Music Synthesizer Parameter Estimation Using Analysis by Synthesis

Student: Siqi Hao[1,2]
Mentor: Paul Vicol[1,2]
Supervisors: Sageev Oore[2,3], Roger Grosse[1,2]
[1] *University of Toronto,* [2] *Vector Institute,* [3] *Dalhousie University*

1. Introduction

Sound synthesizers used by professional musicians and composers have many settings that affect aspects of the sound produced. Adjusting these settings to achieve a desired sound is a difficult and time-consuming process; some synthesizers have hundreds of parameters, leading to a combinatorial explosion in the number of possible configurations. Due to the difficulty of this task, there has been increasing interest in automated parameter estimation using machine learning [**? ? ?**]. Gabrielli et al. proposed a supervised convolutional model to estimate parameters of a pipe organ synthesizer [**?**]. Yee-King et al. evaluated three neural networks (MLP, LSTM, bi-directional LSTM) on the VST (Virtual Studio Technology) synthesizers [**?**].
We explore parameter estimation of musical sounds via analysis by synthesis [**?**], which aims to find the hidden causes that generated the observed data. Given audio from a synthesizer, we aim to recover the parameters which were used to generate that audio. Our model can be used to predict MIDI settings from audio samples, enabling musicians to mimic some notes created by a specific synthesizer (Figure 1). Here, we focus on estimating the parameters of a physics-based synthesizer called the *Karplus-Strong algorithm.* In our approach, we represent audio samples as images using the constant Q transform (CQT) to generate spectrograms. In the following subsections, we provide an overview of the Karplus-Strong algorithm and CQT spectrograms.
Karplus-Strong. Sound is produced by vibrations, and the exact sound of an instrument depends on its physical characteristics. *Physics-based audio synthesis* models the aspects of the process by which sound is generated. One such synthesizer is the Karplus-Strong algorithm [**?**], which is a simple but

(a) MIDI Controller.

(b) Post-processing effects.

Figure 1: We focus on recovering the settings of parameters that influence synthesized audio. The settings, such as those used by MIDI controllers (left) and guitar pedals (right), are often used to achieve effects such as reverb and phaser.

effective model of a plucked string. Starting from a buffer of random noise, it uses a filter to average the current and the previous samples. The averaged result is used as the output for the current timestep, and is fed back into the buffer for future timesteps. Repeating this process simulates the energy decay that occurs after a string is plucked, due to the the fact that most energy is used to push the air to create sound and some gets lost due to the air drag.

Constant Q transform (CQT). CQT [**?**] is a visual representation of audio that is well-suited to musical data (see Figure 2). CQT represents frequencies on a logarithmic scale, which corresponds to human perception [**?**]. Due to this characteristic, the spectrogram produced via CQT exhibits a higher temporal resolution for higher frequencies, and a higher frequency resolution for lower frequencies. In all, CQT spectrogram has qualities that make it suitable for analyzing music audio signals.

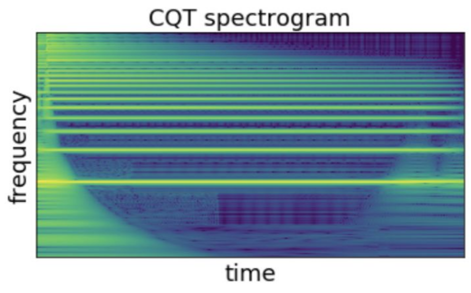

Figure 2: An example CQT spectrogram.

2. Approach

We use the Karplus-Strong algorithm as a synthesizer to generate audio waveforms from physical parameters such as the *pitch* and *smoothing factor*, where the pitch refers to the frequency of the sound and the smoothing factor is related to the damping of a string. Once we have an audio sample produced by the Karplus-Strong algorithm, we explore adding post-processing effects such as reverb and phasing, and analyze the parameters used to implement these effects, such as *room scale* and *phasing decay*, respectively. The room scale simulates the reverberation of sound in rooms of different sizes. The decay factor affects how much the delayed phase information remains over time.

We first attempted to predict the parameters that generated a CQT spectrogram using a convolutional neural network (CNN), but we found that directly predicting the parameters given a spectrogram is challenging, just as it is difficult for most people to identify a single note played on a piano. Instead, we propose an iterative approach for parameter estimation, motivated by the intuition that if one hears two notes in succession, it is easier to tell whether the second has a higher or lower pitch than the first, than to identify the first note without resorting to any comparisons.

Thus, instead of predicting the parameters from one spectrogram, we use a CNN to regress the difference between the parameters underlying two CQT spectrograms: we refer to this difference as the *delta* parameter value (see Figure 3). In order to train a model for this task, we generated a synthetic dataset:

$$\mathcal{D} = \{(\boldsymbol{p_i}, \boldsymbol{CQT_i}, \boldsymbol{p'_i}, \boldsymbol{CQT'_i})\}_{i=1}^{N} \tag{1}$$

consisting of *pairs* of parameter vectors $(\boldsymbol{p_i}, \boldsymbol{p'_i})$ and their corresponding CQT spectrograms. To ensure that our dataset covers a wide range of possible deltas, we first chose the delta values $\Delta\boldsymbol{p_i}$; then we randomly picked the initial parameter value, $\boldsymbol{p_i}$, and we computed the second parameter value by subtracting the delta from the first parameter, $\boldsymbol{p'_i} = \boldsymbol{p_i} - \Delta\boldsymbol{p_i}$. We generated $\boldsymbol{CQT_i}$ and $\boldsymbol{CQT'_i}$ using the Karplus-Strong algorithm based on $\boldsymbol{p_i}$ and $\boldsymbol{p'_i}$, respectively.

Our method works as follows: starting from random initial parameters, we repeatedly predict delta parameters using the trained model and update the current parameters in order to match the target spectrogram. Typically, the updated parameters will oscillate stably in 100 iterations. These steps are formalized in Algorithm 1.

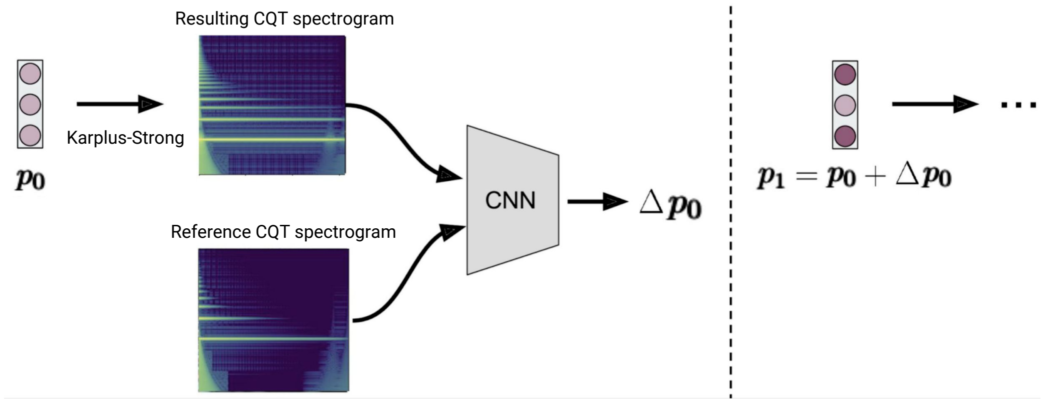

Figure 3: **Iterative Parameter Update Model**

Algorithm 1: Iterative Parameter Updates

```
Input  : $CQT_{REF}$ (The spectrogram for which we want parameters)
Output: $p$ (The underlying physical parameters of the sound)
$p_0 \leftarrow$ GenerateRandomParameters()
$t \leftarrow 0$
$iterations \leftarrow 100$
while $t < iterations$ do
    $CQT_{pred} \leftarrow$ KarplusStrong($p_t$)
    $\Delta p_t \leftarrow$ PredictDelta($CQT_{pred}, CQT_{REF}$)
    $p_{t+1} \leftarrow p_t + \Delta p_t, \ t \leftarrow t + 1$
end
return $p_t$
```

3. Analysis

Our approach learns to predict large delta parameters at first, to move rapidly towards the correct parameter values, and then to gradually fine-tune the parameters with smaller deltas. Our model is able to adjust several parameters simultaneously: the pitch, smoothing factor, room scale, and phasing decay factor. Figure 4 shows the results of our model over multiple update iterations. We regressed all four parameters simultaneously, and found that the model was able to converge fairly accurately to values near the reference.

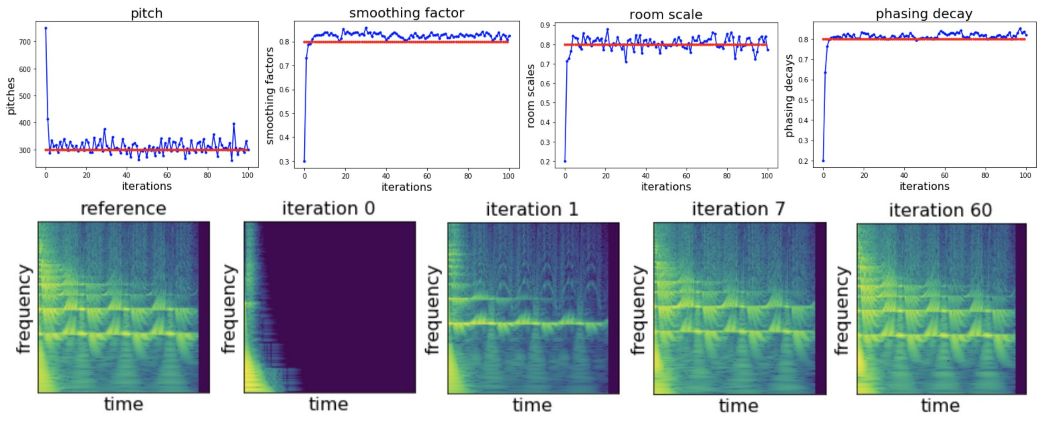

Figure 4: **Iterative Parameter Updates for 4 Parameters. Top:** For each parameter, we show the updates made by our model over 100 iterations. The red line indicates the reference parameter value. **Bottom:** The left-most spectrogram is the reference, for which we wish to recover accurate parameter values; the other spectrograms are generated from the parameters produced by successive iterations of our model (iterations 0, 1, 7, and 60).

Effect of Kernel Shape. Due to the fact that pitch shifting corresponds to vertical translation of CQT spectrograms, we investigated the effect of using different convolutional kernel sizes in our CNN. We found that the shape of the kernels affects the learning capacity. Figure 5 shows the results obtained by variants of our model using different kernel sizes. Comparing the **top left** and **top right** plots, we show that increasing the height of the kernel led to a better prediction of large delta pitches. The vertical kernel (**top left**) performs better than the horizontal kernel (**bottom left**) in the sense that the former captures small delta pitches better. The small, square kernel (**bottom right**) behaves worst in predicting both large and small delta pitches.

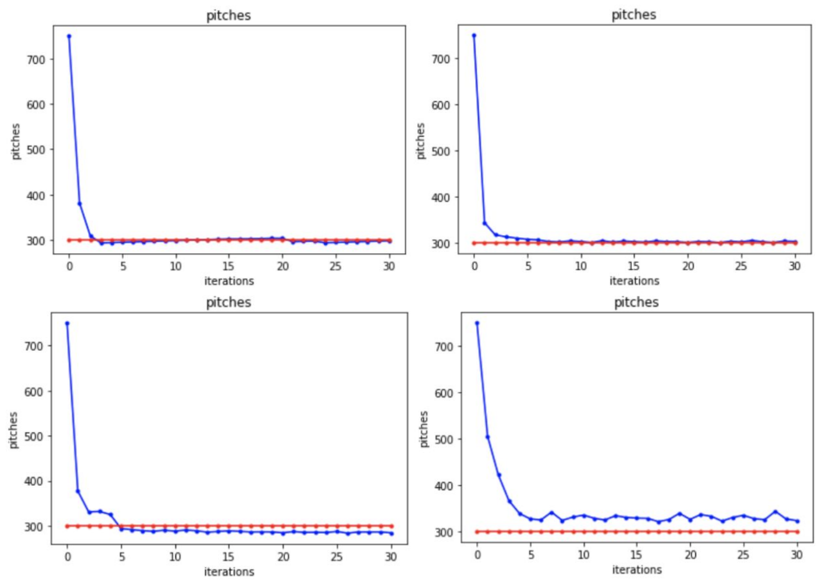

Figure 5: **Iterative Parameter Updates with Different Kernel Sizes.** The red line indicates the reference parameter value. The size of the input spectrograms to the CNN is 336×336. From **left** to **right**, from **top** to **bottom**, the kernel sizes are 112×1, 168×1, 1×112, and 5×5, respectively.

4. Conclusion

We introduced a method to reverse-engineer the physical parameters that give rise to an observed audio sample. We first train a CNN to predict the differences in parameter values between pairs of CQT spectrograms; then, starting from random initial parameters, we iteratively refine the parameters by generating the corresponding CQT spectrogram, using the CNN to predict the parameter delta $\Delta \boldsymbol{p_t}$ needed to move closer to the reference spectrogram, and updating the parameters as $\boldsymbol{p_{t+1}} = \boldsymbol{p_t} + \Delta \boldsymbol{p_t}$. We demonstrated that our approach is powerful enough to simultaneously regress several parameters that govern both the audio generated by the Karplus-Strong algorithm and the post-processing effects applied to the Karplus-Strong sample (e.g., reverb and phaser effects). One promising future work would be to extend this model to other instrumental sound synthesized by other synthesizers.