SynthRL: Cross-domain Synthesizer Sound Matching via Reinforcement Learning

Wonchul Shin and **Kyogu Lee**
Seoul National University
{swc0406, kglee}@snu.ac.kr

Abstract

Generalization of synthesizer sound matching to external instrument sounds is highly challenging due to the non-differentiability of sound synthesis process which prohibits the use of out-of-domain sounds for training with synthesis parameter loss. We propose SynthRL, a novel reinforcement learning (RL)-based approach for cross-domain synthesizer sound matching. By incorporating sound similarity into the reward function, SynthRL effectively optimizes synthesis parameters without ground-truth labels, allowing fine-tuning on out-of-domain sounds. Furthermore, we introduce a transformer-based model architecture and reward-based prioritized experience replay to enhance RL training efficiency, considering the unique characteristics of the task. Experimental results demonstrate that SynthRL outperforms state-of-the-art methods on both in-domain and out-of-domain tasks. Further experimental analysis validates the effectiveness of our reward design, showing a strong correlation with human perception of sound similarity.

1 Introduction

In modern music production, sound design plays a critical role in shaping the auditory identity of a track, enabling producers to express their creativity and evoke specific emotions in listeners. A key tool in sound design is the synthesizer, which allows producers to generate and manipulate a vast variety of sounds by combining various waveforms, filters, and modulation techniques. However, crafting the desired sound with a synthesizer is a complex process, requiring significant study, effort, and time to master the manipulation of its parameters. Furthermore, synthesizers differ in the methods they use to produce sounds, such as additive, subtractive, and frequency modulation (FM) synthesis, meaning that even experienced users may find it challenging to use different types of synthesizers proficiently [Powell, 1987]. Consequently, there has been growing research focused on developing methods to automatically search for synthesizer parameter settings that can replicate the timbre of sounds from other songs or sound samples [Sebastian *et al.*, 2009; Roth and Yee-King, 2011; Tatar *et al.*, 2016]. Using a model that has learned this task called *synthesizer sound matching*, even beginners can easily create the desired sound through a synthesizer and further adjust parameters to refine the sounds.

With advancements in deep learning, recent studies have proposed methods utilizing deep neural networks to estimate synthesizer parameters for desired sound inputs [Yee-King *et al.*, 2018; Barkan *et al.*, 2019; Vaillant *et al.*, 2021; Chen *et al.*, 2022]. One of the primary challenges in these studies is that the sound generation process in synthesizers is non-differentiable. In other words, it is infeasible to incorporate the sound generated by the synthesizer based on the estimated parameters into the differentiable loss function for backpropagation, such as the spectrogram difference from the ground truth sound. Therefore, previous studies have focused on training the model using only a parameter loss that minimizes the error between the ground truth synthesis parameters and the estimated parameters. However, since the goal is to generate a sound that closely resembles the input sound, the inability to directly use the perceptual difference between the ground truth sound and the generated sound in the learning process represents a significant limitation. In addition, using only the parameter loss requires the availability of ground truth parameters, meaning that only sounds generated from known synthesizer parameters (*in-domain*) can be used as training data. For example, users often aim to replicate sounds sourced from a different type of synthesizer or instrument (*out-of-domain*) using their own or more familiar synthesizer. However, since out-of-domain sounds cannot be used to calculate parameter loss, models trained with only in-domain sounds may perform poorly when encountering these unfamiliar sounds.

A prominent line of research to address these limitations involves the implementation of synthesizers using differentiable digital signal processing (DDSP) modules [Masuda and Saito, 2021; Caspe *et al.*, 2022; Uzrad *et al.*, 2024]. The sound synthesis process of a DDSP synthesizer is fully differentiable, allowing the generated sound to be used for training through backpropagation. Nonetheless, this approach confines the application of the sound matching model to specific DDSP synthesizers, hindering users from applying the model to their preferred synthesizers. Moreover, implementing a wide variety of conventional synthesizers in a differentiable manner requires considerable effort for each synthesizer, due

to the distinct and complex sound synthesis techniques they employ.

To address the aforementioned issues in synthesizer sound matching, we propose a novel approach using reinforcement learning (RL), named SynthRL. SynthRL employs the generated sound in the reward function of RL, enabling the model to optimize for greater similarity to the ground truth sound. Consequently, our approach facilitates the incorporation of perceptual loss into the training of non-differentiable synthesizer, as well as the use of out-of-domain sounds. We also demonstrate that reward-based prioritized experience replay (PER) [Schaul, 2015], tailored to the unique characteristics of the synthesizer sound matching task, is crucial for the effective application of RL. Moreover, while our primary focus lies in the application of RL, we also introduce an effective synthesizer sound matching model architecture based on transformer encoder-decoder framework. Through extensive experiments, we demonstrate that our method outperforms existing baselines on both in-domain and out-of-domain using a widely used conventional synthesizer.

2 Related Works

2.1 Synthesizer Sound Matching

The complexity and difficulty of operating synthesizers have motivated considerable efforts to automatically estimate the parameter sets required to create desired sounds. Early studies employed search-based optimization methods such as genetic algorithms [Horner *et al.*, 1993; Garcia, 2002; Mitchell and Sullivan, 2005; Yee-King and Roth, 2008; Tatar *et al.*, 2016] or Particle Swarm Optimization [Sebastian *et al.*, 2009].

More recent advancements have leveraged neural networks, where models are trained to minimize the error between the ground truth parameters and the estimated parameters. For instance, FlowSynthesizer [Esling *et al.*, 2019] integrates an autoencoder with normalizing flows, followed by PresetGenVAE [Vaillant *et al.*, 2021], which further advances similar methods. Sound2Synth [Chen *et al.*, 2022], a previous state-of-the-art (SOTA) method, extracts a diverse set of acoustic features and integrates them as model inputs. These approaches show promising results for in-domain sounds.

The non-differentiability of synthesizers has spurred extensive research into the development of differentiable neural synthesizers [Engel *et al.*, 2020; Shan *et al.*, 2022; Renault *et al.*, 2022; Wiggins and Kim, 2023]. In line with these studies, previous research addressing cross-domain synthesizer sound matching has predominantly employed approaches that develop differentiable synthesizers [Masuda and Saito, 2021; Caspe *et al.*, 2022; Uzrad *et al.*, 2024]. Additionally, InverSynth II [Barkan *et al.*, 2023] proposed a differentiable proxy model of the synthesizer and a inference-time finetuning strategy for improved in-domain sound matching.

To the best of our knowledge, there has been no research focused on cross-domain sound matching for conventional non-differentiable synthesizers so far. In this work, we introduce RL to synthesizer sound matching for the first time, providing a framework that facilitates performance improvements across both in-domain and out-of-domain sounds.

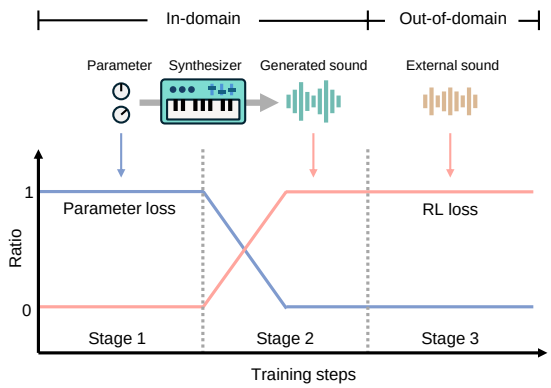


Figure 1: The training pipeline of SynthRL consists of three stages: (1) in-domain training with only parameter loss, (2) gradual introduction of RL loss while reducing parameter loss to zero, and (3) fine-tuning on out-of-domain data using only RL loss.

2.2 Reinforcement Learning for Fine-tuning

Reinforcement learning (RL) has emerged as a powerful paradigm for fine-tuning models, particularly in scenarios where explicit gradients for feedback are unavailable. For example, early applications included tasks such as machine translation [Ranzato *et al.*, 2016; Bahdanau *et al.*, 2017] and summarization [Wu and Hu, 2018; Gao *et al.*, 2019], where non-differentiable evaluation metrics were used as rewards to guide model optimization. In recent years, reinforcement learning from human feedback (RLHF) has gained widespread adoption across diverse domains, including large language models [Ziegler *et al.*, 2019; Ouyang *et al.*, 2022; Bai *et al.*, 2022], image generation [Lee *et al.*, 2023; Fan *et al.*, 2023; Black *et al.*, 2024], and music generation [Cideron *et al.*, 2024]. These studies have demonstrated that RLHF successfully aligns model outputs with human preferences, enhancing the relevance and quality of generated content. Building upon insights from these prior studies, we adopt RL to tackle the challenges posed by the non-differentiability of synthesizers, aiming to optimize performance in cross-domain sound matching.

3 Method

3.1 Overview

The problem of synthesizer sound matching can be framed as predicting the optimal synthesis parameters $\mathbf{x} = \{x_i\}_{i=1}^{n}$, so that when passed to a synthesizer S, the generated audio signal $\hat{y} = S(\hat{\mathbf{x}})$ closely matches the target sound y. The synthesis parameters $\mathbf{x}$ include n attributes that define the behavior of the synthesizer, such as the waveform type of oscillators, amplitude envelope, filter configurations, and modulation depth. We employ a neural network model π_θ, which takes the target sound as input and estimates the corresponding synthesis parameters, $\hat{\mathbf{x}} = \pi_\theta(y)$.

Our model training process consists of three stages, as illustrated in Figure 1. First, we train the model by minimizing the parameter loss using only in-domain sounds. In the subsequent stage, we gradually introduce RL loss to further

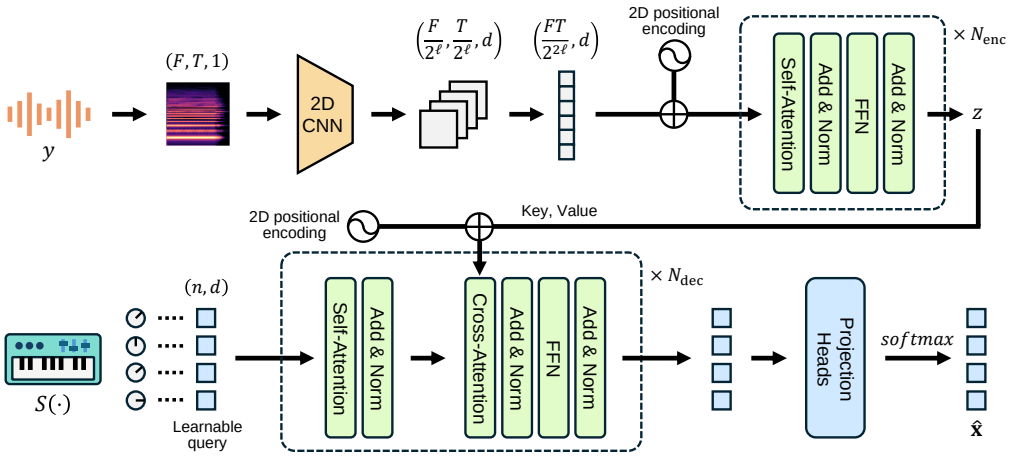


Figure 2: Overall model architecture of SynthRL. A feature map is extracted from the melspectrogram of the target sound using 2D CNN and transformer encoder. The transformer decoder models the relationships among synthesis parameters via self-attention on learnable queries and outputs the estimated parameters through cross-attention with the feature map and projection heads.

enhance performance on in-domain sounds. Finally, RL is applied exclusively to fine-tune the model on out-of-domain sounds, improving the cross-domain generalization of sound matching. Through the following sections, we provide a detailed breakdown of our proposed approach including the model architecture and the specifics of the learning process.

3.2 Model Architecture

Our proposed SynthRL model is based on a transformer encoder-decoder architecture (Figure 2). First, the target waveform y is converted into a single-channel melspectrogram of size $F \times T \times 1$. The encoder begins with a stack of ℓ 2D convolutional layers, each with a stride of 2, reducing the input melspectrogram from size $F \times T \times 1$ to $F/2^\ell \times T/2^\ell \times d$, where d is the output channel dimension. The output of the 2D CNN is flattened into a vector of size $(FT/2^{2\ell}) \times d$ and passed into the transformer encoder with 2D positional encoding added. The final output of the encoder is a feature map $z \in \mathbb{R}^{(FT/2^{2\ell}) \times d}$ of the target sound.

The decoder adopts a non-autoregressive transformer decoder architecture [Carion *et al.*, 2020]. It takes as input a set of n learnable query vectors in parallel, each of dimension d, where n corresponds to the number of synthesis parameters. Each query vector represents a different synthesis parameter. These queries first undergo self-attention to capture dependencies among the synthesis parameters, followed by cross-attention with the feature map z. In this step, the feature map serves as the key and value, allowing the decoder to incorporate relevant information from the target sound representation while attending to each query. The resulting decoder output vectors are then passed through n projection heads, which generate logits to estimate $\hat{x}_i$ for each synthesis parameter.

Each projection head is designed to handle a specific synthesis parameter. In most synthesizers, each synthesis parameter $x_i \in \mathbf{x}$ belongs to either a categorical or numerical type. For example, x_1 could denote the wave type of an oscillator, taking one of four categorical values: sine, triangle, square or sawtooth. In contrast, x_2 might correspond to the filter cutoff frequency, which is a continuous numerical value ranging between 0 and 1. To unify the estimation process, we discretize all numerical parameters into 25 equally spaced classes, converting their estimation into a classification problem. Consequently, each projection head maps the d-dimensional output of its corresponding decoder query to a vector with a dimensionality equal to the number of classes for the associated parameter.

3.3 Parameter Loss

Building upon the classification framework for synthesis parameters introduced earlier, we define the parameter loss as a cross-entropy loss between the ground truth parameters $\mathbf{x}$ and their corresponding estimates $\hat{\mathbf{x}}$. For discretized numerical parameters, however, a direct application of cross-entropy can overlook the inherent structure of these parameters, where errors involving adjacent classes are less significant than those involving distant ones.

To address this issue, we smooth the ground truth label distribution by redistributing a portion of the probability mass from the true class to its neighboring classes. Specifically, the ground truth label is convolved with a Gaussian kernel and the smoothed distribution is normalized to maintain the total probability, following the prior work [Chen *et al.*, 2022]. This approach ensures that predictions closer to the ground truth are penalized less severely, which better aligns with the numerical relationships between the classes.

3.4 Reinforcement Learning Procedure

In this section, we provide a detailed description of the RL procedure applied to synthesizer sound matching. First, we present the RL formulation, followed by the design of the reward function. Finally, we elaborate on the use of PER with our modification to enhance the learning efficiency.

RL Formulation

A typical RL formulation considers problems as Markov Decision Process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, R, P)$, where $\mathcal{S}$ denotes a state space, $\mathcal{A}$ is an action space, R is a reward function, and P is a transition probability. The objective of RL is to learn an optimal policy that maximizes the expected cumulative reward. We formulate the problem of synthesizer sound matching within an RL framework, defining the state as the target sound $y \in \mathcal{S}$, the action as the estimated synthesis parameter $\hat{\mathbf{x}} \in \mathcal{A}$, the reward as the similarity between the target and generated sound. The policy corresponds to the SynthRL model π_θ, which takes y as input and outputs $\hat{\mathbf{x}}$.

In contrast to the majority of RL tasks that revolve around multi-step decision processes, our problem formulation is centered on a single-step decision process which still fits within the MDP framework. This formulation can also be viewed as a contextual bandit problem [Lu *et al.*, 2010]. Therefore, the cumulative reward is equivalent to the single reward for the current state-action pair $(y, \hat{\mathbf{x}})$, and our RL objective function to maximize is given by $J(\theta) = \mathbb{E}_{\hat{\mathbf{x}}\sim\pi_\theta}[R(y, \hat{y})]$, where $\hat{y} = S(\hat{\mathbf{x}})$ is the generated sound by the synthesizer S. We employ REINFORCE [Williams, 1992], which is a simple yet effective algorithm particularly for single-step processes. REINFORCE updates the policy by gradient ascent, where the gradient of the objective function is computed with respect to the policy parameters. The gradient of the objective function is defined as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\hat{\mathbf{x}}\sim\pi_\theta}[R(y, \hat{y}) \nabla_\theta \log \pi_\theta(\hat{\mathbf{x}}|y)]. \quad (1)$$

From this equation, the reward $R(y, \hat{y})$ is a scalar value computed from the target and generated sounds, and the gradient is required only for the log-probability of the policy π_θ. Therefore, our RL objective enables policy optimization for a non-differentiable synthesizer S, utilizing a reward based on sound similarity.

Reward Function

To optimize the policy for estimating synthesis parameters that produce sounds $\hat{y} = S(\hat{\mathbf{x}})$ closely resembling the target sound, we design the reward function based on perceptual similarity metrics. We integrate the following three metrics which are commonly used in sound similarity measurement:

- **Spectrogram MAE** [Arık *et al.*, 2019]:

$$\text{Spec}(y, \hat{y}) = \| \log(\text{STFT}(y)) - \log(\text{STFT}(\hat{y})) \|_1, \quad (2)$$

- **Spectral convergence** [Arık *et al.*, 2019]:

$$\text{SC}(y, \hat{y}) = \frac{\|\text{STFT}(y) - \text{STFT}(\hat{y})\|_\text{F}}{\|\text{STFT}(y)\|_\text{F}}, \quad (3)$$

- **MFCC MAE** [Horner *et al.*, 2011]:

$$\text{MFCC}(y, \hat{y}) = \|\text{MFCC}(y) - \text{MFCC}(\hat{y})\|_1, \quad (4)$$

where $\text{STFT}(\cdot)$ is the short-time Fourier transform and $\text{MFCC}(\cdot)$ is the 13-band mel-frequency cepstral coefficients. These metrics all exhibit lower values as sound similarity increases. The reward function is defined as the reciprocal of a weighted sum of the metrics as follows:

$$R = [w_1 \cdot \text{Spec}(y, \hat{y}) + w_2 \cdot \text{SC}(y, \hat{y}) + w_3 \cdot \text{MFCC}(y, \hat{y})]^{-1}. \quad (5)$$

Algorithm 1 Reward-based PER

Require: Synthesizer S, policy π_θ, learning rate α, buffer capacity m, dataset $\mathcal{D}$
Initialize: Prioritized replay buffer $\mathcal{B}$

for epoch $= 1, N$ **do**
  for each iteration **do**
    Sample mini-batch of target sounds: $y_i \sim \mathcal{D}$
    // Prioritized replay buffer update
    Sample actions from policy: $\hat{\mathbf{x}}_i \sim \pi_\theta(\cdot|y_i)$
    Generate sounds $\hat{y}_i = S(\hat{\mathbf{x}}_i)$
    Calculate rewards $R_i = R(y_i, \hat{y}_i)$ using Eq. 5
    if epoch $\leq m$ **then**
      Store experiences $(y_i, \hat{\mathbf{x}}_i, R_i)$ into $\mathcal{B}$
    else if $R_i > \min(R(y_i, \hat{y}) \in \mathcal{B})$ **then**
      Replace $\underset{(y_i, \hat{\mathbf{x}}, R)\in\mathcal{B}}{\text{argmin}} R(y_i, \hat{y})$ with $(y_i, \hat{\mathbf{x}}_i, R_i)$
    end if
    // Policy training
    Sample experiences: $(y, \hat{\mathbf{x}}, R(y, \hat{y})) \sim \mathcal{B}$
    Update π_θ using Eq. 6:
    $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$
  end for
end for

The weights w_1, w_2, and w_3 are hyperparameters, and we empirically set their default values to $w_1 = 0.27$, $w_2 = 0.7$, and $w_3 = 0.03$ considering the scale of each metric. In the experimental section, we will provide a detailed analysis of the reward function including alignment between the computed reward and subjective evaluations of similarity, as well as limitations of using a single metric as the reward.

Reward-based PER

As mentioned earlier, the REINFORCE algorithm updates the policy π_θ using rewards $R(y, \hat{y})$ obtained from stochastically sampled actions $\hat{\mathbf{x}} \sim \pi_\theta(y)$ by gradient ascent using Equation 1. This learning process enables the policy to iteratively improve by assigning higher probabilities to actions that yield greater rewards. However, we found that directly applying this approach to synthesizer sound matching did not lead to effective learning from our preliminary experiments.

From the perspective of RL, the synthesizer sound matching task involves a complex action space comprising hundreds of synthesis parameters. Moreover, even minor changes to certain parameters can drastically alter the resulting sound, particularly for categorical parameters where discrete changes can result in significant timbre shifts. Consequently, stochastically sampling actions from the policy can give rise to sparse occurrences of high-reward actions, impairing the learning efficiency.

To mitigate this challenge, we leverage PER, a technique originally proposed to enhance learning efficiency in off-policy RL algorithms by prioritizing experiences that would most likely lead to an improvement in the policy or value function. In our approach, we selectively reuse experiences with high rewards which are directly associated with the learning objective. Concretely, for each target sound y, we

Dataset	Method	Spectrogram MAE	Spectral Convergence	MFCC MAE (13-band)	MFCC MAE (40-band)
in-domain	PresetGenVAE	0.675 ± 0.007	1.014 ± 0.011	22.65 ± 0.23	15.03 ± 0.20
	Sound2Synth	0.496 ± 0.002	0.666 ± 0.003	15.40 ± 0.01	9.52 ± 0.01
	SynthRL-p (Ours)	0.477 ± 0.002	0.654 ± 0.005	14.59 ± 0.09	9.01 ± 0.08
	SynthRL-i (Ours)	$\mathbf{0.461 \pm 0.001}$	$\mathbf{0.607 \pm 0.004}$	$\mathbf{13.61 \pm 0.02}$	$\mathbf{8.46 \pm 0.04}$
out-of-domain	PresetGenVAE	1.009 ± 0.013	1.199 ± 0.046	27.55 ± 0.58	15.74 ± 0.52
	Sound2Synth	1.355 ± 0.021	1.752 ± 0.028	35.35 ± 0.85	19.52 ± 0.18
	SynthRL-i (Ours)	1.001 ± 0.032	1.356 ± 0.016	26.87 ± 0.46	14.96 ± 0.27
	SynthRL-o (Ours)	$\mathbf{0.845 \pm 0.013}$	$\mathbf{1.028 \pm 0.018}$	$\mathbf{22.40 \pm 0.39}$	$\mathbf{12.67 \pm 0.10}$

Table 1: Quantitative evaluation results. We report the mean and standard deviation of the evaluation results on the test set for models trained with three random seeds.

maintain a prioritized replay buffer that stores m actions and their corresponding rewards. When the buffer is full, a new action $\hat{\mathbf{x}}$ sampled from the policy $\pi_\theta(y)$ replaces the least-rewarded experience in the buffer if its reward exceeds the current minimum. These experiences stored in the prioritized replay buffer are uniformly sampled to update the policy. In this case, a discrepancy arises between the distribution $\mu(\hat{\mathbf{x}}|y)$ used to sample actions induced by the prioritized replay buffer and the policy $\pi_\theta(\hat{\mathbf{x}}|y)$ being optimized, since REINFORCE is inherently an on-policy algorithm which precludes the reuse of old experiences. This mismatch is corrected by importance sampling, where the gradient is reweighted based on the ratio of probabilities under these two different distributions [Jie and Abbeel, 2010]. As a result, the gradient of the objective becomes as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\hat{\mathbf{x}}\sim\mu}[\frac{\pi_\theta(\hat{\mathbf{x}}|y)}{\mu(\hat{\mathbf{x}}|y)} R(y,\hat{y}) \nabla_\theta \log \pi_\theta(\hat{\mathbf{x}}|y)]. \quad (6)$$

For practical implementation, we approximate the sampling distribution as uniform over the buffer, $\mu(\hat{\mathbf{x}}|y) = 1/m$. The reinforcement learning process using reward-based PER is summarized in Algorithm 1.

4 Experiment

4.1 Dataset

While our proposed method is applicable to various types of synthesizers, we conducted our experiments using the Dexed[1] synthesizer, which has been widely adopted in previous synthesizer sound matching research. Dexed is an open-source synthesizer modeled on the Yamaha DX7 which is one of the most iconic FM synthesizers. It includes 144 synthesis parameters that collectively shape the timbre, comprising 84 numerical and 60 categorical types. We use target sound datasets consisting of single MIDI notes played at pitch 60 and velocity 85. Each note has a duration of 4 seconds, with 3 seconds of sustain followed by 1 second of release, providing a well-defined envelope for the synthesized sounds.

For the in-domain dataset, we use sounds generated from approximately 30,000 publicly available Dexed sound presets collected in prior research [Vaillant *et al.*, 2021]. This dataset excludes redundant or silent sounds, processed by previous researchers. For the out-of-domain dataset, we use sounds generated from a different type of open-source synthesizer Surge XT[2] that employs a subtractive hybrid synthesis method. Due to the fundamental difference in synthesis methods, Surge XT produces a distinct set of sounds compared to those of Dexed. We collected approximately 3,500 non-redundant sounds from publicly available Surge presets. These datasets were divided into training, validation, and test sets using a ratio of 64%, 16%, and 20%, respectively.

4.2 Training Details

Through the following experiments, we compare SynthRL with two baseline methods and investigate the effectiveness of the proposed components in our method. The first baseline is Sound2Synth, a SOTA method demonstrated on the Dexed synthesizer for sound matching. The second is PresetGenVAE, which exhibits relatively smaller performance degradation on the out-of-domain dataset.

SynthRL is first trained on the in-domain dataset for 400 epochs. Over the initial 200 epochs, only the parameter loss described in Section 3.3 is applied. During the next 100 epochs, both the RL loss from Equation 6 and the parameter loss are used, with the ratio of the RL loss linearly increasing from 0 to 1. The final 100 epochs of in-domain training are performed using only the RL loss. After the in-domain training is completed, the model is fine-tuned on the out-of-domain dataset for an additional 400 epochs using only the RL loss.

We assign distinct names to the trained models based on the three training stages, for the ablation study on the RL training which is a key component of our proposed method as follows:

- **SynthRL-p** is the model trained for initial 200 epochs using only the parameter loss (stage 1).
- **SynthRL-i** is further trained model on the in-domain dataset with the addition of RL loss from 200 to 400 epochs (stage 2).
- **SynthRL-o** is the fine-tuned model on the out-of-domain dataset (stage 3).

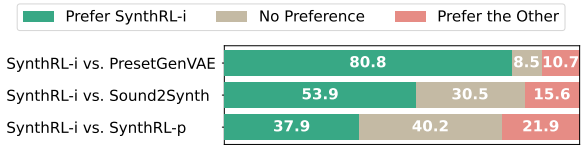
The baseline models are trained on the in-domain dataset for 400 epochs using the official source code released by the

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

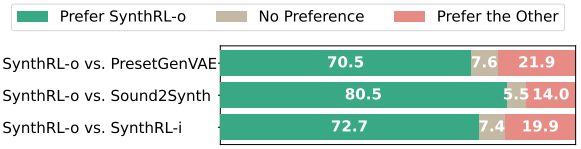
[2] https://github.com/surge-synthesizer/surge

Dataset	Method	MOS
in-domain	PresetGenVAE	3.17 ± 0.32
	Sound2Synth	3.91 ± 0.20
	SynthRL-p (Ours)	4.16 ± 0.17
	SynthRL-i (Ours)	$\mathbf{4.29 \pm 0.14}$
out-of-domain	PresetGenVAE	2.09 ± 0.24
	Sound2Synth	1.94 ± 0.19
	SynthRL-i (Ours)	2.28 ± 0.28
	SynthRL-o (Ours)	$\mathbf{3.01 \pm 0.26}$

Table 2: MOS test results with the average score and 95% CI.



(a) In-domain



(b) Out-of-domain

Figure 3: ABX test results with the average preference rate reported as a percentage.

authors. These in-domain-only trained models are also used for evaluation on the out-of-domain dataset, since they are inherently incapable of out-of-domain training, which provided significant motivation for our research as mentioned earlier. Our source code and audio samples are available on the accompanying webpage[3].

5 Result and Analysis

5.1 Quantitative Evaluation

Following previous studies [Vaillant *et al.*, 2021; Barkan *et al.*, 2023], we evaluate sound similarity quantitatively using spectrogram MAE, spectral convergence, and MFCC MAE. Table 1 presents the results for all evaluation metrics on both the in-domain and out-of-domain datasets. Overall, our method outperforms the baselines across all metrics on both domains.

For the in-domain results, SynthRL-p outperforms the stronger baseline Sound2Synth by an average 4.1% across all metrics. The performance gain of SynthRL-p over the baselines is attributed to the model architecture of SynthRL, as it is trained with only the parameter loss similar to other methods. It indicates that our proposed transformer encoder-decoder based architecture is effective for synthesizer sound matching. A comparison of SynthRL-p and SynthRL-i demonstrates that the addition of RL leads to further performance improvement on average 5.8%. Our RL approach successfully compensates for limitations of the parameter loss, such as the difficulty in learning subtle parameter differences that result in noticeable timbre changes.

On the out-of-domain dataset, the baselines and SynthRL-i all exhibit noticeable performance drops compared to the in-domain case. This degradation inherently arises from the limited in-domain sound distribution of the synthesizer. Owing to the cross-domain applicability of our RL method, SynthRL-o shows a substantial improvement over SynthRL-i by an average 17.9% across all metrics. Compared to the baselines, SynthRL-o outperforms PresetGenVAE by an average 17.2% and Sound2Synth by 37.7%.

5.2 Subjective Evaluation

We conducted Mean Opinion Score (MOS) and ABX similarity tests through Amazon Mechanical Turk (MTurk). In the MOS test, 64 evaluators rated the similarity of the samples to the ground truth on a scale from 1 (completely different) to 5 (identical). In the ABX similarity test, 96 evaluators listened to two samples generated by different models and chose the sample more similar to the ground truth. For each test, 64 randomly selected target sounds were evaluated and all evaluators were instructed to wear headphones in a quiet environment. We intentionally included pairs of completely different and identical sounds in the evaluation samples, and responses from evaluators who failed to distinguish them were considered unreliable and excluded. The results of the MOS and ABX tests are presented in Table 2 and Figure 3, respectively.

For the in-domain dataset, SynthRL-i achieves the highest MOS of 4.29. However, considering the 95% confidence interval (CI), the differences from SynthRL-p (4.16) and Sound2Synth (3.91) are not statistically significant. This result indicates that all three models exhibit strong in-domain performance, rather than highlighting the numerical differences since it is difficult for humans to distinguish subtle differences using 5-point scale. The pairwise performance differences between individual models are more clearly demonstrated in the ABX test results. In this test, SynthRL-i is preferred over all other models, with a minimum of 37.9% win and 21.9% loss compared to SynthRL-p.

In the out-of-domain MOS test, SynthRL-o scores 3.01 with a 95% CI of 0.26, considerably outperforming SynthRL-i (2.28) and the other baselines. This remarkable improvement is further validated by the ABX test, where SynthRL-o achieves significantly higher preference rates compared to all other models, with a minimum of 70.5% over PresetGenVAE. Overall, these results demonstrate that our RL method leads to consistent improvements across both domains from a human perceptual perspective, as well as in quantitative performance.

5.3 Analysis of Reward Design

Our reward design is a combination of three sound similarity metrics, motivated by the experimentally observed limitations of using a single metric. For instance, spectrogram MAE is a widely used metric for sound loss functions, along with various variants such as multi-scale approaches [Engel

[3]https://argaaw.github.io/synthrl-demo/

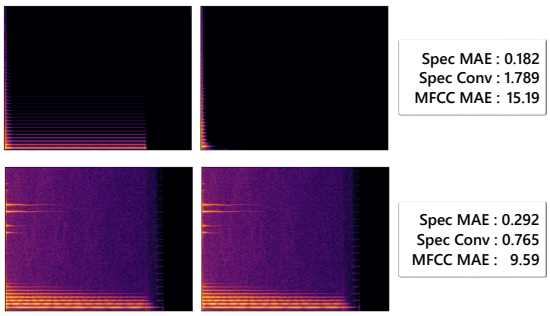


Figure 4: Examples of target (left) and estimated (right) sound pairs along with calculated metric values. The top row shows an example of drawback from SynthRL trained with only spectrogram MAE as the reward, while the bottom row presents an example of nearly indistinguishable sounds for reference.

Figure 5: Scatter plot of MOS obtained from the listening test and calculated rewards. The error bars represent the 95% CI.

et al., 2020]. However, using a reward solely based on spectrogram MAE in RL can lead to undesired learning outcomes, as illustrated in Figure 4. The top row depicts spectrograms of a target sound with few harmonics (left) and the corresponding output of a SynthRL model trained with the single spectrogram MAE-based reward (right). In this case, the two sounds differ significantly as the target sound is sustained for a long duration, whereas the estimated one is transient. Nevertheless, the spectrogram MAE between the two sounds is 0.182, which is notably low compared to 0.292 between spectrograms that are barely distinguishable as shown in the bottom row. This low value stems from the widely distributed low energy regions represented in black on the spectrograms. Consequently, using only spectrogram MAE as the reward leads the model to consistently output transient sounds for diverse target sounds with overall low energy. In contrast, the other two metrics exhibit higher values compared to the bottom row, particularly spectral convergence with a significant high value of 1.789, effectively reflecting the difference of the two sounds. The complementary relationship among these reward components plays a crucial role in successful training of SynthRL across diverse sounds.

We further investigate the alignment between our designed reward function and human perception of sound similarity. A total of 120 evaluators recruited via MTurk rated the similarity of randomly sampled pairs of sounds on a 5-point MOS scale, with each pair being rated by 30 evaluators. Figure 5 shows the scatter plot of the rewards calculated from Equation 5 and the MOS for the sample pairs. It demonstrates a clear positive relationship between the rewards and MOS, with a Pearson correlation of 0.85 ($p < 0.001$). While the deviations are relatively high in the mid-range MOS (2-4), considering the subjective nature of sound similarity judgments when the sounds are neither completely different nor identical, the results suggest that our reward function is well-designed to align with the intended goal.

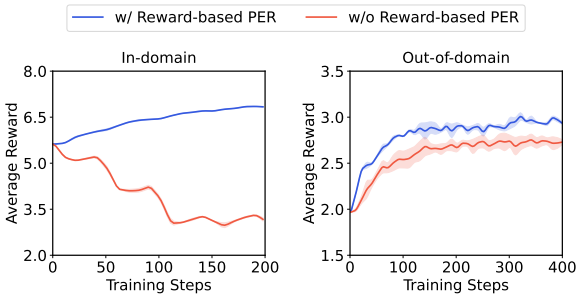


Figure 6: Ablation test results on reward-based PER. All results are averaged over three random seeds and the shaded region represents the standard deviation.

5.4 Effect of Reward-based PER

We conduct an ablation study to analyze the effectiveness of reward-based PER. Figure 6 illustrates average reward curves on validation sets over training. Step 0 in the figure represents the point at which RL training begins for each domain, corresponding to fully trained SynthRL-p for in-domain and fully trained SynthRL-i for out-of-domain. The result demonstrates that training with reward-based PER is more effective in both cases. Interestingly, the performance on in-domain decreases during training when reward-based PER is not used. We conjecture that the degradation occurs because the model already achieves sufficiently high performance through parameter loss training before the introduction of RL loss. In other words, applying RL without prioritized sampling in the complex state and action space of the synthesizer sound matching task can lead to inferior performance compared to training with only parameter loss.

6 Conclusion

We introduce SynthRL, the first application of RL in synthesizer sound matching, addressing the limitation of reliance on parameter loss and the challenges of cross-domain generalization. SynthRL effectively incorporates perceptual loss through RL, enabling fine-tuning on out-of-domain sounds with non-differentiable synthesizers, as well as further improving in-domain performance. Our extensive experiments shed light on the essential factors for successfully applying RL to synthesizer sound matching, including the effective reward design and the importance of prioritized sampling.

References

[Arık *et al.*, 2019] Sercan Ö. Arık, Heewoo Jun, and Gregory Diamos. Fast spectrogram inversion using multi-head convolutional neural networks. *IEEE Signal Processing Letters*, 26(1):94–98, 2019.

[Bahdanau *et al.*, 2017] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations*, 2017.

[Bai *et al.*, 2022] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[Barkan *et al.*, 2019] Oren Barkan, David Tsiris, Ori Katz, and Noam Koenigstein. Inversynth: Deep estimation of synthesizer parameter configurations from audio signals. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12):2385–2396, 2019.

[Barkan *et al.*, 2023] Oren Barkan, Shlomi Shvartzman, Noy Uzrad, Moshe Laufer, Almog Elharar, and Noam Koenigstein. Inversynth II: Sound matching via self-supervised synthesizer-proxy and inference-time finetuning. In *ISMIR*, pages 642–648, 2023.

[Black *et al.*, 2024] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations*, 2024.

[Carion *et al.*, 2020] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[Caspe *et al.*, 2022] Franco Caspe, Andrew McPherson, and Mark Sandler. DDX7: Differentiable fm synthesis of musical instrument sounds. In *ISMIR*, 2022.

[Chen *et al.*, 2022] Zui Chen, Yansen Jing, Shengcheng Yuan, Yifei Xu, Jian Wu, and Hang Zhao. Sound2Synth: Interpreting sound via fm synthesizer parameters estimation. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*, 2022.

[Cideron *et al.*, 2024] Geoffrey Cideron, Sertan Girgin, Mauro Verzetti, Damien Vincent, Matej Kastelic, Zalán Borsos, Brian McWilliams, Victor Ungureanu, Olivier Bachem, Olivier Pietquin, et al. MusicRL: Aligning music generation to human preferences. In *Forty-first International Conference on Machine Learning*, 2024.

[Engel *et al.*, 2020] Jesse Engel, Lamtharn (Hanoi) Hantrakul, Chenjie Gu, and Adam Roberts. DDSP: Differentiable digital signal processing. In *International Conference on Learning Representations*, 2020.

[Esling *et al.*, 2019] Philippe Esling, Naotake Masuda, Adrien Bardet, Romeo Despres, and Axel Chemla-Romeu-Santos. Flow synthesizer: Universal audio synthesizer control with normalizing flows. In *22th International Conference on Digital Audio Effects (DAFx)*. IEEE, 2019.

[Fan *et al.*, 2023] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. DPOK: Reinforcement learning for fine-tuning text-to-image diffusion models. In *Advances in Neural Information Processing Systems*, pages 79858–79885, 2023.

[Gao *et al.*, 2019] Yang Gao, Christian Meyer, Mohsen Mesgar, and Iryna Gurevych. Reward learning for efficient reinforcement learning in extractive document summarisation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019.

[Garcia, 2002] Ricardo A. Garcia. Automatic design of sound synthesis techniques by means of genetic programming. In *Audio Engineering Society Convention 113*. Audio Engineering Society, 2002.

[Guo *et al.*, 2023] Yanjiang Guo, Jingyue Gao, Zheng Wu, Chengming Shi, and Jianyu Chen. Reinforcement learning with demonstrations from mismatched task under sparse reward. In *Conference on Robot Learning*, pages 1146–1156. PMLR, 2023.

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

[Horner *et al.*, 2011] Andrew B. Horner, James W. Beauchamp, and Richard H. Y. So. Evaluation of mel-band and mfcc-based error metrics for correspondence to discrimination of spectrally altered musical instrument sounds. *Journal of the Audio Engineering Society*, 59(5):290–303, 2011.

[Jaques *et al.*, 2017] Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E. Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with KL-control. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 1645–1654. PMLR, 2017.

[Jie and Abbeel, 2010] Tang Jie and Pieter Abbeel. On a connection between importance sampling and the likelihood ratio policy gradient. In *Advances in Neural Information Processing Systems*, volume 23, 2010.

[Lee *et al.*, 2023] Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang S. Gu. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*, 2023.

[Lu *et al.*, 2010] Tyler Lu, David Pal, and Martin Pal. Contextual multi-armed bandits. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence*

and Statistics, volume 9 of *Proceedings of Machine Learning Research*, pages 485–492. PMLR, May 2010.

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

[Mitchell and Sullivan, 2005] Thomas Mitchell and Charlie Sullivan. Frequency modulation tone matching using a fuzzy clustering evolution strategy. In *Audio Engineering Society Convention 118*. Audio Engineering Society, 2005.

[Nair *et al.*, 2018] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299, 2018.

[Ouyang *et al.*, 2022] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, pages 27730–27744, 2022.

[Powell, 1987] Steven Powell. The ABCs of synthesizers. *Music Educators Journal*, 74(4):27–31, 1987.

[Ranzato *et al.*, 2016] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*, 2016.

[Renault *et al.*, 2022] Lenny Renault, Rémi Mignot, and Axel Roebel. Differentiable piano model for MIDI-to-audio performance synthesis. In *25th International Conference on Digital Audio Effects (DAFx)*, 2022.

[Roth and Yee-King, 2011] Martin Roth and Matthew Yee-King. A comparison of parametric optimization techniques for musical instrument tone matching. In *Audio Engineering Society Convention 130*. Audio Engineering Society, 2011.

[Schaul, 2015] Tom Schaul. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[Sebastian *et al.*, 2009] Heise Sebastian, Hlatky Michael, and Loviscach Jörn. Automatic cloning of recorded sounds by software synthesizers. In *Audio Engineering Society Convention 127*. Audio Engineering Society, 2009.

[Shan *et al.*, 2022] Siyuan Shan, Lamtharn Hantrakul, Jitong Chen, Matt Avent, and David Trevelyan. Differentiable wavetable synthesis. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4598–4602. IEEE, 2022.

[Tatar *et al.*, 2016] Kıvanç Tatar, Matthieu Macret, and Philippe Pasquier. Automatic synthesizer preset generation with PresetGen. *Journal of New Music Research*, 45(2):124–144, 2016.

[Uzrad *et al.*, 2024] Noy Uzrad, Oren Barkan, Almog Elharar, Shlomi Shvartzman, Moshe Laufer, Lior Wolf, and Noam Koenigstein. DiffMoog: a differentiable modular synthesizer for sound matching. *arXiv preprint arXiv:2401.12570*, 2024.

[Vaillant *et al.*, 2021] Gwendal L. Vaillant, Thierry Dutoit, and Sébastien Dekeyser. Improving synthesizer programming from variational autoencoders latent space. In *24th International Conference on Digital Audio Effects (DAFx)*, pages 276–283. IEEE, 2021.

[Wiggins and Kim, 2023] Andrew Wiggins and Youngmoo Kim. A differentiable acoustic guitar model for string-specific polyphonic synthesis. In *2023 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 1–5, 2023.

[Williams, 1992] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

[Wu and Hu, 2018] Yuxiang Wu and Baotian Hu. Learning to extract coherent summary via deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, April 2018.

[Yee-King and Roth, 2008] Matthew Yee-King and Martin Roth. Synthbot: An unsupervised software synthesizer programmer. In *Proceedings of the 2008 International Computer Music Conference*. Michigan Publishing, 2008.

[Yee-King *et al.*, 2018] Matthew Yee-King, Leon Fedden, and Mark d'Inverno. Automatic programming of vst sound synthesizers using deep networks and other techniques. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(2):150–159, 2018.

[Ziegler *et al.*, 2019] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

Supplementary Material

A Implementation Details

A.1 Network Architecture

For all experiments, the input audio is sampled at 22,050 Hz and subsequently transformed into a melspectrogram with a frame size of 1024, hop size of 256, and 128 mel bins. Consequently, the transformed melspectrogram has a size of $(F, T) = (128, 345)$ for a 4-second audio input.

The 2D CNN consists of $\ell = 5$ layers with kernel sizes of $(5, 4, 4, 4, 4)$. The output channel dimension of the 2D CNN is set to $d = 512$, which is consistent with the hidden dimensionality of the transformer encoder and decoder. Following the CNN, the transformer encoder and decoder comprise 6 blocks each. The projection heads are implemented as $n = 144$ independent single-layer linear classifiers, where n corresponds to the number of synthesis parameters. Each projection head receives a 512-dimensional vector from the transformer decoder and outputs logits corresponding to the number of discrete classes for the respective synthesis parameter.

A.2 Hyperparameters

The training procedure follows the hyperparameter configurations detailed in Table 3.

Hyperparameter	Value
Batch size	32
Optimizer	AdamW
Optimizer momentum	$\beta_1 = 0.9, \beta_2 = 0.999$
Weight decay	0.0001
Dropout	0.3
Initial learning rate	0.0002
Learning rate schedule	Cosine annealing
Cosine annealing cycle	50 epochs
Warm-up epochs	10

Table 3: Hyperparameters used for the experiments.

For reinforcement learning, the reward is clipped to a maximum value of 10 to prevent instability due to excessively high reward signals. Additionally, the capacity of the prioritized replay buffer for reward-based PER is set to $m = 5$.

B Effect of Three-stage Training Procedure

In SynthRL, we train the model initially using only parameter loss and gradually introduce RL loss, instead of applying RL loss alone from scratch. We conduct ablation study on the three-stage training process described in Figure 1 and Section 4.2. Table 4 shows the evaluation results on the out-of-domain dataset obtained in the absence of specific training stages.

Training stage 1-2-3 corresponds to the results of the complete SynthRL training method, i.e. SynthRL-o, which exhibits the best performance. The performance is significantly worse when RL loss is exclusively applied from scratch (stage 3), with an average degradation of 25.4% across all metrics compared to the results of stage 1-2-3. For training stage 2-3, where both parameter loss and RL loss are used without the initial stage using only parameter loss, the performance is slightly lower than stage 1-2-3, showing an average degradation of 3.93%. These results suggest that the initial training with parameter loss plays a pivotal role in the effective application of RL. Given the complex state-action space of the synthesizer sound matching task, directly exploring the space and discovering high-reward actions from scratch is inefficient. Early training using parameter loss effectively guides the exploration process, which can be interpreted as widely studied RL approaches that leverage expert demonstrations to assist exploration and enhance learning efficiency [Nair *et al.*, 2018; Guo *et al.*, 2023].

Training Stage	Spectrogram MAE	Spectral Convergence	MFCC MAE (13-band)	MFCC MAE (40-band)
3	1.045	1.154	30.00	16.69
1-3	0.881	1.076	23.14	13.43
2-3	0.876	1.087	23.02	13.12
1-2-3	**0.845**	**1.028**	**22.40**	**12.67**

Table 4: Ablation test results on training stages.

In the case of training with parameter loss alone on in-domain, followed by fine-tuning with RL loss on out-of-domain (stage 1-3), the performance shows a 4.6% average decreases compared to the results of stage 1-2-3. This indicates that the gradual introduction of RL loss (stage 2) aids in improving training efficiency. Previous studies also have reported that penalizing the abrupt shift in learning signals and large deviations from the original pretrained model is crucial when fine-tuning model with RL [Jaques *et al.*, 2017; Fan *et al.*, 2023], although our approach does not employ Kullback-Leibler (KL) regularization which is used in these studies for more conservative RL application.