Autonomous Listening-Based Synthesizer Control with Reinforcement Learning for Live Sound Matching

Vincenzo Madaghiele
Department of Musicology
University of Oslo
Oslo, Norway
vincenzo.madaghiele@imv.uio.no

Stefano Fasciani
Department of Musicology
University of Oslo
Oslo, Norway
stefano.fasciani@imv.uio.no

Tejaswinee Kelkar
Department of Musicology
University of Oslo
Oslo, Norway
tejaswinee.kelkar@imv.uio.no

Çağrı Erdem
Department of Informatics
University of Oslo
Oslo, Norway
cagrie@ifi.uio.no

Abstract

In this paper, we build on a previously proposed method for training an artificial agent that performs live control of sound synthesizers, using reinforcement learning and machine-listening to match the sound of a live musician. The agent listens to an incoming live audio stream and attempts to match its sound according to specific audio descriptors by controlling the parameters of an arbitrary synthesizer. We introduce a reinforcement-learning model with a continuous action space that integrates sound synthesis during training and live performance. We propose five complementary reward functions for matching target sounds, and evaluate their effectiveness in three scenarios: in-domain matching of synthesized target sounds, out-of-domain matching of synthesized targets, and out-of-domain matching using a corpus of musician-recorded audio. Finally, we compare the proposed reward functions across these scenarios, we discuss a live implementation and we present the results of quantitative experiments evaluating the matching task.

Keywords

Synthesizer control, Reinforcement learning, Sound matching, Musical agent

1 Introduction

Sound synthesis has revolutionized music-making by introducing tools capable of generating an immense variety of sounds. This sonic diversity often corresponds to a large number of adjustable synthesis parameters [10] that user can tune, whose relationship to the resulting sound are frequently complex and nonlinear. Controlling such a large set—or a selected subset—of parameters in a live setting to achieve dynamic sonic diversity and modulation can be overwhelming [21]; several techniques have been developed to facilitate this task. These techniques include few-to-many mappings that allow control of a sizeable number of synthesis parameters from lower-dimensional spaces [44]; automation of certain parameters using continuous or discrete modulation signals [5]; user-defined curves (e.g., sequencers); and reactive modulations generated from the analysis of audio inputs [45]. These live controls can affect music at different time scales within a performance. For example, a typical sound design technique uses the same modulation signal to adjust oscillator frequency and filter's cutoff, automating changes at the *micro* time scale, while a music sequencer affects the *meso* scale of musical structure by triggering events, and the sequencer's meta-parameters can operate at the *macro* scale of the performance.

In this paper, we present a technique that automates the control of an arbitrary number of synthesis parameters during performance, driven by live input audio, and learn control policies from a user-specific corpus using reinforcement learning (RL). This work expands on our previous proposal [23], in which we introduced a RL environment for training an autonomous agent for live timbre-following on in-domain sounds-that is, the agent learned to match sounds produced by the same synthesizer it controls. In this work, we reformulate the RL environment to incorporate continuous action spaces (from previous discrete steps) and novel rewards. We also extend the previous setup by synthesizing sound during training to mimic a live interaction. We evaluate the proposed environment on both in-domain and out-of-domain live sound-matching tasks, using synthesized sound as the target in the former and a corpus of audio files in the latter. In the corpus-driven, out-of-domain setting, the agent can only approximate the target timbre, an intentional limitation that imparts a characteristic sonic identity to the autonomous system. We quantitatively compare the performance of three alternative reward functions and their combinations across these tasks, highlighting how the agent's learned policies enable it to act with a degree of autonomy in real-time control, and discussing how the divergent mapping learned in out-of-domain policies could be used as responsive live agents. In non-musical fields, RL has demonstrated potential to learn complex strategies to achieve elaborate context-aware behaviors. While this article focuses mainly on sound matching, a clearly defined and mostly objectively measurable goal, we view this work as a first step towards using RL to develop semi-autonomous musical agents able to learn complex divergent mappings for applications in live co-improvised performances.

2 Background

Designing techniques to facilitate or automate the control of sound synthesis during live performance is a long-standing research topic. Divergent mapping methods enable performers to manage numerous synthesis parameters from a reduced set of control signals during performance [21, 46]. Another approach is

NIME '26, June 23–26, 2026, London, UK

indirect parameter control based on the analysis of live incoming audio [34, 45]. This approach has been mainly used to produce *reactive* remappings [4, 30] and timbre resynthesis [7], as well as to control concatenative synthesizers [2].

Remapping and resynthesis are related to the task of *sound matching*. In sound matching, given a target audio signal, the algorithm seeks the combination of synthesis parameters for a specific synthesizer that produces the closest possible match. Ideally, once a suitable set of parameters is found, the sound designer can further tweak them to personalize the result. Early approaches to sound matching employed genetic algorithms [20, 25, 42], while modern techniques use supervised machine learning with deep neural networks [3, 12, 22, 47], differentiable digital signal processing (DDSP) [8, 18, 24, 36], and reinforcement learning [37]. The goal of these techniques is to match a given target sound by repeatedly generating and evaluating candidates, iteratively adjusting the synthesis parameters in the process. Similarity between the target and the candidate sounds is typically assessed by comparing spectrograms, spectrogram-derived representations [6], or audio-descriptor similarities [36]. Recent methods also compute modulation curves to match sounds, achieving the target by dynamically varying synthesis parameters over time during sound generation, rather than producing audio with fixed parameters [29].

In the context of sound matching, we refer to *in-domain* matching when the target and candidate sounds are produced by the same sound source, making it theoretically possible to find the exact combination of synthesis parameters to reproduce the target sound. In contrast, *out-of-domain* sound matching involves target sounds produced by sources other than the controlled synthesizer. As a result, a perfect match is generally unattainable, and there is no ground-truth parameter configuration for the target. Instead, the system seeks the closest achievable approximation, with the attainable degree of similarity depending on the overlap between the target source and the capabilities of the controlled synthesizer. Depending on the application, a poor overlap can also be deliberately chosen and treated as desirable—for example, to encourage creative divergence, impart a distinctive timbral identity to an autonomous agent, or enable stylistic reinterpretations and hybridizations in performance.

While the progress in this field has been significant in recent years, most sound matching applications focus on offline use cases, with only a few techniques being developed for live applications so far, specifically focusing on timbre remapping of voice [13, 40] and snare drum [35, 36]. Live sound matching presents additional difficulties, because it requires retaining continuity in the parameter space and in the timbre space while operating a parameter change on the synthesizer. This problem has been solved in several different ways. Fasciani et al [14] propose to limit the options for parameter transitions within a dynamic neighborhood of the current parameter at each time step, Stowell and Plumbey propose using cross-association regression trees [41], while Shier et al. [36] propose a technique based on learning to match relative movements in timbre spaces.

3 Reinforcement learning framework

We propose a system that follows and matches an incoming audio signal intended to be produced by a live musician by continuously adjusting the parameters of an arbitrary synthesizer. The system uses reinforcement learning (RL), a machine learning technique in which an agent learns to satisfy a reward function by repeatedly acting within an environment. Specifically, we model the system as a dynamic RL environment [31], as the target-the incoming sound-is constantly changing. The interaction scenario in our model is portrayed in Figure 1.

3.1 Environment

We consider a scenario in which a live musician is improvising on an arbitrary musical instrument. The sound produced by the musician is denoted as the target track $x(t)$. An artificial agent controls a sound synthesizer $g(\cdot)$ with P continuous parameters $\boldsymbol{\theta}$, producing an output signal $y(t)$, where $\boldsymbol{\theta} \in [0,1]^P$. The agent issues a new decision at each control interval T_u. Each decision corresponds to a variation in the synthesis parameters, denoted $\Delta\boldsymbol{\theta}[m]$, taken according to a policy π^* learned through repeated interactions with the environment. An episode has a predetermined duration of M steps, with step index $m \in \{1, \dots, M\}$, and each step corresponds to a new decision conditioned on the audio observed in the interval T_u. The state $\mathbf{s}[m]$ is constructed as an encoding $\mathbf{D}(\cdot)$ of the target audio within the current and previous windows, and synthesizer audio withing the current window, together with the current synthesizer parameters $\boldsymbol{\theta}[m]$, as described below.

$$
\begin{aligned}
&W_m = \big[(m-1)T_u,\ mT_u\big) \\
&s_x[m](t) = x(t), \quad t \in W_m \\
&s_y[m](t) = y(t), \quad t \in W_m \\
&\mathbf{s}[m] = \big(\mathbf{D}(s_x[m]),\ \mathbf{D}(s_x[m-1]),\ \mathbf{D}(s_y[m-1]),\ \boldsymbol{\theta}[m]\big)
\end{aligned}
\tag{1}
$$

Here, W_m is the time window associated with step m, and $s_x[m]$ and $s_y[m]$ are the target and synthesizer signals restricted to W_m. The environment state at each time step m comprises the encodings of the current and previous target windows, the encoding of the previous synthesizer output window, and the current synthesizer parameters, where the current target window $\mathbf{D}(s_x[m])$ represents the matching objective. In our model, the encoding function $\mathbf{D}(\cdot)$ is a set of audio descriptors computed over each window and applied identically to both the target and synthesizer signals. The choice of descriptors depends on the characteristics of the target and synthesizer sounds and on specific aesthetic objectives.

During training, this configuration is implemented using prerecorded audio files to simulate $x(t)$. However, during live performance $s_x[m]$ is not known before generating $s_y[m]$. For this reason, we introduce a relatively small LSTM neural network, used only during inference, which predicts $\mathbf{D}(s_x[m])$ from $\{\mathbf{D}(s_x[m-K]) \dots, \mathbf{D}(s_x[m-1])$. This allows us to eliminate the latency during live performance, albeit at the cost of a slight prediction error that depends on the accuracy of the LSTM neural network. This aspect is discussed further in Section 3.5.

3.2 Agent

The agent learns a policy $\pi^*(\mathbf{s}[m])$ that determines actions maximizing a reward function through repeated interaction with the environment. An action corresponds to a parameter variation $\Delta\boldsymbol{\theta}[m]$ applied to the current synthesizer parameters $\boldsymbol{\theta}[m]$.

$$
\begin{aligned}
&\Delta\boldsymbol{\theta}[m] = \pi^*\big(\mathbf{s}[m]\big), \quad \Delta\boldsymbol{\theta}[m] \in [-1,1]^P \\
&\boldsymbol{\theta}[m] = \mathrm{clip}\big(\boldsymbol{\theta}[m-1] + \Delta\boldsymbol{\theta}[m],\ 0,\ 1\big), \quad \boldsymbol{\theta}[m] \in [0,1]^P
\end{aligned}
\tag{2}
$$

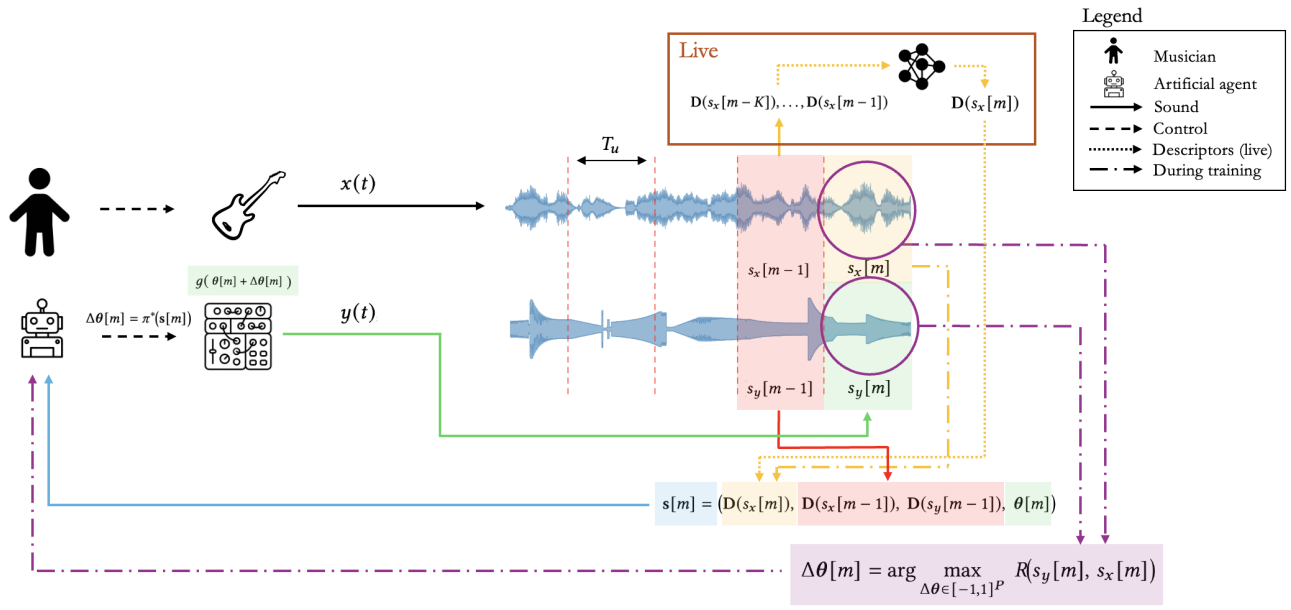

Figure 1: Diagram illustrating the data flow between the user and the artificial agent. The artificial agent modulates the parameters of a sound synthesizer to match the incoming sound $x(t)$ from a live performer with the synthesized sound $y(t)$ produced by the agent-controlled synthesizer. To avoid latency, during live performance the system employs a long-short term memory neural network to predict the descriptors $\mathbf{D}(s_x[m])$ based on the previous K windows. Predicted descriptors are used to compute real-time inference during performance.

The synthesizer sound segment in the interval W_m is generated with the updated parameters $\boldsymbol{\theta}[m]$, i.e., $s_y[m](t)$ is produced by $g(\boldsymbol{\theta}[m])$ for $t \in W_m$. The clipping operation enforces component-wise bounds within $[0, 1]$, ensuring $\boldsymbol{\theta}[m] \in [0, 1]^P$.

We employ a Soft Actor-Critic (SAC) agent [15, 16], an off-policy RL algorithm designed to balance exploration and exploitation in continuous control tasks. Our implementation uses a feed-forward actor neural network with two layers of 128 neurons each and two feed-forward critic networks of the same dimensions.

3.3 Reward

In a live sound-matching task, the agent's purpose is to update the synthesis parameters so that the synthesizer's output matches the latest incoming target sound produced by the musician as closely as possible. At step m, the agent receives a new target segment $s_x[m]$ and must produce a parameter change $\Delta\boldsymbol{\theta}[m]$ such that the next synthesizer segment $s_y[m]$ approximates $s_x[m]$. The agent selects $\Delta\boldsymbol{\theta}[m]$ to maximize a reward that measures similarity between the upcoming synthesizer segment and the current target segment:

$$\Delta\boldsymbol{\theta}[m] = \arg\max_{\Delta\boldsymbol{\theta}\in[-1,1]^P} R(s_y[m],\ s_x[m]) \tag{3}$$

We compare the effectiveness of three combinations of the following five reward functions for the sound-matching task. In this context, rewards are similarity metrics computed from the encodings $\mathbf{D}(\cdot)$ of the target and synthesizer segments.

3.3.1 Multi-resolution STFT. This reward measures the difference between the Short-Time Fourier Transform (STFT) of two signal segments computed at multiple resolutions. This is a popular loss function for sound matching and spectral reconstruction tasks [11, 26, 38].

$$R_{\text{mrSTFT}} = -||\text{mrSTFT}(s_y[m]) - \text{mrSTFT}(s_x[m])||_1 \tag{4}$$

We compute the log-magnitude of the STFT at three resolutions, using window sizes equal to hop size, half the hop size and a quarter of the hop size. The value of the hop size is context-dependent, the values used in our experiment are detailed in Section 4.

3.3.2 Spectral Convergence. This reward, based on multi-resolution STFT, was introduced in [1] and subsequently applied in several sound matching applications [12, 37]. It emphasizes large spectral components, which is particularly beneficial during the initial stages of training.

$$R_{SC} = -\frac{||\text{mrSTFT}(s_y[m]) - \text{mrSTFT}(s_x[m])||_{\text{F}}}{||\text{mrSTFT}(s_y[m])||_{\text{F}}} \tag{5}$$

In equation 5, $\text{mrSTFT}(\cdot)$ is computed as described in above in Section 3.3.1, and $||\cdot||_{\text{F}}$ is the Frobenius norm over time and frequency.

3.3.3 Absolute descriptor distance. This reward measures the difference between the selected audio descriptors $\mathbf{D}(\cdot)$ computed on the synthesized and target segments. This allows the optimization to focus on specific qualities of the sound, such as the pitch, or specific harmonic and timbral features. This reward is similar to the one we proposed in [23].

$$R_{ADD} = -||\mathbf{D}(s_y[m]) - \mathbf{D}(s_x[m])||_1 \tag{6}$$

3.3.4 Difference of descriptor distances. This reward function is inspired by the feature-difference loss proposed by Shier et al. [36]. We compute the difference between the value of descriptors values of synthesizer and target at time step m, and those at time step $m-1$. Rather than matching absolute descriptor values, this

reward aims to match the vector of descriptor changes from one window to the next in the target, consistent with finding in [27] showing that parallel distances in the respective timbre spaces are perceived analogously.

$$R_{DDD} = -||(\mathbf{D}(s_y[m]) - \mathbf{D}(s_y[m-1])) - (\mathbf{D}(s_x[m]) - \mathbf{D}(s_x[m-1]))||_1 \quad (7)$$

3.3.5 Parameters MAE. This reward aims to minimize the parameter variation at each step. It drives the model to prefer policies with minimal variation of parameters at each step.

$$R_{par} = -w \cdot ||\boldsymbol{\theta}[m] - \boldsymbol{\theta}[m-1]||_1 \quad (8)$$

This reward is weighted using a factor w, to prioritize other rewards based on sound-matching. Throughout this paper, we have conducted all experiments using $w = 0.1$.

We propose a RL model that aims to maximize the reward. Given that all rewards are based on distances, to be maximized the reward is the negative of the norm between the measured distances, for all the rewards proposed in this paper. Consequently, the maximum reward corresponds to a value of 0, which implies that the two compared distances are the same.

3.4 Pre-processing and training

The RL-based system can be trained either by generating target sounds with a synthesizer different from the one controlled by the agent or by using a corpus of target audio files For sound matching with a synthesizer as the target (either in- or out-of-domain), the target sound is generated during training using a arbitrary synthesizer, as long as it is coded as part of the training environment. In this case, the agent is trained to match the sound of the target synthesizer produced by a set of static parameters randomly selected at the beginning of each episode. In this way, the agent learns how each synthesis parameter affects the descriptor representation of the corresponding sound. After 10% of the total number of episodes, we introduce the option of fading the target synthesizer's parameters to new values during the episode, training the agent to match the sound of a moving target. Alternatively, when target sounds are instrumental recordings from a corpus, randomly selected segments of duration $M \cdot T_u$ from corpus audio files are used as target for each episode.

At each training step, it is necessary to normalize the values of the audio descriptors $\mathbf{D}(s_y[m])$ and $\mathbf{D}(s_x[m])$. We normalize descriptors by subtracting their mean and dividing by their standard deviation. The mean and standard deviation for the agent-controlled synthesizer are computed prior to training from recordings generated with evenly sampled parameter values. Target descriptors are normalized using the same mean and standard deviation computed from the agent-controlled synthesizer.

Before inference, we train a long short-term memory (LSTM) neural network on the same corpus used to simulate $x(t)$ during training of the RL model, with the task of predicting $\mathbf{D}(s_x[m])$ from the sequence $\{\mathbf{D}(s_x[m-K]), \ldots, \mathbf{D}(s_x[m-1])\}$ composed of the descriptor values computed over the latest K FFT windows. We estimate that inference of the RL model takes up to approximately 10 ms. During training of the LSTM, we remove the latest 10 ms of $s_x[m-1]$, allowing us to predict $\mathbf{D}(s_x[m])$ 10 ms in advance and thereby avoid cumulative latency. The LSTM is trained using the same corpus and the same descriptors $\mathbf{D}(\cdot)$ used to define the RL environment. We train the model using 80% of the corpus for training and 20% of it as test data. We keep 10% of the training data as a validation set. The LSTM model is composed of 4 hidden layers of 50 neurons each, and one linear layer having the same size of $\mathbf{D}(s_x[m-1])$, using a sequence length $K = 10$. The model is trained by minimizing mean-squared error (MSE), using the Adam optimizer with a learning rate of 10^{-4}.

3.5 Implementation

The RL environment is implemented entirely in Python, using the Gymnasium library [43]. During training, sound is synthesized using a non-real time sound graph using the SignalFlow Python library [9]. We use the Librosa library for Python [28] for audio analysis and computation of audio descriptors. The environment is modular, enabling the agent to control any synthesis patch coded in SignalFlow. The agent is implemented using the Stable Baselines 3 library for Python [33], providing integration with all continuous-action-space agents offered by the library. For the out-of-domain matching task with synthesizer as target, the target synthesizer is coded in signalflow as part of the gymnasium environment. For out-of-domain matching with an audio corpus as target, corpus audio files are loaded into the gymnasium environment and analyzed using the Librosa library.

During inference, the environment uses a real-time audio graph from SignalFlow. At initialization, the live system loads a trained agent and the corresponding synthesizer that the agent was trained to control. The system takes a live input signal as the target and continuously analyzes both the live input and the agent's synthesizer output, updating the synthesis parameters in windows of duration T_u. Then 10 ms before the end of each window, the descriptor representation of the input sound of the next window is predicted using the descriptor values extracted from the latest K windows and the pretrained LSTM model. The predicted descriptor values are then used by the agent, together with the descriptors of the current input window and those extracted from the synthesizer audio, to compute the next synthesis parameter modulation, which is applied at the end of the current window.

4 Experiments and results

In this section, we present and discuss experiments to quantitatively validate the proposed model in three separate scenarios: in-domain sound matching with synthesizer as target, out-of-domain sound matching with synthesizer as target and out-of-domain sound matching with a corpus of audio files as target. We evaluate each scenario using the following four synthesizers:

- **Tone generator:** a sine wave with two control parameters: pitch and gain;
- **Frequency Modulation (FM):** a three-operator FM synthesizer with sinusoidal oscillators, in which operator 3 modulates operator 2; operator 1 is modulated by the sum of operators 2 and 3; and the output is the sum of operators 1 and 2. Four control parameters are available: central frequency, modulation width and modulation depth of operator 1 in the chain, and gain;
- **Granular:** a granular synthesizer using a recording of a bell strike as input, with four parameters: the grain rate, grain duration, grain starting position, and gain;
- **Benjolin** [19]: a chaotic synthesizer based on two oscillators, feedback modulation, and a shift register mechanism called the Rungler [39]. The Benjolin has eight parameters: frequency of oscillator 1, amount of Rungler modulation

on oscillator 1, frequency of oscillator 2, amount of Rungler modulation on oscillator 2, cutoff frequency of the low-pass filter, filter resonance, amount of Rungler modulating the filter, and modulation depth of oscillator 2 over the filter cutoff frequency.

These synthesizers are progressively more complex to control, with an increasing number of parameters that non-linearly relate to changes in timbre. We compare the scores obtained with different reward functions[1]. Different audio descriptors are used in environment state and in the sound-matching reward. The descriptors used for each synthesizer are listed in Table 1. In out-of-domain cases, the descriptors for setup are selected based on shared timbral characteristics between the two sounds, as well as aesthetic preferences regarding which target attribute the agent should match. All experiments were conducted with a sample rate of 44.1 kHz, a T_u interval of 0.093 seconds; an FFT window of 4098 samples, and a hop size of 2048 samples for computing of audio descriptors.

All models were trained on a workstation running Ubuntu 24.04, equipped with an Intel Core i9-14900KF CPU (24 cores), an NVIDIA GeForce RTX 4090 GPU (24 GB), and 192 GB DDR5-5200 RAM. On this machine, training the RL model for 10^6 steps requires approximately 5 hours, and training the LSTM model for 3000 epochs requires approximately 3 hours. On a MacBook Pro with an Apple M2 Pro chip and 16 GB of unified memory, the same RL and LSTM models require approximately 10 and 4 hours, respectively. In all experiments, RL models were trained for 10^6 steps and LSTM models for 3000 epochs.

Table 1: Audio descriptor sets used for each synthesizer in in-domain sound matching. For out-of-domain objectives, subsets of these descriptors are used depending on the target sound.

Synthesizer	Audio descriptors
Tone generator	RMS, centroid
FM	RMS, centroid, flatness, rolloff
Granular	RMS, centroid, flatness, rolloff, chroma
Benjolin	RMS, centroid, flatness, rolloff, MFCC

These descriptor choices are the results of heuristic knowledge about each synthesizer's timbre, preliminary experiments and aesthetic preferences in case of out-of-domain tasks. In the following section, we report the quantitative performances of models trained on several combination of target and synthesizer sounds. For each model, we report the value of each reward at the end of training, alongside the values of metrics commonly used for evaluation of sound matching. Besides the previously defined multi-resolution STFT (mrSTFT and spectral convergence (SC), we also compute multi-scale spectral distance (MSS). Multi-scale spectral distance, as defined in [17], is the ℓ_1 norm computed over log-scaled mel-spectrogram at multiple time–frequency resolutions.

4.1 In-domain with synthesizer as target

In this task, the agent learns to match a target sound produced by the same synthesizer it controls. Agents are trained as described in Section 3.4, with target parameters fading to randomly selected combinations at random times during episodes. The reward is $R = R_{\text{mrSTFT}} + R_{SC} + R_{ADD} + R_{par}$, rewards as described in Section 3.3. Table 2 reports the values of the rewards computed over 10 evaluation episodes of 1000 steps each.

The best scores are achieved by the tone generator, for which the agent closely match parameter changes. This is expected, as the tone generator has only two parameters, each directly linked to one of the descriptors used during training. The most challenging case is the Benjolin, likely because of the larger number of parameters and because the spectromorphological variation it can produce are not well captured by the descriptors in the state representation, providing the agent with insufficient information about the true state of the environment.

4.2 Out-of-domain with synthesizer as target

In this setting, the agent learns to match the sound produced by a synthesizer different than the one it controls. The target synthesizer's parameters are randomly changed at random times during an episode, creating a moving target. As noted in [37], perfect matches are not achievable in out-of-domain sound matching tasks; outcomes therefore depend heavily on how the agent-controlled synthesizer's sound space relates to that of the target. We report results over representative pairs of target and agent-controlled synthesizers. The results are summarized in Table 3. The rewards used in this setting is $R = R_{ADD} + R_{DDD} + R_{par}$. We chose this formulation after heuristic experiments indicated that R_{mrSTFT} was not effective for out-of-domain tasks, most likely because differences between the two sound spaces do not yield sufficiently similar spectrograms for the metric to guide learning effective. The descriptors used here are, for both the target and agent-controlled synthesizers, the set associated with the agent-controlled synthesizer (see Table 1).

The table and the corresponding audio examples indicate that matching out-of-domain sounds is indeed more challenging than matching in-domain ones. FM matches the tone generator relatively closely because, with low values of modulation width and depth, FM can produce sounds that are relatively similar to the tone generator's sine wave. The agent learns to reduce those two parameters accordingly to track the target. Results are also quite satisfactory when FM and Benjolin attempt to match each other, likely due to similarities in their sound-generation principles—both rely, to varying degrees, on frequency modulation, yelding relatively similar timbres.

4.3 Out-of-domain with audio corpus as target

In this setting, the target sounds are segments drawn from a corpus of audio files. At the start of each episode, a section from one of the corpus files is selected. Each instrumental corpus has a total duration of approximately 45 minutes and is composed of selected tracks from MoisesDB [32] (a dataset for source separation). All instruments considered here are drawn from a subset of tracks by the artist *Firefly*, labeled with the *singer-songwriter* genre, thereby ensuring stylistic and timbral homogeneity within each instrument corpus. The reward used in this setting is $R = R_{ADD} + R_{DDD} + R_{par}$.

We present representative live sound-matching results with different target instruments and different agent-controlled synthesizers. We pair instruments with different synthesizers to

[1]Supplementary materials for this paper, including audio, video examples, and code are available at the paper's repository: https://github.com/vincenzomadaghiele/RL-synth-control

Table 2: Quantitative evaluation of in-domain sound matching over four different synthesizers. We report the values of reward functions R_{ADD}, $R_{\mathbf{mrSTFT}}$ and R_{SC} after training, as well as the values of multi-scale spectral distance (MSS). The values of the reported rewards are negative, because the model aims to maximize the rewards, while values of SC and MSS are positive. In all cases, models with better performances have reward values closer to 0.

Synthesizer	R_{ADD}	R_{mrSTFT}	R_{SC}	MSS
Tone generator	-0.287 ± 0.1	-6.211 ± 2.0	-0.847 ± 0.3	9.644 ± 3.1
FM	-0.350 ± 0.1	-10.605 ± 3.5	-0.599 ± 0.2	10.577 ± 3.5
Granular	-1.207 ± 0.8	-7.644 ± 2.7	-0.504 ± 0.2	7.406 ± 2.6
Benjolin	-2.467 ± 1.5	-14.210 ± 4.5	-0.878 ± 0.3	13.820 ± 4.4

Table 3: Quantitative evaluation of the out-of-domain sound-matching task with the target generated by a synthesizer. We report rewards R_{ADD} and R_{DDD}, and values of metrics mrSTFT, SC and MSS. The values of the reported rewards are negative, because the model aims to maximize the rewards, while values of mrSTFT, SC and MSS are positive. In all cases, models with better performances have reward values closer to 0.

Target	Synthesizer	R_{ADD}	R_{DDD}	mrSTFT	SC	MSS
Tone generator	FM	-0.531 ± 0.5	-0.531 ± 0.5	6.921 ± 2.2	1.490 ± 0.5	11.571 ± 3.6
Granular	Tone generator	-6.299 ± 2.3	-1.431 ± 1.3	6.174 ± 2.0	0.758 ± 0.2	10.209 ± 3.3
Benjolin	FM	-0.117 ± 0.1	-0.055 ± 0.1	10.963 ± 3.5	0.506 ± 0.2	9.802 ± 3.2
FM	Benjolin	-5.081 ± 4.5	-5.158 ± 4.4	13.646 ± 4.3	1.169 ± 0.4	11.635 ± 3.7
Benjolin	Granular	-2.133 ± 2.9	-2.269 ± 3.0	8.754 ± 2.9	0.635 ± 0.2	9.180 ± 3.0
Granular	Benjolin	-4.129 ± 2.7	-4.687 ± 3.6	12.387 ± 3.9	0.501 ± 0.2	11.112 ± 3.5

demonstrate model convergence across varied data and to illustrate alternative application scenarios. The results are summarized in Table 4.

The sound-matching scenarios in this section were selected to emulate musical use-cases. The first case exemplifies basic pitch and envelope following, given the limited capabilities of the tone generator. The results here are strongly influenced by how pitch and amplitude are measured (centroid and RMS in this case), as well as by the synthesizer's pitch and amplitude ranges. If these do not align with those of the target sound, the matching can be inaccurate.

The remaining three scenarios illustrate more idiosyncratic musical applications. In the bass-FM case, the goal was to match the bass timbre using the FM synthesizer. To this end, we employed the FM-associated descriptors from Table 1. As evident from the audio examples, the agent tracks the general sonic behavior of the bass but does not fully capture its timbral idiosyncrasies.

The objective of the granular synthesizer matching the guitar was harmonization. In this setup, the granular synthesizer was trained to match the guitar's RMS and chroma descriptors. The guitar track includes strumming and arpeggios, yielding complex chroma variations influenced by factors beyond harmony. Consequently, the resulting chroma matching reflects timbral aspects beyond pitch class and and does not always behave as expected.

The final case involves the Benjolin matching drums. Here, the aesthetic objective was for the Benjolin following the rhythm and timbre of the drums. Accordingly, the descriptors used were RMS, spectral flatness, spectral centroid and spectral rolloff. The results are quite satisfactory: the Benjolin adjusts gain and the amount of Rungler modulation on oscillator 2 to track the drummer's rhythm across varied scenarios, and modifies its timbre by changing the oscillator 1 frequency and its Rungler modulation.

5 Discussion

We identify that multi-resolution STFT and absolute descriptor distance are effective RL reward functions for in-domain matching. Experiments indicate that suitable rewards for out-of-domain matching are case-specific and depend on the desired aesthetic results as well as on the relations between timbres of the instruments involved. The difference-of-descriptor-distances reward is effective for out-of-domain cases with a synthesizer as the target; using it alone can lead to unpredictable system behaviors, while combining it with other rewards can improve performances. Experiments in out-of-domain matching also indicate that the choice of descriptors significantly affects model convergence, especially when the target sound is a musical performance, where descriptors values are highly idiosyncratic. This suggests that the RL model could benefit from more expressive representations of the target audio stream, such as learned embedding, sequence models, or multi-granular features. Preliminary experiments have shown that this RL model tends to perform better with few descriptors; possibly larger models would be able to accommodate more expressive representations, however this requires further testing.

The value ranges of the metrics obtained using the method we propose do not reach the quality of the state of the art in offline sound matching. To the best of the author's knowledge, there are no standard benchmarks to compare the performances of our model with other models specifically optimized for online sound matching tasks controlling synthesis engines. We have tested the performances of our method over a wide range of synthesis techniques (FM, granular, chaotic), and the method seems to obtain fairly consistent performances across the tested scenarios. The model's performances seem to decrease as the number of parameters and the complexity of the synthesis engine increases, as we can notice for instance in the cases in which Benjolin is controlled by the agent.

Table 4: Experimental results for out-of-domain sound matching with a corpus of instrumental audio tracks as target. We report rewards R_{ADD} and R_{DDD}, and values of metrics mrSTFT, SC and MSS. The values of the reported rewards are negative, because the model aims to maximize the rewards, while values of mrSTFT, SC and MSS are positive. In all cases, models with better performances have reward values closer to 0.

Target	Synthesizer	R_{ADD}	R_{DDD}	mrSTFT	SC	MSS
Bass	Tone generator	-0.260 ± 0.4	-0.283 ± 0.5	12.218 ± 4.1	0.495 ± 0.2	11.022 ± 3.9
Bass	FM	-0.595 ± 0.7	-0.461 ± 0.6	8.974 ± 3.4	0.397 ± 0.1	8.994 ± 3.1
Guitar	Granular	-0.364 ± 0.3	-0.265 ± 0.2	9.141 ± 3.3	0.681 ± 0.2	10.072 ± 3.2
Drums	Benjolin	-3.583 ± 2.2	-2.759 ± 2.1	13.514 ± 4.3	0.619 ± 0.2	11.485 ± 3.7

One of the aims of this work is to generate divergent mappings beyond the in-domain use cases, thereby obtaining responsive behaviors from the agents during live performance. We have briefly tested the live application of this model by training the granular synthesizer to match live electric guitar, and the Benjolin to match a solo snare drum; video examples of these live interactions are available on the paper's companion page. In both cases, the live responsiveness of the agent, paired with the stark difference in timbre between target instrument and synthesizer, seemed to provide a contrasting effect, which could be used as a live sound augmentation of the instruments' acoustic or electronic sound. In these two tested cases, the imperfections of the learned policies could sometimes generate divergent behavior, for example responding with high-pitched Benjolin sounds during moments of silence in the snare-drum performance. These divergent behaviors, while resulting from imperfectly learned policies (in this example, missed matching of RMS), are still closely paired with the musician's live performance and can be used creatively as an instrumental augmentation, for example controlling the volume of the agent's synthesizer with a volume pedal during performance to mitigate occasional unwanted behavior.

6 Conclusions

In this paper, we proposed a novel method for live sound matching using reinforcement learning. Our experiments show that the model is flexible across diverse input sounds and controlled synthesizers, provided that appropriate descriptors and reward functions are chosen. While the technique we propose is not able to match the performances of the state of the art in offline sound matching with the setup tested in this paper, the novel approach we test has potential for improvement in both strict live sound matching applications, and to establish possibly divergent mapping for responsive, semi-autonomous artificial behaviors during live performance. Experiments with out-of-domain sound matching using a corpus of audio recordings suggest that the method can be extended to develop responsive behaviors more complex than sound matching, provided that the desired aesthetic result can be captured by an appropriate reward function. In future work, we plan to explore this direction by experimenting with user preferences and multi-granular rewards that encode more complex aspects of musical control.

7 Ethical Standards

This work was funded by the Department of Musicology at the University of Oslo as part of the first author's doctoral research fellowship. The research relies on open-source software frameworks and publicly available data. MoisesDB, used as a dataset in this work, is released under Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International license. We have released the code developed for this project as open source software to ensure reproducibility. The authors report no conflicts of interest. The computational demands of the machine-learning models used in this work are comparable to an individual's daily computer usage; therefore, the machine-learning component was not a major contributor to the project's environmental impact.

Acknowledgments

The authors would like to thank Jordie Shier for the inspiring conversations about timbre remapping and RL modeling, and Aleksander Tidemann for his drum performance in the second video example.

References

[1] Sercan Ö Arık, Heewoo Jun, and Gregory Diamos. 2018. Fast spectrogram inversion using multi-head convolutional neural networks. *IEEE Signal Processing Letters* 26, 1 (2018), 94–98.

[2] Gérard Assayag, Laurent Bonnasse-Gahot, and Joakim Borg. 2022. Cocreative Interaction: Somax2 and the REACH Project. *Computer Music Journal* 46, 4 (2022), 7–25.

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

[4] Joakim Borg, Gérard Assayag, and Mikhail Malt. 2022. Somax 2 a reactive multi-agent environment for co-improvisation. In *SMC 2022-Sound Music & Computing*. Sound and Music Computing conference, Saint-Étienne, France.

[5] Øyvind Brandtsegg, Sigurd Saue, and Thom Johansen. 2011. A Modulation Matrix for Complex Parameter Sets.. In *NIME*. New Interfaces for Musical Expression, Oslo, Norway, 316–319.

[6] Fred Bruford, Frederik Blang, and Shahan Nercessian. 2024. Synthesizer Sound Matching Using Audio Spectrogram Transformers. *Proceedings of the 27th International Conference on Digital Audio Effects (DAFx24)* (2024).

[7] Antoine Caillon and Philippe Esling. 2021. RAVE: A variational autoencoder for fast and high-quality neural audio synthesis. *ArXiv* abs/2111.05011 (2021).

[8] Michelle Carney, Chong Li, Edwin Toh, Nida Zada, Ping Yu, and Jesse Engel. 2021. Tone Transfer: In-Browser Interactive Neural Audio Synthesis. In *Joint Proceedings of the ACM IUI 2021 Workshops*. CEUR Workshop Proceedings, ACMIUI-WS, College Station, United States.

[9] SignalFlow Contributors. 2024. SignalFlow: Sound Synthesis Framework. https://signalflow.dev/

[10] Palle Dahlstedt. 2001. Creating and exploring huge parameter spaces: Interactive evolution as a tool for sound generation. In *International Computer Music Conference*. ICMC, La Habana, Cuba.

[11] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. 2020. DDSP: Differentiable Digital Signal Processing. *International Conference on Learning Representations* abs/2001.04643 (2020).

[12] Philippe Esling, Naoto Masuda, Axel Bardet, Robin Despres, and Axel Chemla-Romeu-Santos. 2019. Universal audio synthesizer control with normalizing flows. In *Proceedings of the 22nd International Conference on Digital Audio Effects (DAFx-19)*. DAFx, Birmingham, UK.

[13] Stefano Fasciani. 2014. *Voice Controlled interface for Digital Musical Instrument*. Ph. D. Dissertation. National University of Singapore.

[14] Stefano Fasciani and Lonce Wyse. 2018. Vocal control of sound synthesis personalized by unsupervised machine listening and learning. *Computer Music Journal* 42, 1 (2018), 37–59.

[15] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. Pmlr, International Conference on Machine Learning, Stockholm, Sweden, 1861–1870.

[16] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel,

et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905* (2018).

[17] Ben Hayes, Charalampos Saitis, and GyÃ¶rgy Fazekas. 2025. Audio synthesizer inversion in symmetric parameter spaces with approximately equivariant flow matching. *Proceedings of the 26th International Society for Music Information Retrieval Conference* (Sept. 2025). https://doi.org/10.5281/zenodo.17717337

[18] Ben Hayes, Jordie Shier, György Fazekas, Andrew McPherson, and Charalampos Saitis. 2024. A review of differentiable digital signal processing for music and speech synthesis. *Frontiers in Signal Processing* 3 (2024), 1284100.

[19] Rob Hordijk. 2009. The Blippoo Box: A chaotic electronic music instrument, bent by design. *Leonardo Music Journal* 19 (2009), 35–43.

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

[21] Andy Hunt, Marcelo M Wanderley, and Ross Kirk. 2000. Towards a model for instrumental mapping in expert musical interaction. In *International Computer Music Conference*. Berlin, ICMC, Berlin, Germany.

[22] Katsutoshi Itoyama and Hiroshi G Okuno. 2014. Parameter estimation of virtual musical instrument synthesizers. In *International Computer Music Conference*. ICMC, Athens, Greece.

[23] Vincenzo Madaghiele and Stefano Fasciani. 2024. A listening agent for live control of synthesis parameters using reinforcement learning. In *Proceedings of AI and Music Creativity Conference (AIMC)*. Oxford (UK).

[24] Naotake Masuda and Daisuke Saito. 2021. Synthesizer Sound Matching with Differentiable DSP. In *Proceedings of the 22nd International Society for Music Information Retrieval*. ISMIR, Online, 428–434.

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

[26] Carlos Mateo and Juan Antonio Talavera. 2020. Bridging the gap between the short-time Fourier transform (STFT), wavelets, the constant-Q transform and multi-resolution STFT. *Signal, Image and Video Processing* 14, 8 (2020), 1535–1543.

[27] Stephen McAdams and Jean-Christophe Cunible. 1992. Perception of timbral analogies. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* 336, 1278 (1992), 383–389.

[28] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. 2015. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, Vol. 8. 18–25.

[29] Christopher Mitcheltree, Hao Hao Tan, and Joshua D Reiss. 2025. Modulation Discovery with Differentiable Digital Signal Processing. In *2025 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, WASPAA, Tahoe City, California, USA, 1–5.

[30] Jérôme Nika, Ken Déguernel, Axel Chemla-Romeu-Santos, Emmanuel Vincent, and Gérard Assayag. 2017. DYCI2 agents: merging the "free", "reactive", and "scenario-based" music generation paradigms. In *International Computer Music Conference*. ICMC, Shanghai, China.

[31] Sindhu Padakandla. 2021. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–25.

[32] Igor Pereira, Felipe Araújo, Filip Korzeniowski, and Richard Vogl. 2023. Moisesdb: A dataset for source separation beyond 4-stems. *Proceedings of the 24th Int. Society for Music Information Retrieval Conference* (2023).

[33] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22, 268 (2021), 1–8. http://jmlr.org/papers/v22/20-1364.html

[34] Diemo Schwarz. 2000. A system for data-driven concatenative sound synthesis. In *Digital Audio Effects (DAFx)*. DAFx, Verona, Italy, 97–102.

[35] Jordie Shier, Rodrigo Constanzo, Charalampos Saitis, Andrew Robertson, Andrew McPherson, et al. 2025. Designing Percussive Timbre Remappings: Negotiating Audio Representations and Evolving Parameter Spaces. *Proceedings of New Interfaces for Musical Expression (NIME* (2025).

[36] Jordie Shier, Charalampos Saitis, Andrew Robertson, and Andrew McPherson. 2024. Real-time Timbre Remapping with Differentiable DSP. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, S M Astrid Bin and Courtney N. Reed (Eds.). NIME, Utrecht, Netherlands, Article 55, 9 pages. https://doi.org/10.5281/zenodo.13904884

[37] Wonchul Shin and Kyogu Lee. 2025. SynthRL: Cross-domain Synthesizer Sound Matching via Reinforcement Learning. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, James Kwok (Ed.). International Joint Conferences on Artificial Intelligence Organization, Montreal, Canada, 10162–10170. AI, Arts & Creativity.

[38] Riccardo Simionato and Stefano Fasciani. 2023. Fully Conditioned and Low Latency Black-boc Model of Analog Compression. In *Proceedings of the International Conference on Digital Audio Effects (DAFx23)*. Aalborg University Copenhagen, Copenhagen, Denmark, 287–296.

[39] Dan Slater. 1998. Chaotic sound synthesis. *Computer Music Journal* 22, 2 (1998), 12–19.

[40] Dan Stowell. 2010. *Making music through real-time voice timbre analysis: machine learning and timbral control.* Ph. D. Dissertation. School of Electronic Engineering and Computer Science, Queen Mary University of London.

[41] Dan Stowell and Mark D. Plumbley. 2010. Cross-associating unlabelled timbre distributions to create expressive musical mappings. In *Proceedings of the First Workshop on Applications of Pattern Analysis (Proceedings of Machine Learning Research, Vol. 11)*, Tom Diethe, Nello Cristianini, and John Shawe-Taylor (Eds.). PMLR, Cumberland Lodge, Windsor, UK, 28–35.

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

[43] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. 2025. Gymnasium: A Standard Interface for Reinforcement Learning Environments. arXiv:2407.17032 [cs.LG]

[44] Doug Van Nort, Marcelo M Wanderley, and Philippe Depalle. 2014. Mapping control structures for sound synthesis: Functional and topological perspectives. *Computer Music Journal* 38, 3 (2014), 6–22.

[45] Marcelo M Wanderley and Philippe Depalle. 2004. Gestural control of sound synthesis. *Proc. IEEE* 92, 4 (2004), 632–644.

[46] Travis J West, Marcelo Wanderley, and Baptiste Caramiaux. 2020. Making Mappings: Examining the Design Process. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Romain Michon and Franziska Schroeder (Eds.). Birmingham City University, Birmingham, UK, 291–296. https://doi.org/10.5281/zenodo.4813365

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