaluated by rendering audio. For example, a single session with population size of 200 running for 200 generations would require 40,000 renders of the synthesizer.

In recent years, supervised machine learning methods such as multiple linear regression [13] and neural networks (NNs) [1], [2], [14] have been used for sound matching. In this approach, sound matching is approached as a regression (for continuous synthesis parameters) or classification (for discrete/categorical parameters) problem, where the synthesis parameters are estimated from features of the target sound (Fig. 1). For training such a model, a synthesizer is often randomly sampled to create a dataset consisting of synthesis parameters and the resulting audio. When a large dataset of handcrafted presets are available, they can be used for training data as well.

A similar field to synthesizer sound matching is *intelligent music production*, which aims to automate the process of mixing and mastering by estimating the parameters of audio effects. For example, siamese network was used to estimate the parameters of a dynamic compressor from the original and processed audio [15]. Another work dealt with the mixing parameters of a multi-track project such as pan, gain and reverb for each track [16].

While NNs allow for fast estimation of synthesis parameters during inference, they optimize the parameter loss and not the actual match quality of the synthesized audio. This is because gradients can not be propagated through the conventional synthesizer. To circumvent the same problem in the case of black-box audio effects, stochastic gradient approximation methods were applied [17]. However, the gradients obtained for the audio effect are only approximate.

Finally, it is also important to consider that the users of a sound matching application want to reproduce out-of-domain sounds. For example, use of vocal imitation as a query for sound matching has been proposed [5]. Perhaps a user will want to imitate acoustic instrument sounds using a synthesizer. Such sounds cannot be matched perfectly, and the synthesis algorithm imposes an upper bound on the match quality. However, synthesizers can imitate some of their qualities, perhaps leading to the discovery of unique timbre. Thus, out-of-domain sounds should be the focus of sound matching.

However, match quality of out-of-domain sounds been neglected in most works. In [2], models were evaluated in terms of spectral distance against out-of-domain sounds. Still, conventional neural network models must be trained on labeled data, consisting of pairs of audio output and the synthesis parameters used to produce it. As such, out-of-domain sounds are unseen during training, resulting in a domain-gap during training and inference.

B. Neural Audio Synthesis

Recently, audio synthesis methods based on neural networks have garnered attention. Since a neural network is a black-box with millions of model parameters, they do not allow for direct interaction with parameters like a conventional synthesizer. As such, neural networks must offer another way to control the synthesis. This is achieved through either model conditioning or learning a latent representation of musical sounds.

For example, SING is a neural audio synthesis model that can be conditioned by the pitch, velocity, and instrument labels [18]. Embeddings for the instrument can be learned to adjust the timbre more flexibly [19]. Alternatively, an autoencoder can be used to learn the latent representation of musical sounds with which a user can control the output. A basic autoencoder with feedforward layers was used to reconstruct spectral frames [20]. A WaveNet autoencoder has been used to model raw audio of musical sounds [21]. Although most neural audio synthesis methods have high latency, some newer models such as RAVE boast real-time audio synthesis [22].

Compared to conventional synthesizers, these neural audio synthesis models can potentially create more realistic sounds and offer a different way to control the synthesis result. However, their controls are not as flexible as a conventional synthesizer and have yet to be employed in common music production workflows.

C. Differentiable DSP

While neural audio synthesis models generate raw audio using only neural networks, differentiable digital signal processing (DDSP) integrates conventional signal processing modules into deep learning, exploiting their strong inductive biases on musical audio. In the original DDSP paper, the parameters of a differentiable audio synthesis model were estimated by a neural network in an end-to-end manner [4]. More specifically, a differentiable version of an additive synthesis model called the harmonics-plus-noise model, a variant of the sinusoids-plus-noise model [23], was used. While this model can technically be viewed as a synthesizer, its controls are much more complicated than conventional synthesizers, as the amplitude of each harmonic and the full frequency response of the filter at each frame must be specified.

The idea of DDSP has seen numerous applications in the past few years. Adversarial loss was used with a hierarchical generator network to improve the quality of the output of a DDSP synthesis model [24]. Pitch detection was accomplished by using differentiable DSP in a self-supervised framework [25]. Other synthesis algorithms such as waveshaping synthesis [26] and wavetable synthesis [27] has been introduced to the DDSP framework.

DDSP has seen applications in audio effects as well. An infinite impulse response (IIR) filter was implemented using differentiable DSP and its parameters were trained to emulate a guitar pedal [28]. Similarly, differentiable biquad filters were used for parametric equalizer matching, where optimizing spectral loss was shown to be superior to parameter loss [29]. Mixing parameters such as reverb and panning has been reconstructed from both the wet and dry signal using DDSP-based modules [16].

A line of work similar to DDSP but in the field of images is differentiable rendering, which aims to integrate conventional image rendering into a deep learning framework [30]. Different types of stroke renderers (oil-painting, watercolor, etc.) were implemented in a differentiable manner to produce stylized versions of photos [31]. In another work, 3D attributes of an object were estimated from a 2D image in an end-to-end framework [32]. This network was first trained by a 3D attribute prediction loss using ground-truth labels, and a projection loss with unlabeled data was introduced afterwards.

III. Proposed Method

A. Overview

A diagram of the proposed method is shown in Fig. 2. Two different datasets can be used for training the model: in-domain and out-of-domain (described in detail in Section V-D). Mel-spectrogram is calculated for each sound and fed into the estimator network (Section III-B). The estimated parameters are

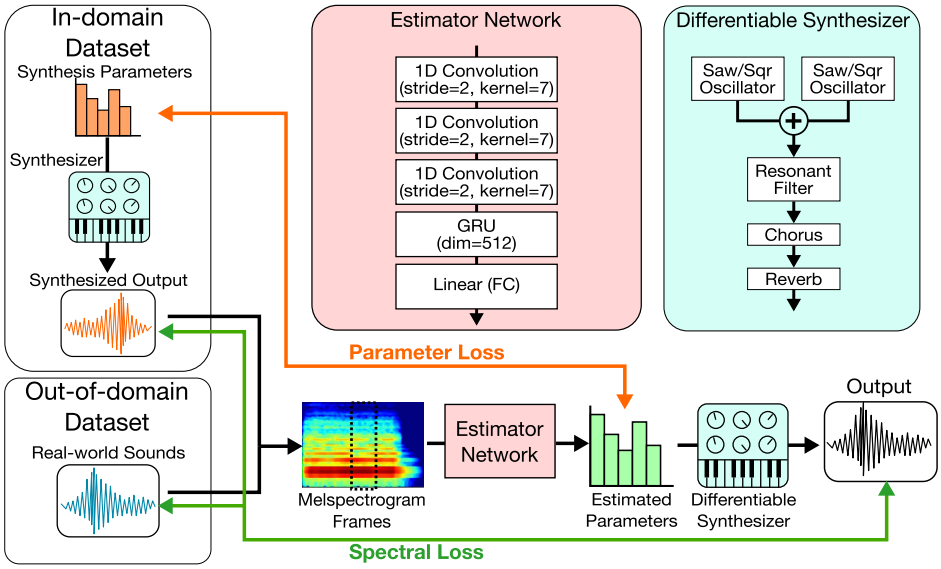


Fig. 2. The architecture of the proposed method. The estimator network outputs the estimated synthesis parameters, which can then be rendered to audio using the differentiable synthesizer.

used for calculating the parameter loss for data in the in-domain dataset whose ground-truth parameters are known. Then, the estimated parameters are fed into the differentiable synthesizer (Section III-C) to render audio. Spectral loss is calculated using the rendered audio and the original audio. Both the parameter loss and the spectral loss are used to optimize the estimator network.

B. Estimator Network

The estimator network takes the melspectrogram of the target sound as input. First, 1D convolutions are applied to the melspectrogram frames over the frequency dimension. This is intended to obtain a representation of the instantaneous timbre of the audio. Then, the output sequence is fed into a gated recurrent unit (GRU). Finally, a linear layer is used to compute the synthesis parameter to be fed into the synthesizer. Since the synthesizer assumes parameter values to be normalized between 0 and 1, a sigmoid function is applied to the output of the linear layer.

Past works have explored the use of 2D convolutions over time-frequency features of audio [2], [14]. In these works, the features are downsampled and a final linear layer is used to calculate a static parameter over the entire sound. On the other hand, our network can estimate time-varying parameters from the outputs at every frame, and also static parameters by using only the last output.

C. Differentiable Synthesizer

A synthesizer much akin to conventional synthesizers used in music production is implemented using differentiable DSP modules in PyTorch. The original DDSP paper [4] used a harmonic-plus-noise model for accurate reconstruction of musical sounds. On the other hand, our task is synthesizer sound matching, i.e., estimation of parameters of a conventional musical synthesizer. So our synthesizer is intentionally limited in terms of synthesis capability but features more interactable parameters such as cutoff frequency. The overall synthesizer structure is shown in Fig. 3.

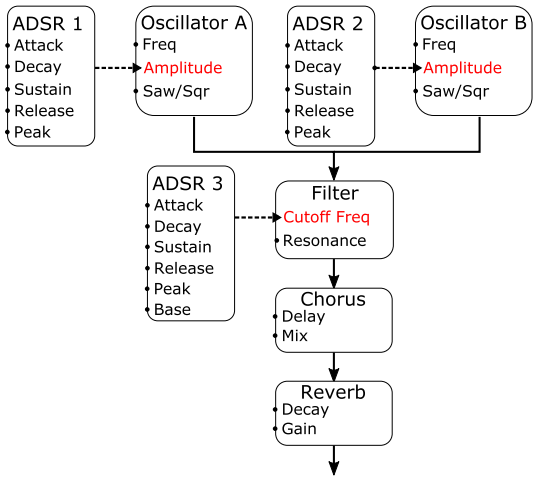


Fig. 3. *The full architecture of the synthesizer.* Dynamic parameters whose values change during the sound are written in red. These dynamic parameters can be modulated by an ADSR envelope.

While this synthesizer has fewer parameters than the harmonics-plus-noise model in the original DDSP, we find that parameter estimation is more difficult for this synthesizer than the harmonics-plus-noise model. We suppose that this is due to the complexity of the relationship between the output sound and synthesis parameters. In the harmonics-plus-noise model, a change in a single parameter only affects certain bins of the output spectrogram, since the parameters correspond to partials of the sound. On the other hand, a single parameter in a typical synthesizer, such as cutoff frequency, can affect many spectral bins, and many parameters can affect the same bin. Another difficulty for parameter estimation is that our synthesizer cannot

accurately recreate the out-of-domain sounds. The estimator must utilize the synthesizer to roughly imitate features of the target sound, which imposes a unique challenge.

1) Main Section: The core of our synthesizer is an *additive-subtractive* synthesis engine that approximates a classical subtractive synthesizer using additive synthesis. This corresponds to the modules Oscillator A/B and Filter in Fig. 3. This design is inspired by popular additive-subtractive synthesizer software such as *Harmor* by Image-Line and *Razor* by Native Instruments. This synthesizer features two oscillators with varying frequency and amplitude. The frequency of the second oscillator unit f_2 is defined by $f_2 = \alpha f_1 (\alpha \geq 1)$. This is to keep the ordering of the oscillators so that the second oscillator is at a higher pitch than the first.

Each oscillator can be interpolated between a sawtooth wave and a square wave. The sawtooth and square wave oscillator with fundamental frequency f can be decomposed into sine waves as follows:

$$x_{\text{sawtooth}}(t) = \frac{2}{\pi} \sum_{k=1}^{\infty} \frac{\sin(k \cdot 2\pi f t)}{k}, \tag{1}$$

$$x_{\text{square}}(t) = \frac{4}{\pi} \sum_{k=1}^{\infty} \frac{\sin\{(2k-1)2\pi f t\}}{2k-1}. \tag{2}$$

The strength of each sinusoid is linearly interpolated between the two extremes by the saw/square wave mix parameter.

Then, the output of two oscillators are mixed and fed into a filter with the same frequency response as that of a 2-pole low-pass filter. This filter alters the timbre by attenuating the harmonics above the cutoff frequency and accentuating the harmonics around the cutoff frequency according to its resonance parameter. While a differentiable implementation of a resonant IIR filter has been proposed [28], IIR involves recurrent computation which is expensive in a deep learning framework. Thus, we efficiently approximate a resonant filter by applying the frequency response as a multiplier to the amplitudes of the harmonics.

The parameters of the main section are as follows: the amplitude, frequency, and saw/square wave mix of each oscillator and the cutoff frequency and resonance of the filter.

2) Effects: We implemented two audio effects commonly found in synthesizers: chorus and reverb. The two effect modules are shown at the bottom of Fig. 3. The output of the filter module of the main section is fed into the chorus and the output of the chorus is fed into the reverb.

The chorus effect is a delay unit with variable delay. This delay amount is modulated by a low-frequency oscillator (LFO) to create an effect similar to pitch vibrato. This delayed signal is mixed with the original signal to create the chorus effect. The variable delay is implemented using differentiable grid sample operations [33]. The *delay* parameter controls the base value of the delay, and the *mix* parameter controls the ratio between the original and delayed signal.

The reverb effect is implemented as finite impulse response (FIR) convolution, similar to the original DDSP [4]. The impulse response is a decaying white noise. The *decay* parameter controls the speed of the decay and *mix* parameter controls how much of the reverb signal is mixed with the original.

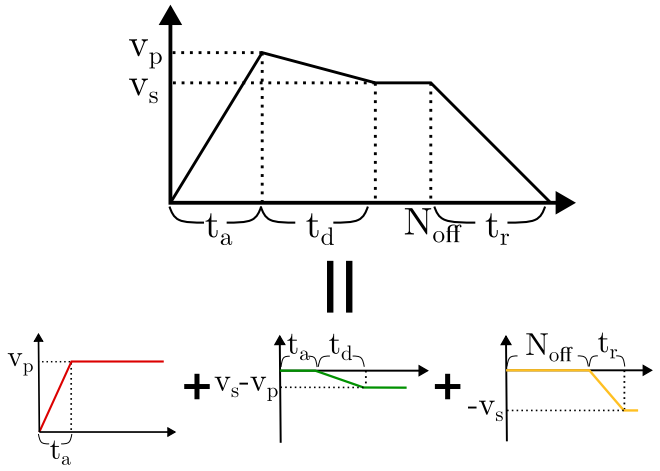


Fig. 4. Diagram of an ADSR envelope (top) and its decomposition (bottom) where the note-on time (start of envelope) and the base value are zero.

3) Modulation: The parameters of the synthesizer can change over time to create movement in a sound. Conventional synthesizers use modulation sources such as ADSR envelope generators to control the movement of some important synthesis parameters.

In this synthesizer, the amplitude of two oscillators and cutoff frequency of the filter are modulated by separate ADSR envelopes (ADSR 1, 2, and 3 in Fig. 3). The parameters of the ADSR envelope are: attack time t_a, decay time t_d, sustain level v_s, release time t_r, base value v_b and peak value v_p. The output of the ADSR envelope starts at base value and rises (or falls) to the peak value and falls (or rises) back to the base value. A diagram of the ADSR envelope is shown in the top of Fig. 4. The attack time parameter dictates the time it takes for the envelope to reach peak level. The decay time parameter controls the length of the decay stage. After the decay stage, the output level is held at a level dictated by the sustain parameter. After the note off point (N_{off}), the release stage, whose length is defined by the release parameter, begins, and the level gradually approaches the base value. For the envelopes controlling the amplitude, the base value is zero to ensure that the sound begins and ends in silence.

We can implement this ADSR envelope in a differentiable and parallelized manner by splitting it into three stages as shown in the bottom of Fig. 4. Assuming that the note on time and base value is zero and $v_p > v_s > 0$, the value $v(t)$ of the ADSR envelope at time t is expressed as follows:

$$v(t) = \left| \frac{v_p}{t_a} t \right|_{[0, v_p]} + \left| \frac{v_s - v_p}{t_d} (t - t_a) \right|_{[v_s - v_p, 0]} + \left| -\frac{v_s}{t_r} (t - N_{off}) \right|_{[-v_s, 0]}, \tag{3}$$

where $|x|_{[a,b]}$ denotes a *clamping* operation equivalent to $\min(\max(x, a), b)$.

D. Loss Function

The estimator network can be trained using both the parameter loss and the spectral loss. The parameter loss is defined as the L1 loss between the estimated parameter and the ground-truth synthesis parameter. All synthesis parameters are normalized to be between 0 and 1.

For the spectral loss, we use a multiscale spectral distance which measures the distance between two sounds using spectrograms calculated in multiple FFT sizes [4]. We adopt a variant where spectral convergence is used along with log-spectrogram loss [22]. This distance is defined as:

$$D_i = \frac{\|S_i(x) - S_i(\hat{x})\|_F}{\|S_i(x)\|_F} + \|\log S_i(x) - \log S_i(\hat{x})\|_1, \quad (4)$$

where $\|\cdot\|_F$ and $\|\cdot\|_1$ are the Frobenius norm and L1 norm respectively, and $S_i(\cdot)$ is the magnitude spectrogram of signal x at a certain FFT size. We use FFT sizes of (64, 128, 256, 512, 1024, 2048) with 75% overlap between frames.

The total loss used to train the estimator network is then:

$$L_{\text{total}} = \alpha \sum_i D_i + \beta L_{\text{param}}, \quad (5)$$

where L_{param} is the parameter loss and α, β are multipliers for each loss.

IV. Synthetic Benchmark Experiment

In order to design our training strategy for the sound matching model, we first evaluate the behavior of the multiscale spectral loss coupled with the differentiable synthesizer in a synthetic benchmark experiment.

A. Experimental Setup

The goal of this experiment is to find out if the spectral loss can provide a useful gradient when a single synthesis parameter needs to be adjusted. We started with a single sound $\boldsymbol{x}$ generated using the synthesizer f in a typical parameter setting denoted as $\boldsymbol{v} = (v_0, v_1, \ldots, v_n)$ for a synthesizer with n parameters. Thus, the synthesis process can be denoted as $\boldsymbol{x} = f(\boldsymbol{v})$. The values for $\boldsymbol{v}$ were chosen to produce a typical sound showcasing most of the functions of the synthesizer. We created variants of this sound by modifying a single parameter so that $\boldsymbol{v}' = (v_0, \ldots, v'_k, \ldots, v_n)$. For this parameter, we performed a sweep over the possible value range. We then calculate the multiscale spectral loss $L(\boldsymbol{x}, \boldsymbol{x}')$ between the original sound $\boldsymbol{x}$ and the modified sound $\boldsymbol{x}' = f(\boldsymbol{v}')$. From this, we can see how the spectral loss relates to changes in a certain parameter. This loss should be close to zero when the modified parameter is close to the original value. We also perform backpropagation through the differentiable synthesizer to calculate the analytic gradient $\frac{dL}{dv'_k}$ with regards to the modified parameter v'_k.

B. Results

We show the results of the synthetic benchmark experiment in Fig. 5. While the synthesizer has 19 parameters in total, we show the results for 11 parameters. For most parameters, the loss curve is smooth, and the gradient is in the red region, indicating that the gradient $\frac{dL}{dv'_k}$ is positive when $v'_k > v_k$ and negative when $v'_k < v_k$. This property indicates that the gradient descent with regards to spectral loss should be effective in tuning the parameter v'_k to the correct value v_k.

However, the loss and the gradient for the oscillator frequency (*OSC1 Freq*, top right of Fig. 5) are more concerning. The loss decreases sharply around the original value, and the gradients are more chaotic, suggesting that spectral loss may not be useful for tuning an oscillator, at least when the frequency value is drastically different from the target frequency. This is in line with the findings of previous work, where spectral loss was found to be ineffective for tuning a pure sine oscillator [34]. They found that this is because a severely out-of-tune oscillator only affects spectral bins away from the correct spectral bins, and the obtained analytic gradient is near zero. In our case, we note that there are some local optima for the loss. This is because when the frequency of one oscillator is a multiple of the other's, some of the harmonic partials align, resulting in lower values for spectral loss.

Another parameter of concern is the delay parameter for the chorus (*Chorus Delay*, left and second from bottom of Fig. 5). The gradient are mostly zero except for the small region near the original value v_k. We suppose that this is due to the chorus delay parameter causing changes in pitch that the spectral loss is not suited for handling, as we saw in the case of oscillator frequency.

TABLE I
Features of the Used Synthesizer Architectures

	Effects	Envelope	External f0	Unordered Oscs
Raw				
Raw-Env		✓		
FX	✓			
FX-Env	✓	✓		
Raw-f0			✓	
FX-f0	✓		✓	
Unordered				✓

V. Sound Matching Setup

We explain the setup of the experiments, including the details of the synthesizer and the estimator network. In V-B, we design a training strategy considering the findings of the synthetic benchmark experiment in Section IV.

A. Synthesizer Architectures

Our main experiments deal with 4 different synthesizer architectures: Raw, Raw-Env, FX, FX-Env. As shown in Table I, only the FX, FX-Env models use the effect modules (chorus and reverb). We experiment with Raw and Raw-Env models in Section VI-A. We experiment with FX and FX-Env models in Section VI-B.

For the Raw and FX models, the envelope is not used during training. Thus, the network estimates the dynamic parameters such as filter cutoff and oscillator amplitude in a frame-by-frame manner. Other static parameters take only one value for the

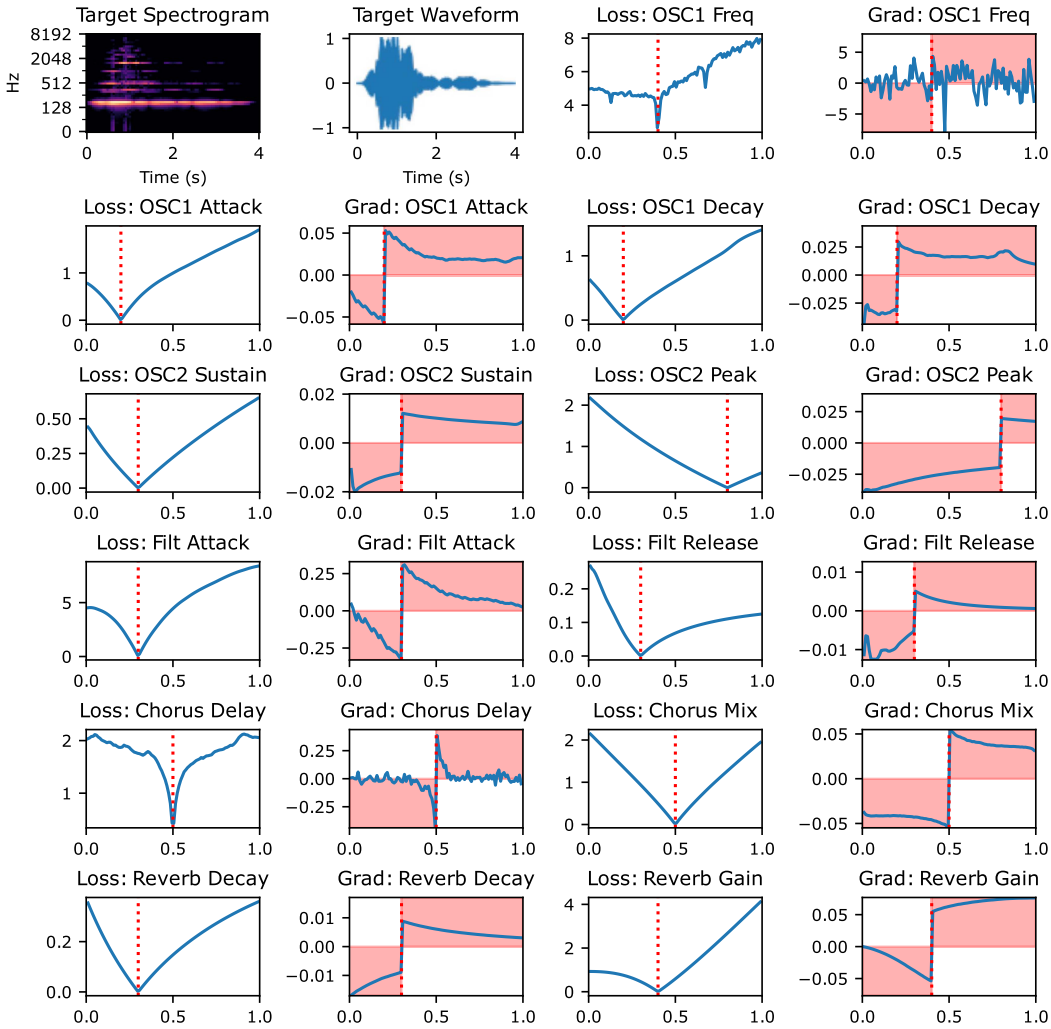


Fig. 5. *Results of the synthetic benchmark experiments.* The red vertical line indicates the original parameter value v_k. In the top left, we show the spectrogram and waveform of the original sound. For each modified parameter, we show the graph of the loss and the analytic gradient.

entire sound. For the Raw-Env and FX-Env models, we use the envelope during training. This means that the network does not estimate dynamic parameters frame-by-frame, but instead estimates the parameters of the envelope.

In Section VI-C, we experiment with synthesizers Raw-f0 and FX-f0, where the frequency of an oscillator is estimated by a separate network for fundamental frequency detection [35].

In Section VI-D, we experiment with the Unordered model, where the frequency of each oscillator is estimated separately. This means that either oscillator 1 or 2 can be higher than the other. This setting is common in many conventional synthesizers, but we hypothesize this may be one of the major obstacles for sound matching.

B. Training Strategy

In Section IV, we saw that spectral loss may be unsuited for estimating the frequency of the oscillator, especially when the distance between estimated frequency and the ground-truth frequency is large. Thus, we adopt a pre-training procedure, where the network is first trained with parameter loss on the in-domain data and then fine-tuned with spectral loss on the out-of-domain data. It is expected that the model learns to estimate the frequency in the pre-training stage.

This can also be seen as a case of semi-supervised learning [36]. The labeled in-domain data is used to train the network to predict parameters that cannot be learned efficiently with the spectral loss. The unlabeled out-of-domain data is used to close the domain-gap between training and inference.

To examine the effect of loss functions and training data, the performance of networks trained using three different training strategies are compared.

- *Parameter-loss only strategy* (hereinafter, denoted as *P-loss*). The network is trained using only parameter loss for 400 epochs. This is in line with conventional NN-based sound matching methods and serves as the baseline of our experiment.
- *In-domain spectral loss strategy* (*Synth*). The network is pre-trained using parameter loss for 50 epochs. For the next 150 epochs, a spectral loss is gradually introduced by increasing the weighting of the spectral loss linearly and decreasing that of the parameter loss (Fig. 6). Finally, the

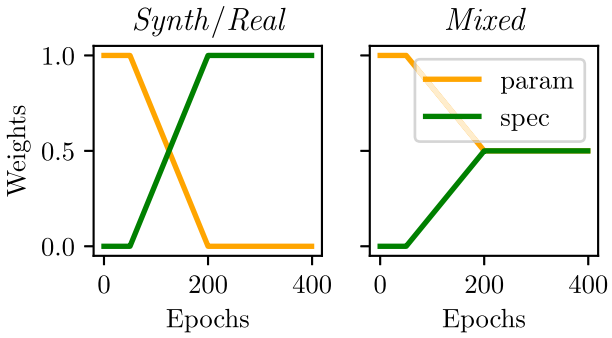


Fig. 6. *Weight scheduling of each loss during training.* Spectral loss is denoted in green, and parameter loss is denoted in orange.

strategy is trained for 200 epochs using only the spectral loss on the in-domain dataset.

- *Out-of-domain spectral loss strategy* (*Real*). This network is trained in the same way as the *Synth* strategy for the first 200 epochs. Then, the strategy is trained for 200 epochs using the spectral loss on the out-of-domain dataset.
- *Mixed strategy* (*Mixed*). For synthesizers with effects, we experiment with an alternative strategy, *Mixed*, where both parameter loss and spectral loss are used for the entirety of the training. This is because the spectral loss seems to be ineffective for learning to control the delay parameter for the chorus effect, as we saw in Section IV. In the *Mixed* strategy, The network is pre-trained in the same way as the *P-loss* model for the first 50 epochs. In the following 150 epochs, the weight of spectral loss is increased to be 0.5 and the weight of parameter loss is annealed to be 0.5. In the last 200 epochs, both the parameter loss and the spectral loss is used until the end (Fig. 6), and 50% of both in-domain and out-of-domain dataset are used as training data. For the out-of-domain dataset, the parameter loss is treated as zero.

Adam optimizer was used with the initial learning rate of 0.005. The learning rate is decreased with an exponential decay rate of 0.99 every epoch (16000 iterations). These hyperparameters were tuned using the *P-loss* model. The network was trained with a batch size of 64. To match the scale of the parameter loss and spectral loss, the L1 parameter loss is multiplied by a factor of 20 during training.

C. Estimator Network

Melspectrogram frames with 128 bands were extracted from the input waveform with an FFT size of 1024 samples and a hop size of 256 samples. Each frame is fed into 3 layers of 1D convolution with batch normalization after each layer. Then, the output is fed into a gated recurrent unit (GRU) layer with 1024 units. Finally, the output of GRU is fed into a linear layer with the same output dimension as the number of parameters for the synthesizer used.

As another baseline, we experimented with the *Conv2D* model composed of 2D convolutional layers that takes spectrogram as input, following the architecture of previous work [14]. This network is trained using only the parameter loss and in-domain data. Kernel size, padding and stride has been adjusted to process longer audio, and batch normalization was added after each convolution layer to improve performance. Since the original paper approached the sound matching problem as a classification problem by discretizing the parameters, they used a softmax output layer. This has been changed to a sigmoid layer to fit with our regression framework. Since this network can only estimate static parameters for an entire sound, it was only tested with Raw-Env and FX-Env architectures.

D. Dataset

1) In-Domain: The in-domain dataset is generated by randomly sampling synthesis parameter settings and rendering them with the same synthesizer used during training. The value of each synthesis parameter is uniformly randomized. This includes parameters for the oscillator and the filter, as well as the parameters for the ADSR envelope. The note-off point triggering the release stage of the envelope is at 3 seconds, and the audio was recorded for 4 seconds. Parameter settings that resulted in silence were removed from the dataset. 20,000 sound-parameter pairs were generated, and partitioned into an 80-10-10 train-validation-test split.

2) Out-of-Domain: For the out-of-domain sounds, the NSynth dataset [21] was used. This dataset includes acoustic and synthetic musical sounds from sample libraries. They were played with MIDI notes in various pitch lasting 3 seconds and recorded for 4 seconds at sampling rate of 16 kHz. 20,000 sounds were randomly selected from the full dataset and partitioned into an 80-10-10 train-validation-test split.

E. Measures for Match Quality

For spectral measures of match quality, we show the log-spectral distortion (denoted as LSD), and mel-cepstral distortion (MCD), and loudness loss (Loud). LSD is a measure between two spectra and is defined as:

$$D_{LS} = \sqrt{\sum_i 10 \log_{10} \frac{S(i)}{\hat{S}(i)}} \tag{6}$$

MCD is a measure commonly used for measuring the quality of voice conversion and is calculated as the euclidean distance of mel-frequency cepstrum coefficients (MFCCs) [37]. When calculating the MFCCs, silent sections were trimmed from the audio, and the 1st MFCC which corresponds to signal power was discarded. The MFCC sequences were aligned using DTW before calculating their distance. Since MFCCs are known to correlate with timbre [38], MCD is expected to be a measure for how close the timbre of two sounds are. Loudness loss is calculated as the average L1 distance between the loudness of the target sound and the output sound at each spectral frame. Loudness was calculated using A-weighting with an FFT size of 2048 and a frame rate of 50 Hz.

For in-domain sounds, we can calculate the L1 parameter loss between the ground-truth parameters of the target sound and the estimated parameters (denoted as Param). Note that all parameters are normalized to be between 0 and 1.

TABLE II
IN-DOMAIN SOUND ESTIMATION RESULTS FOR SYNTHESIZERS WITHOUT EFFECTS

	Param	LSD	Loud (dB)	MCD	Envelope
Raw (P-loss)	**0.045**	11.87	-30.78	**167.6**	
Raw (Synth)	0.061	**11.00**	**-31.02**	176.0	
Raw (Real)	0.165	14.29	-30.81	364.0	
Raw-Env (P-loss)	**0.130**	16.54	-30.07	314.0	✓
Raw-Env (Synth)	0.140	**13.91**	**-30.48**	**274.8**	✓
Raw-Env (Real)	0.245	17.23	-30.32	456.8	✓
Conv2D	0.148	18.27	-29.97	433.8	✓

TABLE III
OUT-OF-DOMAIN SOUND ESTIMATION RESULTS FOR SYNTHESIZERS WITHOUT EFFECTS

	Param	LSD	Loud (dB)	MCD	Envelope
Raw (P-loss)	—	17.75	-28.68	650.9	
Raw (Synth)	—	17.42	-28.82	630.4	
Raw (Real)	—	**12.53**	**-29.23**	**415.7**	
Raw-Env (P-loss)	—	19.42	-27.51	879.5	✓
Raw-Env (Synth)	—	18.71	-28.28	735.1	✓
Raw-Env (Real)	—	**15.12**	**-28.98**	**470.0**	✓
Conv2D	—	19.20	-28.02	816.2	✓

For subjective evaluation of match quality, paired comparison tests were conducted using a crowd-sourcing service. 150 participants in total were gathered indiscriminately, with no requirements of musical experience. It is expected that most participants do not have experience in using synthesizers. We believe that this selection is adequate for a measure of perceptual sound similarity, as one of the main targets of a sound matching system are synthesizer novices. Each participant answered 18 questions. In each question, participants first listen to the target sound with a headphone. Then, they listen to matches produced by two models and answer which match was the closest to the target sound. The order in which the matches are presented was randomized. Out-of-domain sounds were used as the target sound, since they are closer to queries in a real sound matching application than in-domain sounds.

VI. SOUND MATCHING RESULTS

We perform objective and subjective evaluation of the sound matching results and discuss our findings. Audio examples and source code are available on the accompanying webpage[1].

A. Without Effects

The objective measures of sound matching for synthesizers without effects (Raw, Raw-Env) are shown in Tables II and III. For both Raw and Raw-Env synthesizers, the model with *P-loss* strategy performed well for in-domain sounds, but for out-of-domain sounds it performed poorly. The model with *Real* strategy, which used out-of-domain data during training, performed the best against the out-of-domain test set as well. This suggests that the gap between in-domain data and out-of-domain data is significant and should be addressed by domain adaptation. The baseline *Conv2D* model underperformed the proposed network with *P-loss* strategy in all measures. While the *Synth* strategy was inferior to the *P-loss* strategy in terms of parameter loss, it was superior to the *P-loss* strategy in terms of some spectral measures. This is in line with the findings of previous research [2] in that better parameter loss does not always result in better spectral matches.

[1][Online]. Available: https://hyakuchiki.github.io/SSSSM-DDSP/

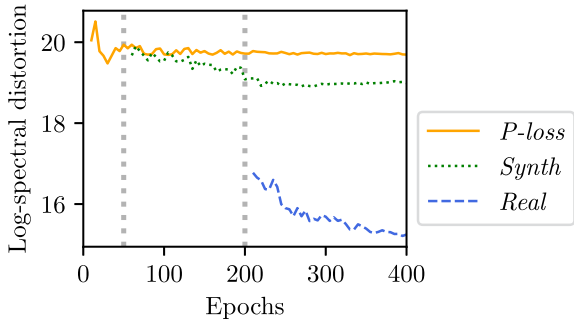


Fig. 7. *Spectral loss on out-of-domain sounds during training estimators for the Raw-Env synthesizer.* The gray lines at 50th epoch and 200th epoch indicate the change in the weighting of the loss and training data.

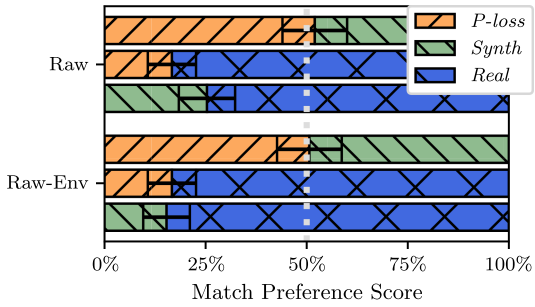


Fig. 8. *Subjective evaluation of the match quality for synthesizers without effects.* The matches produced by the three strategies were compared in a round-robin manner. Error bars denote 95% confidence intervals.

Using the synthesizer with envelope (Raw-Env) resulted in overall worse matches in terms of spectral measures compared to the models with frame-by-frame estimation. This is expected, as the dynamics of the output sound is restricted by the simplistic envelope model.

To examine the behavior of the network in regards to the training procedures, we monitored the LSD for the in-domain and out-of-domain validation set during training. This is shown in Fig. 7. For the *Synth* and *Real* strategies, we can see that introducing the spectral loss from the 50th epoch caused a gradual decrease in LSD. After the 200th epoch, the *Real* strategy used the out-of-domain data for training, and this caused the sharp drop in LSD after this point. From this, we can see the effectiveness of using out-of-domain sounds. The *P-loss* strategy was ineffective in improving spectral loss beyond a certain point.

The results of subjective evaluation is shown in Fig. 8. For both synthesizers, matches produced by *Real* strategy was significantly preferred over those produced by *P-loss* and *Synth* strategy. This result suggests that using out-of-domain sounds during training is crucial for producing perceptually similar matches in a sound matching application. There was no significant difference between the *Synth* and *P-loss* strategies.

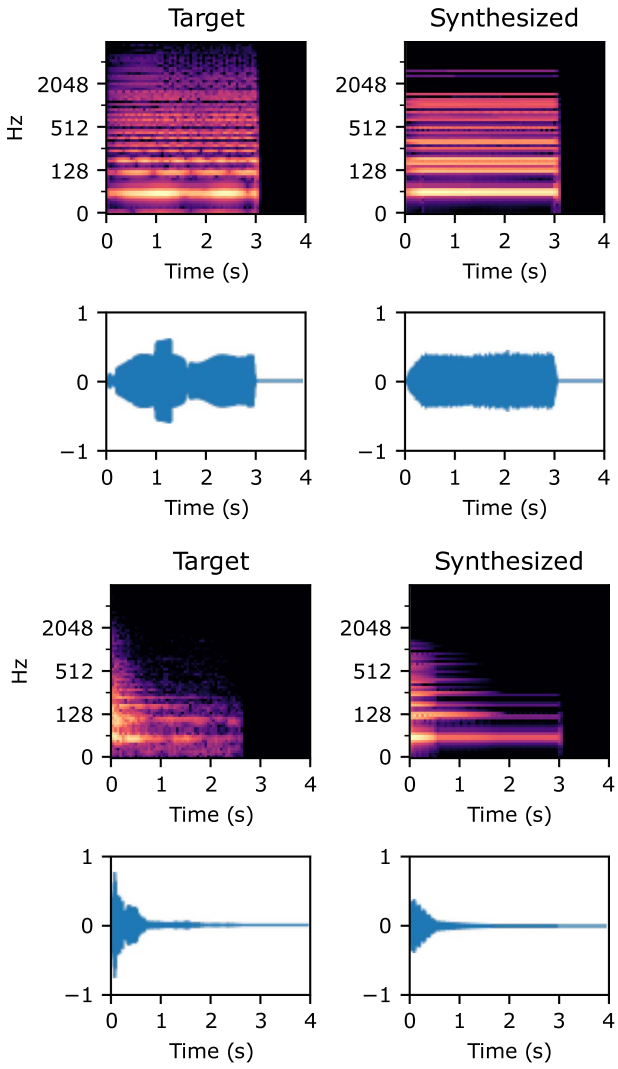


Fig. 9. Examples of sound matching results produced by the Raw-Env synthesizer. The parameters were estimated from out-of-domain sounds from the test set using a model trained with the *Real* strategy.

Examples of sound matching are shown in Fig. 9. The first target sound was a sustained synth lead. Some fluctuations in amplitude could not be produced with a simple ADSR envelope we adopted for the synthesizer, but the model replicated most of the spectral characteristics. The second target sound was a low-pitched orchestral hit, whose complex timbre cannot be reproduced with a synthesizer. The model seemed to produced a slightly detuned bass sound to reproduce some characteristics of the target sound.

B. With Effects

The objective measures of sound matching for synthesizers with effects (FX, FX-Env) are shown in Tables IV and V. The *P-loss* strategy performs the best in most objective measures on the in-domain dataset, but the *Real* strategy performed the best in most objective measures on the out-of-domain dataset.

Upon examining the distribution of the estimated parameters, we found that strategies switching to only spectral loss in the second half of training (*Synth* and *Real*) underutilize the chorus effect. We visualize the distribution of chorus parameters estimated for the in-domain data in Fig. 10. Although all ground-truth parameters are distributed uniformly, *Synth* and *Real* strategies show a very skewed distribution. The mix parameter for the chorus effect remains high for the *Synth* and

TABLE IV
IN-DOMAIN SOUND ESTIMATION RESULTS FOR SYNTHESIZERS WITH EFFECTS

	Param	LSD	Loud (dB)	MCD	Envelope
FX (P-loss)	**0.098**	**15.06**	-27.70	**309.5**	
FX (Synth)	0.278	15.90	**-28.30**	398.5	
FX (Real)	0.286	17.90	-27.83	530.8	
FX (Mixed)	0.122	19.80	-27.90	640.3	
FX-Env (P-loss)	**0.163**	**17.68**	-27.30	**426.6**	✓
FX-Env (Synth)	0.279	18.07	**-28.31**	511.8	✓
FX-Env (Real)	0.315	20.55	-27.77	753.2	✓
FX-Env (Mixed)	0.189	23.34	-27.97	874.1	✓
Conv2D	0.164	19.05	-26.94	527.2	✓

TABLE V
OUT-OF-DOMAIN SOUND ESTIMATION RESULTS FOR SYNTHESIZERS WITH EFFECTS

	Param	LSD	Loud (dB)	MCD	Envelope
FX (P-loss)	—	17.70	-27.67	707.7	
FX (Synth)	—	16.34	**-29.22**	622.2	
FX (Real)	—	**12.91**	-29.07	**410.1**	
FX (Mixed)	—	15.72	-28.92	581.1	
FX-Env (P-loss)	—	19.61	-26.14	877.8	✓
FX-Env (Synth)	—	18.57	-28.72	794.7	✓
FX-Env (Real)	—	**15.47**	-28.87	**507.6**	✓
FX-Env (Mixed)	—	16.44	**-28.89**	624.0	✓
Conv2D	—	18.05	-27.65	858.2	✓

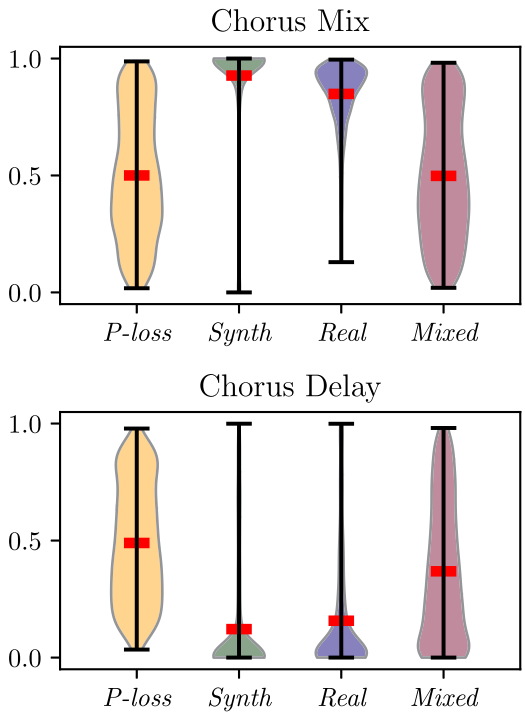


Fig. 10. *Distribution of chorus parameters when estimating for the in-domain test set of the FX-Env synthesizer.* The black bar indicates the extremes of the distribution, while the red line indicates the mean of the parameter distribution.

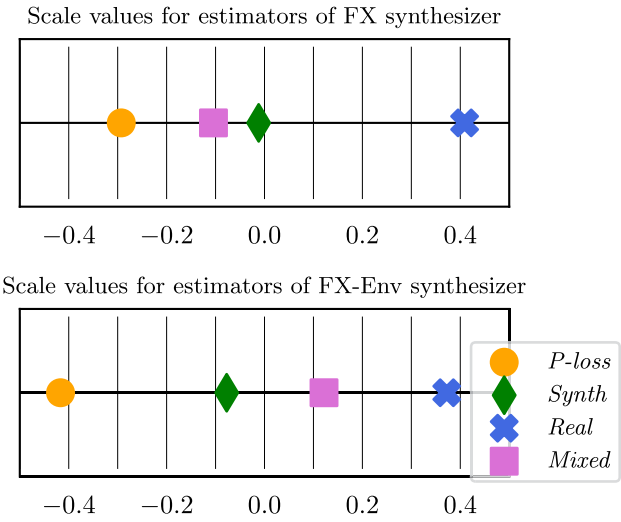


Fig. 11. *Results of subjective evaluation of the match quality for synthesizers with effects.* Scale values were calculated from the paired comparisons using case 5 of Thurstone's method.

TABLE VI
IN-DOMAIN SOUND ESTIMATION RESULTS FOR MODELS WITH FREQUENCY CONDITIONING

	Param	LSD	Loud (dB)	MCD
Raw-f0 (P-loss)	**0.026**	7.67	-29.26	157.1
Raw-f0 (Synth)	0.036	**7.46**	**-29.48**	**153.1**
Raw-f0 (Real)	0.093	7.63	-29.23	197.4
FX-f0 (P-loss)	**0.097**	9.49	-26.71	220.4
FX-f0 (Synth)	0.184	**8.87**	**-28.60**	**195.8**
FX-f0 (Real)	0.204	9.29	-28.22	258.4
FX-f0 (Mixed)	0.100	9.28	-28.04	217.5

TABLE VII
OUT-OF-DOMAIN SOUND ESTIMATION RESULTS FOR MODELS WITH FREQUENCY CONDITIONING

	Param	LSD	Loud (dB)	MCD
Raw-f0 (P-loss)	—	14.65	-27.86	702.3
Raw-f0 (Synth)	—	14.18	-28.15	597.2
Raw-f0 (Real)	—	**12.99**	**-29.18**	**493.1**
FX-f0 (P-loss)	—	15.14	-26.36	773.2
FX-f0 (Synth)	—	14.21	-28.15	641.3
FX-f0 (Real)	—	**12.40**	**-29.23**	**493.1**
FX-f0 (Mixed)	—	13.42	-29.16	522.9

TABLE VIII
IN-DOMAIN SOUND ESTIMATION RESULTS FOR SYNTHESIZERS WITH UNORDERED/ORDERED OSCILLATORS

	Param	LSD	Loud (dB)	MCD
Unordered (P-loss)	**0.111**	18.16	-30.31	509.8
Unordered (Synth)	0.201	**16.15**	**-30.38**	**447.7**
Unordered (Real)	0.202	17.45	-30.23	612.9
Raw (P-loss)	**0.045**	11.87	-30.78	**167.6**
Raw (Synth)	0.061	**11.00**	**-31.02**	176.0
Raw (Real)	0.165	14.29	-30.81	364.0

TABLE IX
OUT-OF-DOMAIN SOUND ESTIMATION RESULTS FOR SYNTHESIZERS WITH UNORDERED/ORDERED OSCILLATORS

	Param	LSD	Loud (dB)	MCD
Unordered (P-loss)	—	18.70	-28.78	776.6
Unordered (Synth)	—	17.46	-29.23	692.6
Unordered (Real)	—	**15.21**	**-28.98**	**576.7**
Raw (P-loss)	—	17.75	-28.68	650.9
Raw (Synth)	—	17.42	-28.82	630.4
Raw (Real)	—	**12.53**	**-29.23**	**415.7**

Real strategies, but the delay parameter is lower compared to the *P-loss* strategy. This makes the audible effects of the chorus module very subtle. This underutilization of the chorus effect seems to be caused by the mismatch between spectral loss and the chorus delay parameter we observed in Section IV. By using the parameter loss throughout training, the *Mixed* strategy is able to keep a uniform distribution of chorus effect parameters, which indicates that the chorus effect is properly utilized.

We show the results of the subjective evaluation in Fig. 11. We see that while the *Real* strategy performed the best, the *Mixed* strategy improves the performance on out-of-domain sounds compared to the *P-loss* schedule.

C. Frequency Conditioning

We experiment with a synthesizer architecture where the frequency of the oscillator is separately estimated by CREPE, a network specialized for pitch detection [35]. This synthesizer features only one oscillator compared to the twin oscillator setup in other synthesizer settings. Specifically, we experiment with a synthesizer without FX (Raw-f0) and with FX (FX-f0).

As shown in Table VI, the *P-loss* model performed the best in terms of parameter loss but *Synth* model performed better in terms of spectral measures and loudness. Also, as shown in Table VII, the *Real* model performed the best in terms of matching out-of-domain sounds. Through an informal subjective evaluation conducted by the authors, we found that the audible differences between each model was smaller than in the case of Raw and FX, since all models provided fairly adequate matches.

While a one-to-one comparison of the these results with their CREPE-less counterpart (Raw, FX) are impossible due to the difference in the synthesizer architecture, separate estimation of frequency seems to be effective for sound matching. However, it would be difficult to apply this approach for estimating polyphonic sounds such as chords, whose fundamental frequency cannot be reliably estimated.

D. With Unordered Oscillators

We experiment with a synthesizer whose oscillator frequencies are unordered; either oscillator can be at a higher pitch than the other. This means that for any synthesizer preset, there is another that sounds exactly the same but with parameters swapped between the two oscillator.

In Tables VIII and IX, we compare the sound matching results for synthesizers with unordered oscillators (Unordered) and ordered oscillators (Raw). The results for Raw is taken from the previous experiment (Section VI-A). We note that the results of Unordered synthesizer is inferior to that of the Raw synthesizer. We suspect that this reduction in performance is because when

the oscillators are unordered, it is unclear which part of the sound should be matched with a particular oscillator. This is similar to the "permutation problem" noted in audio source separation, where sound sources can be assigned to clusters in different ways [39]. This problem may have been present in many previous synthesizer sound matching systems that use synthesizers with unordered oscillators [1], [2].

Even in this setting, the trends seen in other experiments were present. As shown in Table VIII, the *Synth* model was shown to improve performance in terms of spectral measures for in-domain sounds. Also, the *Real* performed the best in terms of out-of-domain sounds (Table IX).

VII. Discussion

A. Utilization of Effects

We believe that underutilization of the chorus effect seen in Section VI-B is concerning, even if the chorus effect is not advantageous for maximizing the perceptual similarity of the match sound to the target sound. Every feature of the synthesizer is important for creating the "signature sound" of the synthesizer, and underutilization of a feature would mean that this signature sound may be lost. It is expected that users may want to produce a good match while still keeping the characteristics of the synthesizer. Synthesizer matching should perhaps be viewed more as a *style transfer* problem.

In Section VI-B, we analyzed the utilization of the chorus effect through the distribution of estimated parameters. A more sophisticated measure for indicating the presence of unique characteristics of the synthesizer is left for future work.

B. Loss Function

In section IV, the gradients of the spectral loss were shown to be ineffective for tuning some parameters such as the frequency of the oscillators. Thus, there maybe more effective measures for match quality than spectral loss. While perceptual distance models using neural audio embeddings [40] may be an interesting alternative, they did not improve match similarity in preliminary experiments we conducted so far.

C. Envelopes

In a conventional synthesizer, dynamic parameters are not specified every frame but rather through ADSR envelope generators for ease of use. Thus, when adopting our method in a conventional synthesizer application, the estimated parameters of the Raw-env and Chorus-Env models can be used directly, but the models with frame-by-frame estimation requires processing to obtain ADSR parameters. In previous work, the frame-wise parameters were smoothed and the split-points were estimated [41].

D. Domain Adaptation

The main advantage of our differentiable synthesizer is that out-of-domain sounds can be used for training. In fact, the *Real* model out-performed others in terms of matching out-of-domain sounds in most settings. From this, we see that the domain gap present in synthesizer sound matching should be addressed if possible.

In the case of sound matching for non-differentiable synthesizers using neural networks, alternative techniques may be effective in closing the domain-gap. Since ground-truth synthesis parameters are unknown for out-of-domain sounds, unsupervised domain adaptation [42] techniques such as domain adversarial training using gradient reversal layers may be used [43].

E. Using Presets

The use of hand-crafted presets as training data may relieve the aforementioned domain gap. Since presets are handcrafted for use in music production and often mimic the sound of other musical instruments, the characteristics of preset sounds should be much closer to real sounds than sounds generated by randomizing the parameters of the synthesizer. In some of the previous work, presets were used as training data, as they dealt with popular synthesizers for which parseable presets are abundantly available (over 10 K) [2], [44]. Our method is effective for learning to control custom-designed synthesizers for which presets are not available.

VIII. Conclusion

In this paper, we presented a novel framework for synthesizer sound matching by implementing a synthesizer using differentiable DSP. This framework works in a semi-supervised manner, utilizing labeled in-domain data and unlabeled out-of-domain data. We showed that the use of out-of-domain sounds as training data is effective for improving the quality of matches against out-of-domain sounds. Since out-of-domain sounds represent the type of sounds that a user will want to match in a sound matching application, our method will be effective in improving the user experience of such application.

References

[1] M. J. Yee-King, L. Fedden, and M. D'Inverno, "Automatic programming of VST sound synthesizers using deep networks and other techniques," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 2, pp. 150–159, Apr. 2018.

[2] P. Esling, N. Masuda, A. Bardet, R. Despres, and A. Chemla-Romeu-Santos, "Universal audio synthesizer control with normalizing flows," in *Proc. Int. Conf. Digit. Audio Effects*, 2019.

[3] N. Masuda and D. Saito, "Synthesizer sound matching with differentiable DSP," in *Proc. Int. Soc. Music Inf. Retrieval Conf.*, 2021, pp. 428–434.

[4] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *Proc. Int. Conf. Learn. Representations*, 2020.

[5] M. Cartwright and B. Pardo, "SynthAssist: Querying an audio synthesizer by vocal imitation," in *Proc. Int. Conf. New Interfaces For Musical Expression*, 2014, pp. 363–366.

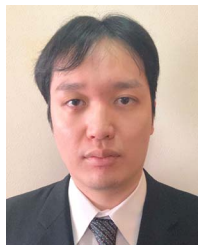
[6] P. Esling and C. Agon, "Multiobjective time series matching for audio classification and retrieval," *IEEE Trans. Audio, Speech Lang. Process.*, vol. 21, no. 10, pp. 2057–2072, Oct. 2013.

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

[8] Y. Lai, S.-K. Jeng, D.-T. Liu, and Y.-C. Liu, "Automated optimization of parameters for FM sound synthesis with genetic algorithms," in *Proc. Workshop Comput. Music Audio Technol.*, 2006, pp. 33–38.

[9] M. Macret, P. Pasquier, and T. Smyth, "Automatic calibration of modified FM synthesis to harmonic sounds using genetic algorithms," in *Proc. 9th Sound Music Comput. Conf.*, 2012.

[10] J. Riionheimo and V. Välimäki, "Parameter Estimation of a Plucked String Synthesis Model Using a Genetic Algorithm with Perceptual Fitness Calculation," *EURASIP J. Appl. Signal Process.*, vol. 8, pp. 791–805, 2003.
[11] N. Masuda and D. Saito, "Quality diversity for synthesizer sound matching," in *Proc. IEEE 24th Int. Conf. Digit. Audio Effects*, 2021, pp. 300–307.
[12] K. Tatar, M. Macret, and P. Pasquier, "Automatic synthesizer preset generation with PresetGen," *J. New Music Res.*, vol. 45, no. 2, pp. 124–144, 2016.
[13] K. Itoyama and H. G. Okuno, "Parameter estimation of virtual musical instrument synthesizers," in *Proc. 40th Int. Comput. Music Conf.*, 2014, pp. 1426–1431.
[14] O. Barkan, D. Tsiris, N. Koenigstein, and O. Katz, "InverSynth: Deep estimation of synthesizer parameter configurations from audio signals," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 11, pp. 2385–2396, Dec. 2019.
[15] D. Sheng and G. Fazekas, "A feature learning siamese model for intelligent control of the dynamic range compressor," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2019, pp. 1–8.
[16] J. T. Colonel and J. Reiss, "Reverse engineering of a recording mix with differentiable digital signal processing," *J. Acoustical Soc. Amer.*, vol. 150, no. 1, pp. 608–619, 2021.
[17] M. A. Martinez Ramirez, O. Wang, P. Smaragdis, and N. J. Bryan, "Differentiable signal processing with black-box audio effects," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2021, pp. 66–70.
[18] A. Défossez, N. Zeghidour, N. Usunier, L. Bottou, and F. Bach, "SING: Symbol-to-instrument neural generator," in *Proc. Adv. Neural Inf., Process. Syst.*, 2018, pp. 9041–9051.
[19] J. W. Kim, R. Bittner, A. Kumar, and J. P. Bello, "Neural music synthesis for flexible timbre control," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 176–180.
[20] A. M. Sarroff and M. Casey, "Musical audio synthesis using autoencoding neural nets," in *Proc. Int. Comput. Music Conf.*, 2014, pp. 1411–1417.
[21] J. Engel et al., "Neural audio synthesis of musical notes with WaveNet autoencoders," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1068–1077.
[22] A. Caillon and P. Esling, "RAVE: A variational autoencoder for fast and high-quality neural audio synthesis," 2021. [Online]. Available: http://arxiv.org/abs/2111.05011
[23] X. Serra, "Musical sound modeling with sinusoids plus noise," in *Musical Signal Processing*, C. Roads, S. Pope, A. Picialli, and G. D. Poli, Eds. Leiden, The Netherland: Swets and Zeitlinger, 1997, pp. 91–122.
[24] M. Michelashvili and L. Wolf, "Hierarchical timbre-painting and articulation generation," in *Proc. Int. Soc. Music Inf. Retrieval Conf.*, 2020, pp. 916–922.
[25] J. Engel, R. Swavely, A. Roberts, L. Hantrakul, and C. Hawthorne, "Self-supervised pitch detection by inverse audio synthesis," in *Proc. Workshop Self-Supervision Audio Speech 37th Int. Conf. Mach. Learn.*, 2020.
[26] B. Hayes, S. Charalampos, and G. Fazekas, "Neural waveshaping synthesis," in *Proc. Int. Soc. Music Inf. Retrieval Conf.*, 2021, pp. 254–261.
[27] S. Shan, L. Hantrakul, J. Chen, M. Avent, and D. Trevelyan, "Differentiable wavetable synthesis," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2022, pp. 4598–4602.
[28] B. Kuznetsov, J. D. Parker, and F. Esqueda, "Differentiable IIR filters for machine learning applications," in *Proc. Int. Conf. Digit. Audio Effects*, 2020, pp. 297–303.
[29] S. Nercessian, "Neural parametric equalizer matching using differentiable biquads," in *Proc. Int. Conf. Digit. Audio Effects*, 2020, pp. 265–272.
[30] H. Kato, Y. Ushiku, and T. Harada, "Neural 3D mesh renderer," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3907–3916.
[31] Z. Zou, T. Shi, S. Qiu, Y. Yuan, and Z. Shi, "Stylized neural painting," in *Proc. IEEE/CVF Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 15689–15698.
[32] S. Yao et al., "3D-aware scene manipulation via inverse graphics," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1887–1898.
[33] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2017–2025.
[34] J. Turian and M. Henry, "I'm sorry for your loss: Spectrally-based audio distances are bad at pitch," in *Proc. 1st I. Can't Believe It's Not Better Workshop*, NeurIPS, 2020.
[35] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "CREPE: A convolutional representation for pitch estimation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 161–165.
[36] J. E. Van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Mach. Learn.*, vol. 109, no. 2, pp. 373–440, 2020.
[37] J. Kominek, T. Schultz, and A. W. Black, "Synthesizer voice quality of new languages," in *Proc. Workshop Spoken Lang. Technol. Under-Resourced Lang.*, 2008, pp. 63–68.
[38] H. Terasawa, M. Slaney, and J. Berger, "Perceptual distance in timbre space," in *Proc. 11th Meeting Int. Conf. Auditory Display*, vol. 5, 2005, pp. 61–68.
[39] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 31–35.
[40] A. L. Cramer, H. H. Wu, J. Salamon, and J. P. Bello, "Look, listen, and learn more: Design choices for deep audio embeddings," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2019, pp. 3852–3856.
[41] K. Jensen, "Envelope model of isolated musical sounds," in *Proc. 2nd COST G-6 Workshop Digit. Audio Effects*, 1999.
[42] G. Wilson and D. J. Cook, "A survey of unsupervised deep domain adaptation," *ACM Trans. Intell. Syst. Technol.*, vol. 11, no. 5, pp. 1–46, Jul. 2020.
[43] Y. Ganin et al., "Domain-adversarial training of neural networks," *J. Mach. Learn. Res.*, vol. 17, no. 59, pp. 1–35, 2016.
[44] G. L. Vaillant, T. Dutoit, and S. Dekeyser, "Improving synthesizer programming from variational autoencoders latent space," in *Proc. IEEE 24th Int. Conf. Digit. Audio Effects*, 2021, pp. 276–283.



Naotake Masuda received the B.E. and M.S.I.T. degrees in 2018 and 2020, respectively, from The University of Tokyo, Tokyo, Japan, where he is currently working toward the Doctor of Engineering degree. He was an Intern with the Institute for Research and Coordination, Acoustics/Music (IRCAM), Paris, France, in 2019, during his stay with Sorbonne University. His research interests include synthesizers, neural audio synthesis and signal processing. He was the recipient of the IPSJ Yamashita SIG Research Award from the Information Processing Society of Japan in 2022.

Daisuke Saito (Member, IEEE) received the B.E., M.S., and Dr. Eng. degrees from The University of Tokyo, Tokyo, Japan, in 2006, 2008, and 2011, respectively. From 2010 to 2011, he was a Research Fellow (DC2) of the Japan Society for the Promotion of Science. He is currently an Associate Professor with the Graduate School of Engineering, The University of Tokyo. His research interests include speech engineering, including voice conversion, speech synthesis, acoustic analysis, speaker recognition, and speech recognition. Dr. Saito is a Member of the *International Speech Communication Association*, *Acoustical Society of Japan*, *Information Processing Society of Japan*, *Institute of Electronics, Information and Communication Engineers*, and *Institute of Image Information and Television Engineers*. He was the recipient of the ISCA Award for the Best Student paper of INTERSPEECH 2011, Awaya Award from the ASJ in 2012, and Itakura Award from ASJ in 2014.