SYNTHESIZER SOUND MATCHING WITH DIFFERENTIABLE DSP

Naotake Masuda **Daisuke Saito**
The University of Tokyo
{n_masuda, dsk_saito}@gavo.t.u-tokyo.ac.jp

ABSTRACT

While synthesizers have become commonplace in music production, many users find it difficult to control the parameters of a synthesizer to create the intended sound. In order to assist the user, the *sound matching* task aims to estimate synthesis parameters that produce a sound closest 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, synthesis parameters are only indirectly correlated with the audio output. Another problem is that query made by the user usually consists of real-world sounds, different from the synthesizer output used during training. In this paper, we propose a novel approach to the problem of synthesizer sound matching by implementing a basic subtractive synthesizer using differentiable DSP modules. This synthesizer has interpretable controls and is similar to those used in music production. We can then train an estimator network by directly optimizing the spectral similarity of the synthesized output. Furthermore, we can train the network on real-world sounds whose ground-truth synthesis parameters are unavailable. We pre-train the network with parameter loss and fine-tune the model with spectral loss using real-world sounds. We show that the proposed method finds better matches compared to baseline models.

1. INTRODUCTION

Synthesizers have become an essential tool in modern music production, owing to their ability to produce a wide array of sounds. The user can directly interact with the parameters of the audio synthesis algorithm to discover interesting sounds. Despite the prevalence of synthesizers in modern music production, most music producers still find it difficult to control the parameters of a synthesizer. The relationships between a synthesis parameter and the perceptual qualities of the output sound is unclear, and complex synthesis algorithms create unexpected outputs. As such, producers often rely on *presets*, parameter settings crafted by sound designers. By using presets, producers can easily incorporate appealing sounds. However, finding an appropriate preset can be difficult, and the sonic palette of the synthesizer is limited by the availability of presets.

Thus, there is great need for user assistance in the sound design process using a synthesizer. One way of assisting the user is automatic programming of the synthesizer to imitate a certain sound. We will refer to this task as *sound matching*. Given a query sound that the user wants to imitate, parameters for a certain synthesizer that produces the best match possible with the synthesizer is estimated. Recently, neural networks have been employed for the sound matching task [1–3]. They are trained to predict the ground-truth synthesis parameter from the synthesized sound, minimizing the error of estimated parameters.

However, the parameter loss maybe a suboptimal loss for optimization. In fact, the model that performs the best in terms of parameter loss does not always return the best match in terms of spectral features [3]. Since we are interested only in the audio match quality, it is better to optimize the network using a loss directly related to the audio. Unfortunately, conventional synthesizers do not allow for backpropagation of the gradients, leaving this problem unaddressed in previous works.

Another problem is that the models in previous works can only be trained on sounds created by the synthesizer (we will refer to them as *in-domain* sounds), as the best matching parameters for sounds not created by the synthesizer (*out-of-domain*) are unknown. In a sound matching application, the system should expect queries consisting of sounds not created by the same synthesizer, originating from acoustic instruments or different synthesizers. Thus, there is a gap in the domain between training and inference, which has been unaddressed by previous works.

In this paper, we address the two problems mentioned above by implementing a synthesizer with interpretable controls using differentiable DSP [4] modules. In our method, the estimator network is optimized in an end-to-end framework including the synthesizer. This allows us to utilize not only parameter loss but also spectral loss, which is more directly related to the audio output of the system. Also, since the ground-truth parameter values are unnecessary when optimizing for spectral loss, the model can be trained using out-of-domain sounds that better represent queries in real-life applications. The proposed model is pre-trained using parameter loss on in-domain data and fine-tuned with spectral loss on out-of-domain data. We show the effectiveness of our method in matching out-of-domain sounds through quantitative measures and subjective evaluations.

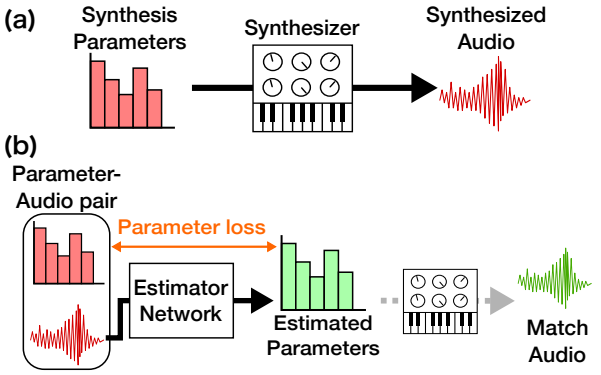

Figure 1. *NN-based synthesizer sound matching.* (a) A synthesizer renders audio according to synthesis parameters. (b) An estimator network is trained to estimate the parameters from the sound, optimizing the parameter loss.

2. RELATED WORKS

2.1 Synthesizer Sound Matching

Synthesizer sound matching is a task that aims to estimate the parameters of a synthesizer to produce a sound similar to the query sound. This supposes an application where the user has a sound that is similar to the desired sound, and queries the system for a parameter setting that produces a similar sound. The user can then adjust the sound further using the synthesizer. This is akin to the *query-by-example* approach in sound retrieval, where a sound that best matches the query sound is retrieved from the database [5]. In fact, sound matching has been realized by retrieval of presets from a database [6]. However, 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.

One of the earliest work in sound matching used genetic algorithm (GA) to match a target spectrum frame [7]. Subsequent works using GA challenged more complex tasks, using conventional synthesizer software to match the entire sound [8]. GA-based sound matching directly optimizes for the similarity of the query sound and the match sound in terms of audio features such as spectral distance. However, GA-based sound matching can take 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 run with population size of 200 for 200 generations would require 40,000 renders of the synthesizer.

Recently, supervised machine learning methods such as multiple linear regression [9] and neural networks (NNs) [1–3] have been used for sound matching. These methods view sound matching as a regression problem, where the synthesis parameters are estimated from audio features. This is illustrated in Figure 1. 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 [10]. However, the gradients obtained for the audio effect are only approximate.

It is also important to consider the actual applications of sound matching. It is expected that users will want to match sounds that were not originally made by the synthesizer. For example, use of vocal imitation as a query for sound matching has been proposed [6]. Perhaps a user will want to imitate acoustic instrument sounds using a synthesizer. While such sounds cannot be matched perfectly, synthesizers can imitate some of their qualities, leading to the discovery of unique sounds. Thus, out-of-domain sounds should be the focus of sound matching. For conventional neural network models trained on pairs of synthesis parameters and the corresponding audio output, such out-of-domain sounds are unseen during training.

2.2 Neural Audio Synthesis

Recently, neural networks have garnered attention as a new way to synthesize musical sounds. Since a typical neural network has millions of model parameters with no interpretability, it is impossible to directly interact with the parameters of a neural audio synthesizer as one would with the synthesis parameters of 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.

SING is a neural audio synthesizer that can be conditioned by the pitch, velocity, and instrument labels [11]. Embeddings for the instrument can be learned to adjust the timbre more flexibly [12]. Alternatively, an autoencoder can be used to learn the latent representation of musical sounds. A standard autoencoder with feedforward layers was used to reconstruct spectral frames [13]. A WaveNet autoencoder can be used to model raw audio of musical sounds [14]. Autoencoder models encode the audio into a compact representation and decode it to reconstruct the audio. By modifying this representation, the output sound can be controlled.

Compared to conventional synthesizers, neural audio synthesizers can potentially create more realistic sounds and offer a novel way to control musical sounds. However, they do not provide full control to the user over the synthesis process, and their use in practical music production has been limited so far.

2.3 Differentiable DSP

While neural audio synthesizers aim to generate raw audio using only neural networks, differentiable digital signal processing (DDSP) aims to integrate conventional signal processing elements with deep learning [4]. The parameters of a differentiable audio synthesis model are estimated by a neural network in an end-to-end manner. In the original DDSP paper, a differentiable version of an additive synthesis model called the harmonics-plus-noise model is used to generate audio. This is a variant of the sinusoids-plus-noise model [15]. While the harmonics-plus-noise

model can be considered as a synthesizer, it is much more complicated than conventional synthesizers, as the amplitude of each harmonic and the full frequency response of the filter must be specified. The harmonics-plus-noise synthesizer allows for accurate reconstruction of real instrument sounds, but the synthesis parameters are far too many to allow for direct interaction.

The idea of DDSP has inspired a handful of works. Adversarial loss was used with a hierarchical generator network to improve the quality of the output [16]. Pitch detection of musical signals was accomplished by using differentiable DSP in a self-supervised framework [17]. New differentiable DSP modules have been proposed as well. An infinite impulse response (IIR) filter was implemented using differentiable DSP and its parameters were trained to emulate a guitar pedal [18]. Similarly, differentiable biquad filters were used for parametric equalizer matching, where optimizing spectral loss was shown to be superior to parameter loss [19]. Our work expands this idea to the synthesizer sound matching problem.

Parallels can be drawn between differentiable DSP and differentiable rendering. Differentiable rendering aims to integrate rendering of 3D objects into a deep learning framework [20]. 3D attributes of an object were estimated from a 2D image in an end-to-end framework [21]. This network was first trained by a 3D attribute prediction loss using ground-truth labels, and a projection loss using a renderer was introduced afterwards. This is similar to our training procedure, in which the network is pre-trained by synthesis parameter estimation loss, and spectral loss using the differentiable synthesizer is introduced afterwards.

3. PROPOSED METHOD

3.1 Overview

A diagram of the proposed method is shown in Figure 2. Melspectrogram frames calculated from the audio are fed into a neural network to estimate the synthesis parameters frame-by-frame. The in-domain dataset consists of synthesis parameters and the synthesized sound. This is generated by random sampling of synthesis parameters. For in-domain sounds, the parameter estimation loss is calculated between the estimated and the ground-truth synthesis parameters, in a similar manner to conventional NN-based synthesizer sound matching. Finally, a differentiable synthesizer is used to render the audio from the synthesis parameters. This allows for end-to-end training using spectral loss, for both in-domain and out-of-domain sounds. By using out-of-domain sounds for training, it is expected that our proposed method will be better able to generalize to actual queries consisting of real-world sounds.

3.2 Differentiable Synthesizer

An *additive-subtractive* synthesizer that approximates a classical subtractive synthesizer using additive synthesis was implemented in PyTorch. This design is inspired by popular additive-subtractive synthesizer software such as *Harmor* by Image-Line or *Razor* by Native Instruments. This synthesizer features two oscillators with varying pitch and amplitude. The waveform of each oscillator can be interpolated between a sawtooth wave and a square wave. Each oscillator is implemented in an additive way, meaning that sine oscillators with different frequencies are added up to create a waveform with richer harmonic. More specifically, sawtooth waveform and square waveform with fundamental frequency f can be decomposed into sine waves as follows:

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

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

The output of two oscillators are mixed and fed into a resonant low-pass filter. This filter can alter the timbre by attenuating the harmonics above the cutoff frequency and accentuating the harmonics around the cutoff frequency according to its resonance parameter. While previous works has proposed a differentiable implementation of a resonant IIR filter [18], IIR inherently involves recurrent computation which is computationally expensive. Thus, we approximate a resonant filter by applying the frequency response of the filter as a multiplier to the amplitudes of the harmonics. Ultimately, the parameters of this synthesizer are as follows: amplitude, frequency and saw/square wave mix of each oscillator and the cutoff frequency and resonance of the filter.

The parameters of the synthesizer can change over time to create movement in a sound. Conventional synthesizers use envelope generators and low-frequency oscillators (LFOs) to control the modulation of some important synthesis parameters. In our experiments, the amplitudes of each oscillator and the cutoff frequency of the filter were estimated frame-by-frame, and other parameters were set to a single value which was estimated from the last output of the GRU.

Our method aims to assist the user in controlling a practical synthesizer, so we implemented a differentiable synthesizer with familiar controls such as cutoff frequency. While this synthesizer has fewer parameters compared to 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 indirectness of the relationship between the synthesis parameters and the output spectrum. A single synthesis parameter in the harmonics-plus-noise model roughly corresponds to a single spectrogram time-frequency component of the output, especially when it is conditioned by the fundamental frequency. On the other hand, a synthesis parameter in the proposed synthesizer model can affect many components, and its effects are dependent on each other. Furthermore, our synthesizer is intentionally limited in terms of the sounds it can create. The estimator must utilize the synthesizer to roughly imitate features of the target sound, which imposes a unique challenge.

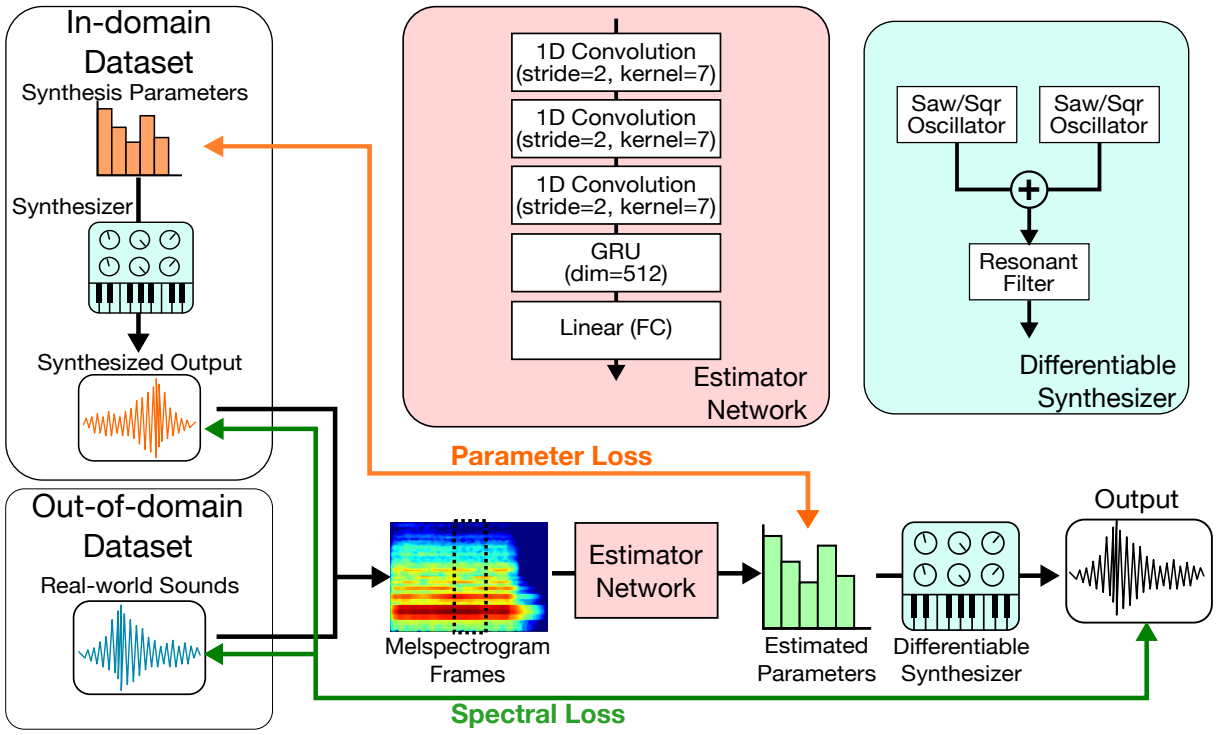

Figure 2. *The architecture of the proposed model.* Two different losses can be calculated in this framework: the parameter loss and the spectral loss. Parameter loss can be calculated only for in-domain data, whose ground-truth synthesis parameters are available.

3.3 Training

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. The spectral loss is a multi-scale spectrogram loss [4], which is defined as the sum of L1 loss of spectrograms and log-spectrograms in multiple resolutions. In the experiments, we use FFT sizes of (64, 128, 256, 512, 1024, 2048). Frames were overlapped at 75% with the next frame.

During preliminary experiments, we found that training the model with only spectral loss was ineffective. We suppose that this is due to the indirectness of the relationship between the synthesis parameters and the output spectrum. We found that pre-training the model by parameter loss and fine-tuning the model using spectral loss was the most effective. More specifically, our training procedure can be split into three steps. First, the network is trained by parameter loss on the in-domain dataset. Next, spectral loss is gradually introduced and eventually replaces the parameter loss completely. Finally, the model is trained using the out-of-domain dataset.

This final step can be considered as a form of *domain adaptation*. Specifically, unsupervised domain adaptation aims to transfer the knowledge of labeled source domain to a target domain with no labels [22]. While ground-truth parameter values are unavailable for the out-of-domain sounds, we can transfer the knowledge of models trained using in-domain sounds to out-of-domain sounds by switching to the spectral loss.

4. EXPERIMENT SETUP

4.1 Training Procedure

The proposed method aims to improve the quality of sound matching by use of spectral loss and adaptation to out-of-domain data. To examine the effect of each, the performance of models trained using three different training procedures are compared.

- *Parameter-loss only model* (hereinafter, denoted as *P-loss*). This model 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 model* (*Synth*). This model 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. Finally, the model is trained for 200 epochs using only the spectral loss on the in-domain dataset.

- *Out-of-domain spectral loss model* (*Real*). This model is trained in the same way as the *Synth* model for the first 200 epochs. Then, the model is trained for 200 epochs using the spectral loss on the out-of-domain dataset.

The learning rate is decreased with an exponential decay rate of 0.99 every epoch (16000 iterations). To match

the scale of the parameter loss and spectral loss, the L1 parameter loss is multiplied by a factor of 10 during training. The network was trained with a batch size of 64.

4.2 Estimator Network

The estimator network is trained to predict the synthesis parameter at each time step from the melspectrogram of input audio. 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 to obtain a high-level representation of spectral features. Then, the output is fed into a gated recurrent unit (GRU) layer. Finally, the output of GRU is fed into a linear layer. Since all synthesis parameters are normalized to be within $(0, 1)$, sigmoid nonlinearity is applied to the network output.

4.3 Dataset

4.3.1 In-domain

The in-domain dataset is generated by randomly sampling synthesis parameter settings and rendering them with the same synthesizer used in the model. The value of a static parameter is uniformly randomized. The temporal evolution of a dynamic parameter is modelled by an attack-decay-sustain-release (ADSR) envelope generator used in most conventional synthesizers. The parameters of this envelope generator are attack time, decay time, sustain level, release time and the peak/floor levels. These envelope parameters are uniformly randomized for each dynamic parameter. Gaussian noise is added to the output of this envelope generator to model the fluctuation present in real-world sounds. 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.

4.3.2 Out-of-domain

For the out-of-domain sounds, the NSynth dataset [14] 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 16kHz. 20,000 sounds were randomly selected from the full dataset and partitioned into an 80-10-10 train-validation-test split.

5. 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] .

[1] https://hyakuchiki.github.io/DiffSynthISMIR/

	In-domain			Out-of-domain	
Models	Param	LSD	Multi	LSD	Multi
P-loss	**0.065**	16.14	4.72	19.60	8.84
Synth	0.083	**14.38**	**3.37**	19.13	5.79
Real	0.177	15.35	3.87	**17.27**	**3.90**

Table 1. *Objective measures of sound matching* (Param: L1 parameter loss, LSD: log-spectral distortion, Multi: multi-scale spectrogram loss). Smaller values indicate better performance.

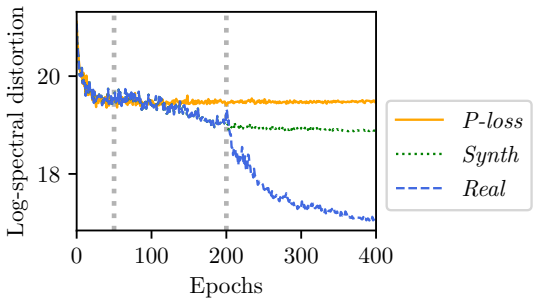

Figure 3. *Spectral loss on out-of-domain sounds during training.* The gray lines at 50th epoch and 200th epoch indicate the change in the weighting of the loss.

5.1 Quantitative Results

As a quantitative measure of the quality of sound match, we use log-spectral distortion (*LSD*) and the multi-scale spectral loss (*Multi*). We also calculate the L1 parameter loss (*Param*) for in-domain sounds. The results are shown in Table 1. The *P-loss* model achieved the best performance in terms of parameter loss, but performed poorly compared to other models in terms of spectral measures. This suggests that parameter loss is an inadequate criterion for match quality. The *Synth* model performed the best for matching the spectra of in-domain sounds, but the *Real* model was superior for out-of-domain data. This result shows that the fine-tuning was effective in transferring the knowledge learned from in-domain sounds to out-of-domain sounds. Since the out-of-domain data better represents query sounds in real-life applications, the *Real* model is the most promising for sound matching.

To examine the effects of the training procedures, we monitored the LSD for the out-of-domain validation set during training. This is shown in Figure 3. We can see that introducing the spectral loss from the 50th epoch caused a gradual decrease in LSD for the models *Synth* and *Real*. After the 200th epoch, the *Real* model was trained using out-of-domain data. From the sharp drop in LSD after this point, we can see the effectiveness of using out-of-domain sounds. The *P-loss* model was ineffective in improving spectral loss beyond a certain point.

5.2 Subjective Evaluation

For subjective evaluation of the match quality, paired comparison was conducted with reference stimuli via a crowd

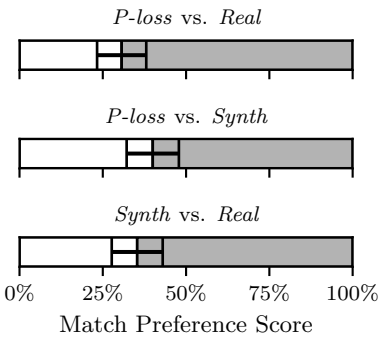

Figure 4. *Results of subjective evaluation of the match quality.* The three models were compared in a round-robin manner. Error bars denote 95% confidence intervals.

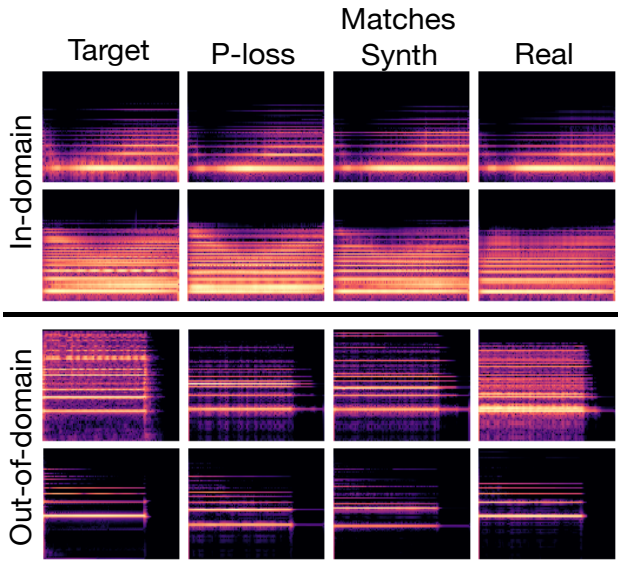

Figure 5. *Examples of sound matching results.* The three models were used to match the same target sounds taken from the in-domain and out-of-domain test set. We display the spectrogram (frequency is log-scaled, 0-8000Hz) of the audio output.

sourcing system. In total, 25 listeners from various backgrounds answered 18 questions. In each question of the test, listeners were exposed to the original target sound and the matches created by two models. The listeners answered which match sounded more similar to the target sound. The order in which the two matches were presented was randomized. The target sounds were randomly chosen from the test set of the out-of-domain data, as they better represent queries in real-life applications of sound matching than in-domain sounds.

Results of the preference test is shown in Figure 4. First, the *P-loss* model was compared with the *Real* model. The audio outputs of the *Real* model was preferred more frequently than the *P-loss* model, indicating that the proposed method of using spectral loss and using out-of-domain sounds during training was effective in producing perceptually better matches, compared to the conventional method of optimizing only parameter loss. Second, the models *P-loss* and *Synth* were compared. The *Synth* model performed better than the *P-loss* model, indicating that the use of spectral loss was effective in producing perceptually better matches. Finally, the models *Real* and *Synth* were compared. The *Real* model performed better than the *Synth* model. This highlights the importance of fine-tuning the estimator model with data that more closely resembles the query.

We show examples of sound matching in Figure 5. We can note that while all models perform comparably well on most in-domain sounds, the *P-loss* model tended to fail in reproducing features such as pitch and spectral envelope of the the out-of-domain sounds. The first out-of-domain sound in Figure 5 is a brass sound with rich harmonics, but only the *Real* model successfully produced timbre resembling a brass sound. For the second out-of-domain sound, the *P-loss* and *Synth* models failed to estimate the pitch, resulting in a sound with lower pitch.

6. CONCLUSIONS AND FUTURE DIRECTIONS

We presented a novel method of synthesizer sound matching by implementing the synthesizer using differentiable DSP. The proposed method is able to directly optimize spectral loss in an end-to-end manner. Furthermore, the model is able to utilize real-world sounds during training. By pre-training the model with parameter loss on the synthetic data and fine-tuning on real-world sounds with spectral loss, we showed that the proposed method can provide perceptually better matches to real-world sounds compared to baseline models.

While dynamic parameters were estimated frame-by-frame in our experiments, typical synthesizers use ADSR and LFO modules to model the dynamics of parameters. Such modules are required for intuitive control of the dynamics and playing notes with different length, but the use of such modules has been left unaddressed by our work. One solution is to estimate the envelope parameters from the frame-wise synthesis parameters [23]. Another solution is to implement such modules in a differentiable manner and use them during training. Preliminary experiments using differentiable envelope modules yielded promising results, although the match is less accurate for real-world sounds due to the simplification of the dynamics.

Another direction is to experiment with different synthesis techniques and audio effects. Preliminary experiments on using spectral loss to estimate the parameters of an FM synthesizer was shown to be less successful. This may be due to the fact that FM synthesis creates many inharmonic partials resulting in a complex spectrum. Recent research suggests that deep audio embeddings may be a better distance metric than multi-scale spectral loss for complex synthesizer sounds [24]. Such alternative criteria for the match quality is worth considering not only for improving the estimation quality, but also for providing unique matches that capture a certain feature of the query.

7. REFERENCES

[1] O. Barkan, D. Tsiris, N. Koenigstein, and O. Katz, "InverSynth: Deep estimation of synthesizer parameter configurations from audio signals," *IEEE/ACM Trans. on Audio, Speech, and Language Processing*, vol. 27, no. 11, pp. 2385–2396, 2019.

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

[3] P. Esling, N. Masuda, A. Bardet, R. Despres, and A. Chemla-Romeu-Santos, "Universal audio synthesizer control with normalizing flows," in *Proc. of the 22nd Int. Conf. on Digital Audio Effects*, 2019.

[4] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP:Differentiable Digital Signal Processing," in *Proc. of the Int. Conf. on Learning Representations*, 2020.

[5] P. Esling and C. Agon, "Multiobjective time series matching for audio classification and retrieval," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 21, no. 10, pp. 2057–2072, oct 2013.

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

[7] 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, mar 1993.

[8] K. Tatar, M. Macret, and P. Pasquier, "Automatic synthesizer preset generation with PresetGen," *Journal of New Music Research*, vol. 45, no. 2, pp. 124–144, 2016.

[9] K. Itoyama and H. G. Okuno, "Parameter estimation of virtual musical instrument synthesizers," in *Proc. of the 40th Int. Computer Music Conf.*, 2014, pp. 1426–1431.

[10] M. A. Martinez Ramirez, O. Wang, P. Smaragdis, and N. J. Bryan, "Differentiable signal processing with black-box audio effects," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2021, pp. 66–70.

[11] A. Défossez, N. Zeghidour, N. Usunier, L. Bottou, and F. Bach, "SING: Symbol-to-instrument neural generator," in *Advances in Neural Information Processing Systems*, 2018, pp. 9041–9051.

[12] J. W. Kim, R. Bittner, A. Kumar, and J. P. Bello, "Neural music synthesis for flexible timbre control," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2019, pp. 176–180.

[13] A. M. Sarroff and M. Casey, "Musical audio synthesis using autoencoding neural nets," *Proc. of the 40th Int. Computer Music Conf., ICMC*, pp. 1411–1417, 2014.

[14] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi, "Neural audio synthesis of musical notes with WaveNet autoencoders," in *Proc. of the 34th Int. Conf. on Machine Learning*, 2017, pp. 1068–1077.

[15] X. Serra, "Musical sound modeling with sinusoids plus noise," in *Musical Signal Processing*, C. Roads, S. Pope, A. Picialli, and G. D. Poli, Eds., 1997, pp. 91–122.

[16] M. Michelashvili and L. Wolf, "Hierarchical timbre-painting and articulation generation," in *Proc. of the Int. Society for Music Information Retrieval Conf.*, 2020, pp. 916–922.

[17] J. Engel, R. Swavely, A. Roberts, L. Hantrakul, and C. Hawthorne, "Self-supervised pitch detection by inverse audio synthesis," in *Workshop on Self-Supervision in Audio and Speech at the 37th Int. Conf. on Machine Learning*, 2020.

[18] B. Kuznetsov, J. D. Parker, and F. Esqueda, "Differentiable IIR filters for machine learning applications," in *Proc. of the 23rd Int. Conf. on Digital Audio Effects*, 2020, pp. 297–303.

[19] S. Nercessian, "Neural parametric equalizer matching using differentiable biquads," in *Proc. of the 23rd Int. Conf. on Digital Audio Effects*, 2020, pp. 265–272.

[20] H. Kato, Y. Ushiku, and T. Harada, "Neural 3D mesh renderer," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 3907–3916.

[21] S. Yao, T. M. H. Hsu, J. Y. Zhu, J. Wu, A. Torralba, W. T. Freeman, and J. B. Tenenbaum, "3D-aware scene manipulation via inverse graphics," in *Advances in Neural Information Processing Systems*, 2018, pp. 1887–1898.

[22] G. Wilson and D. J. Cook, "A survey of unsupervised deep domain adaptation," *ACM Trans. on Intelligent Systems and Technology*, vol. 11, no. 5, Jul. 2020.

[23] K. Jensen, "Envelope model of isolated musical sounds," in *Proc. of 2nd COST G-6 Workshop on Digital Audio Effects*, 1999.

[24] J. Turian, J. Shier, G. Tzanetakis, K. McNally, and M. Henry, "One billion audio sounds from GPU-enabled modular synthesis," *arXiv preprint arXiv:2104.12922*, 2021.