Towards realistic MIDI instrument synthesizers

Rodrigo Castellon
Stanford University

Chris Donahue
Stanford University

Percy Liang
Stanford University

Abstract

Despite decades of research and widespread commercial interest, synthesis of realistic musical instrument audio from real-time MIDI input remains elusive. In this work, we propose a learning-based system for creating bespoke, MIDI-driven instrument synthesizers. Our system takes as training data audio recordings of a particular instrument and, optionally, scores corresponding to those recordings. To prepare the data, we use an off-the-shelf transcription (or score alignment) strategy to create time-aligned MIDI and audio. Then, we train recurrent neural networks to learn a distribution over instrument audio given MIDI. We demonstrate our method on the task of violin synthesis. Our user study indicates that our proposed system can learn a real-time-capable, MIDI-controllable violin synthesizer which produces fairly realistic audio given only a few minutes of amateur recordings.

1 Introduction

The ability to synthesize MIDI (discrete musical notation) as *realistic* instrument-specific audio—audio which is indistinguishable from a recording of a human musician performing a real instrument—would expand the creative potentials of composers and music producers alike. Despite decades of progress, existing solutions fall short. For example, most digital pianos ship with the ability to synthesize instruments via frequency modulation [1], but the resulting audio is not realistic. Professional musicians instead turn to *concatenative* synthesizers which stitch together high-fidelity recordings of humans playing individual notes on specific instruments, but such synthesizers are expensive to create and require painstaking configuration to achieve truly realistic audio.

Recent work has used machine learning to automate the task of building realistic instrument synthesizers. Several strategies [15, 3, 14, 2, 11, 6] are capable of producing instrument audio with varying degrees of quality, but none of them can be controlled via MIDI. Engel et al. [4, 5] learn to generate individual instrument note recordings which can be stitched together to synthesize MIDI, but the result does not sound realistic. Hawthorne et al. [7] learn to generate realistic piano audio from MIDI, but their strategy is specific to the piano and requires hundreds of hours of training data.

In this work, we propose a system which learns to synthesize instrument audio from MIDI, and can be applied to any monophonic instrument. Our system takes as input audio recordings of a particular instrument and, optionally, scores corresponding to those recordings. To construct training data, we convert the audio into time-aligned MIDI by either transcribing the audio or (preferably) aligning a score if available. The resultant synthesizers are capable of synthesizing instrument audio from MIDI in both offline (rendering musical scores) and real-time (rendering human performances) settings.

Given only a few minutes of amateur recordings as input, our system is capable of producing audio which humans perceive as far more realistic than that of a freely-available concatenative synthesizer [13]. However, the audio from our system is not quite as realistic as that of a commercially-available solution [8] which stitches together high-fidelity recordings of professional violinists.[1]

[1] Sound examples: `https://rodrigo-castellon.github.io/midi2params`
Code: `https://github.com/rodrigo-castellon/midi2params`

4th Workshop on Machine Learning for Creativity and Design at NeurIPS 2020, Vancouver, Canada.

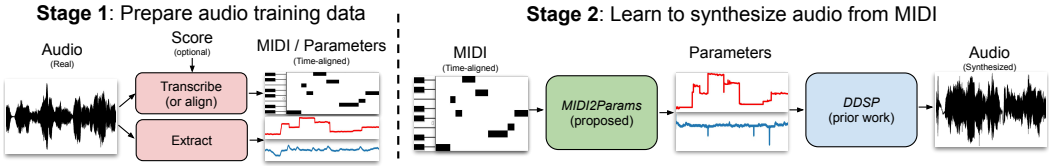

Figure 1: Overview of our proposed two-stage system for learning MIDI instrument synthesizers.

Table 1: Human-judged realism (higher better) for our proposed system (*) and several "basslines".

System	MIDI Control	Realism
Concatenative (freely-available)	Yes	0.19
DDSP(Heuristic(MIDI))	Yes	0.30
* DDSP(MIDI2Params$_{\text{transcribed}}$(MIDI))	Yes	0.41
* DDSP(MIDI2Params$_{\text{aligned}}$(MIDI))	Yes	0.56
Concatenative (commercially-available)	Yes	0.61
DDSP (oracle)	No	0.59
Human recordings	No	0.82

2 System Overview

Our system builds on the *DDSP* method proposed by Engel et al. [6]. DDSP is capable of producing realistic instrument audio given only a few minutes of recordings of an instrument, but by default is controlled by high-rate *parameters* instead of the more-convenient MIDI. Hence, our system introduces a module called *MIDI2Params* which estimates DDSP-style inputs (described below) from MIDI. Our system (Figure 1) involves two stages: a *data preparation* stage, and a *learning* stage.

Data preparation. We first convert the input audio into appropriate training data for MIDI2Params and DDSP. Specifically, we transcribe the audio to time-aligned MIDI using an off-the-shelf method (or optionally align a MIDI score if available), and also convert the audio into corresponding DDSP parameters. These parameters are two time-varying signals that describe the *fundamental frequency* (musical pitch) and *loudness* (musical dynamics) of the audio signal. Details in Appendix A.

Learning. From the first stage, we have aligned audio, MIDI, and parameters. To synthesize, our system learns to convert MIDI to audio, using the parameters as an intermediary. We first train a *MIDI2Params* model: an autoregressive RNN which learns to estimate suitable parameters from MIDI (details in Appendix B). We then train DDSP using default settings which learns to convert parameters to audio. The composition of these two models yields a MIDI-controllable synthesizer.

We note that an upcoming publication by Jonason et al. [9] proposes a similar system, though their system uses bidirectional (non-causal) RNNs which are not suitable for real-time use.

3 Experiments and User Study

As a test bed for comparing our method to other strategies, we collected 15 minutes of amateur recordings from a human violinist. We trained a DDSP model on this data and two MIDI2Params models, one trained on MIDI which was transcribed from the audio (MIDI2Params$_{\text{transcribed}}$) and one trained on MIDI which was produced by aligning a MIDI score to the audio (MIDI2Params$_{\text{aligned}}$). We compare these two MIDI2Params-based systems to (1) a freely-available concatenative synthesizer, (2) a commercially-available concatenative synthesizer, (3) a heuristic to replace MIDI2Params, (4) DDSP, and (5) human recordings. Details of these baselines and oracles are in Appendix C.

To evaluate the realism of our proposed strategy, we ran a user study on Amazon Mechanical Turk. We play users a pair of ten second clips (same musical material) from two different instrument synthesizers (randomly selected without replacement), and ask which of the two clips they find to be more realistic. In Table 1, we report the *realism score* of each method: the percentage of comparisons where that method was preferred above any other. Our system achieves a higher score (0.56) than both a heuristic (0.30) and a freely-available concatenative synthesizer (0.19), though falls short of the commercial solution (0.61). Further discussion and user study details are in Appendix D.

Broader Impact

The pursuit of more realistic MIDI instrument synthesizers has the unfortunate potential to supplant human performers of real instruments. To some extent, this may have already taken place as a result of concatenative instrument synthesizers, which are already readily employed e.g. in modern film scores and pop music. It is unclear if a continued improvement in the perceptual quality of instrument synthesizers (by machine learning or otherwise) may further supplant human musicians. Despite the potential downsides, increased access to realistic, efficient, and affordable instrument synthesizers could open up practices such as orchestral composition and film scoring to a much wider audience, continuing the recent trend of "bedroom production"—the commodification of music production techniques which were once only available in professional recording studios.

Acknowledgments

We would like to thank Ismail Ercan for contributing violin performances as training data. We also thank John Hewitt and William Wang for thoughtful feedback and suggestions.

References

[1] John M Chowning. The synthesis of complex audio spectra by means of frequency modulation. *Journal of the Audio Engineering Society*, 1973.

[2] Sander Dieleman, Aaron van den Oord, and Karen Simonyan. The challenge of realistic music generation: modelling raw audio at scale. In *NeurIPS*, 2018.

[3] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. In *ICLR*, 2019.

[4] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. Neural audio synthesis of musical notes with WaveNet autoencoders. In *ICML*, 2017.

[5] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. GANSynth: Adversarial neural audio synthesis. In *ICLR*, 2019.

[6] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. DDSP: Differentiable digital signal processing. In *ICLR*, 2020.

[7] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *ICLR*, 2018.

[8] East West Communications Inc. East/West Hollywood Solo Instruments. URL `http://www.soundsonline.com/hollywood-solo-series`.

[9] Nicolas Jonason, Bob Sturm, and Carl Thomé. The control-synthesis approach for making expressive and controllable neural music synthesizers. In *Joint Conference on AI Music Creativity*, 2020.

[10] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello. CREPE: A convolutional representation for pitch estimation. In *ICASSP*, 2018.

[11] Kundan Kumar, Rithesh Kumar, Thibault de Boissiere, Lucas Gestin, Wei Zhen Teoh, Jose Sotelo, Alexandre de Brébisson, Yoshua Bengio, and Aaron C Courville. MelGAN: Generative adversarial networks for conditional waveform synthesis. In *NeurIPS*, 2019.

[12] Matthias Mauch, Chris Cannam, Rachel Bittner, George Fazekas, Justin Salamon, Jiajie Dai, Juan Bello, and Simon Dixon. Computer-aided melody note transcription using the tony software: Accuracy and efficiency. 2015.

[13] Tom Moebert. FluidSynth. URL `http://www.fluidsynth.org/`.

[14] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taigman. A universal music translation network. In *ICLR*, 2019.

[15] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio. *arXiv:1609.03499*, 2016.

[16] Colin Raffel and Daniel PW Ellis. Optimizing DTW-based audio-to-MIDI alignment and matching. In *ICASSP*, 2016.

A Dataset and Preparation

We envision our strategy being used by amateur musicians to convert their instrumental performances into playable MIDI instruments. Hence, instead of using studio recordings of professional violinists as training data, we collected a dataset which more accurately reflects potential downstream use of our system: recordings from an amateur violinist on a hand-held microphone (Zoom H2). The dataset contains around 20 minutes of recordings across six pieces; four are used for training and one each for validation and testing. We release the dataset alongside this work.[2]

A.1 Preparation

To create supervised training data for both our MIDI2Params model and DDSP (i.e., Params2Audio), we first convert the audio from our dataset into both time-aligned MIDI and DDSP-style audio parameters.

MIDI by transcription. One strategy for converting the audio into MIDI is to *transcribe* it, i.e., directly estimate MIDI from the audio recording. To do this, we use the proprietary Melodyne 5 software.[3] For each violin recording, we do the following:

1. Open Melodyne 5
2. Load recording (stored as 48kHz mono 32-bit float WAV file)
3. Melodyne should display "monophonic detection" as file is being analyzed
4. Options -> Time Grid -> Deselect "Active"
5. Options -> Time Grid -> Select "Seconds"
6. Select all notes in timeline (Ctrl+A)
7. File -> Export -> MIDI; entire length; common file for all tracks

While we would have strongly preferred to use freely-available solutions such as Tony [12], we found that Melodyne produced far better transcription results for our dataset.

MIDI by score alignment. Another strategy for "converting" the audio into MIDI is to *align* an existing MIDI score. Alignment is an easier problem than transcription, and as such we recommend this strategy when a MIDI score is available. We collected MIDI scores for all the pieces in our dataset and subsequently used the alignment strategy of Raffel and Ellis [16].[4]

DDSP audio parameters. Because our system uses DDSP off-the-shelf, we use the same methodology as in the original DDSP paper to convert the audio into time-varying fundamental frequency and loudness. Specifically, we use CREPE [10] to extract fundamental frequency, and the recommended heuristic to extract loudness.

The dataset link in the footnote of this page includes not only the audio for our custom dataset but also the results of our MIDI and parameter pre-processing.

B MIDI2Params

Our MIDI2Params module learns to predict the two time-varying DDSP parameters of fundamental frequency and loudness given MIDI as input. These parameters are real-valued and sampled at a rate of 250Hz. Because two human performances of the same musical material may differ, we would like to learn a *distribution* over parameters given MIDI. To this end, we adopt a conditional language modeling strategy over a discretization of these real-valued parameters. Specifically, we learn a model of the form

$$P(D_t^{f_0}, D_t^l \mid D_{<t}^{f_0}, D_{<t}^l, M_{\leq t}^p, M_{\leq t}^{\text{on}}, M_{\leq t}^{\text{off}}).$$

[2]Dataset: `https://drive.google.com/drive/folders/1MZhRKKcu2DgbkRoClUmckgHo8RKGUhqf`

[3]Free trial of Melodyne: `https://www.celemony.com/en/trial`

[4]Code for aligning MIDI to audio: `https://github.com/craffel/align_midi`

Here, $D_t^{f_0}$ and D_t^l represent the fundamental frequency and loudness parameters (outputs) at timestep t. M_t^p, M_t^{on}, and M_t^{off} describe the MIDI data (inputs) at timestep t—respectively, the MIDI pitch and whether or not there is an onset or offset at this timestep.

Because we can directly convert MIDI pitch into the appropriate fundamental frequency, we observe that it is sufficient to have our model predict the expressive *deviation* from this frequency. This allows our strategy to model expressive phenomena like vibrato (frequency oscillation centered around a pitch) while constraining it to always reproduce the input MIDI notes properly. Hence, $D_t^{f_0}$ is the amount of deviation from the "correct" fundamental frequency expressed in *cents* (a musical semitone consists of 100 cents).

All quantities were represented in "one-hot" form, where the quantities ranged $-50 \leq D_t^{f_0} \leq 50$, $-120 \leq D_t^l \leq 0$, $0 \leq M_t^p \leq 128$, $M_t^{\text{on}} \in \{0, 1\}$, $M_t^{\text{off}} \in \{0, 1\}$. We train our model by minimizing the cross entropy between the model's predictions and the parameters from the real data. At test time, we sample from the model's distribution without modification.

Code and hyperparameters can be found at `https://github.com/rodrigo-castellon/midi2params`.

C Baselines

We compared our proposed MIDI2Params-based systems to several baselines and oracles. All strategies use the same MIDI file as input: the MIDI score for the single piece in the test split after alignment to the original human audio. A description of each strategy follows:

Concatenative (freely-available) The "low bar" for instrument synthesis. This baseline is from a freely-available concatenative synthesizer called FluidSynth.[5] This synthesizer concatenates together recordings of a synthetic violin. It is commonly used to render MIDI files as audio for quick previewing, but is generally perceived to sound unrealistic.

Concatenative (commercially-available) The "state-of-the-art" for instrument synthesis; what a modern professional musician would employ to get the best sound short when a real musician is out of reach. These recordings are from a proprietary concatenative synthesizer called EastWest Hollywood Solo Violin Gold.[6] This synthesizer pairs high-fidelity studio recordings of professional violinists playing individual notes with sophisticated logic to stitch these recordings together in a realistic fashion. We use the "05 Legato > Solo Violin Legato Expressive Vibrato" preset.

DDSP(Heuristic(MIDI)) The "low bar" for our MIDI2Params model. This baseline uses a heuristic to convert MIDI into DDSP-style audio parameters. Specifically, we directly convert MIDI pitch to fundamental frequency (adding vibrato) and create loudness by applying a simple piece-wise linear envelope to the MIDI onsets and offsets.

DDSP. The oracle for our MIDI2Params model, i.e., the upper bound of realism that our MIDI2Params model could conceivably reach. For this baseline, instead of estimating DDSP input parameters from MIDI, we run the audio parameters extracted from the human recording directly into the DDSP model trained on our violin dataset.

Human recordings. The gold standard: the original human recording of the test piece.

D User Study Details

We conduct a user study on Amazon Mechanical Turk to measure how realistic our proposed model sounds in comparison to a slew of baselines. The format of our user study is inspired by that of Hawthorne et al. [7]. We release all of the code and sound files needed to re-run our user study.[7]

[5] `http://www.fluidsynth.org/`

[6] `http://www.soundsonline.com/hollywood-solo-series`

[7] User study files: `https://drive.google.com/file/d/12uSA047p1euI1TMUh2OC-dFWyOoSLUsG`

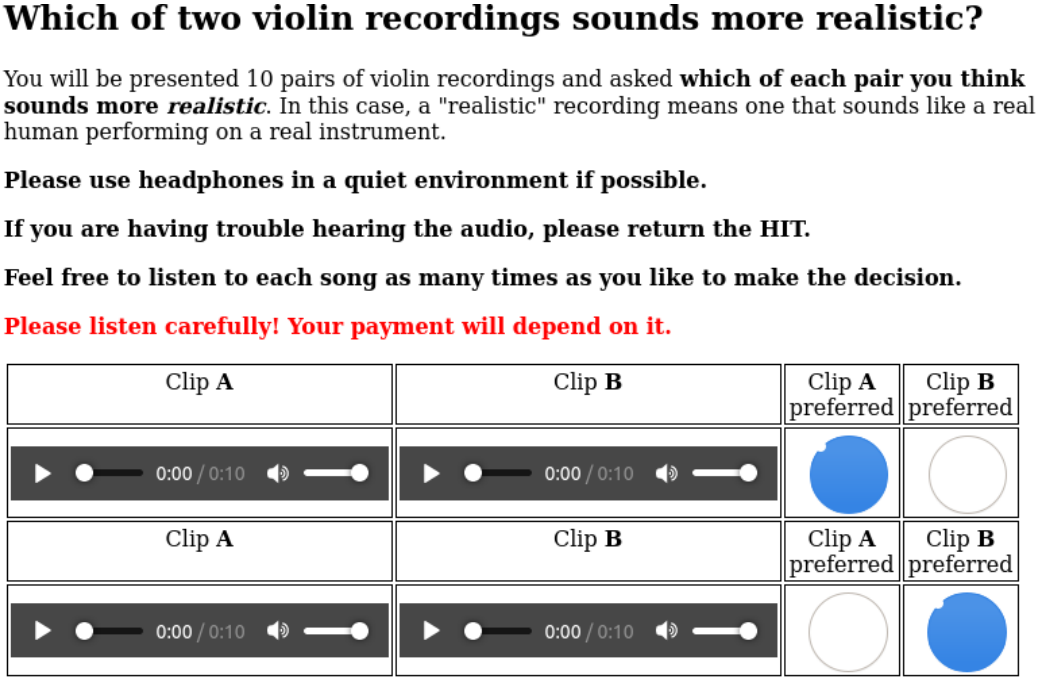

Figure 2: The web interface for our user study (eight pairs not shown).

D.1 Procedure

An annotator is presented with pairs of audio clips, each 10 seconds in length. Each pair contains the same section of MIDI synthesized by two different strategies, and the annotator's task is to report which of the two clips sounds more *realistic*, i.e., which sounds like a real human performing on a real instrument.

Each annotation task consists of 10 pairs of clips. Eight of these pairs consist of two synthesizers sampled without replacement from the union of our proposed methods and all baselines. Two of these pairs are used as control tasks and always consist of the human recording and the freely-available concatenative synthesizer (unrealistic). We exclude annotations from any annotator who "fails" these control tasks, i.e., marks that they find the synthesizer more realistic than the human recording.

For each method, we compute its *realism*: the proportion of pairs where a particular method was preferred above the other method in that pair. Note that, due to the random pairings, some methods are judged more often than others, though all are judged between 120 and 130 times.

We also collect self-assessed musical expertise from annotators. Of the annotators not filtered out by our control tasks, 54% indicated that they had *no* formal musical training, 23% reported that they had *some* musical training, and the remaining 23% reported that they had *substantial* musical training (more than five years of experience).

D.2 Discussion

The results of our study appear in Table 1. Interestingly, human recordings were not always considered to be the most realistic (they received a score of 82% as opposed to 100%). DDSP received a score of 59%, which can be thought of as a reasonable upper bound for the best score we could expect our MIDI2Params-based systems to achieve. This upper bound falls short of the commercial solution (61%); it's possible that DDSP would be able to exceed the realism of the commercial solution if we used higher quality recordings as source material. Both of our MIDI2Params models score below the upper bound (41% and 56% compared to 59%), but far exceed the realism of both a freely-available concatenative synthesizer (19%) and a heuristic alternative for converting MIDI into DDSP-style params (30%).

Interestingly, the MIDI2Params model which was trained on aligned MIDI scores performed substantially better than the MIDI2Params model trained on transcribed MIDI. More work may have to be done on transcription to improve the quality further and remove the need for scores as input to our system. However, because most people who may interact with our system will be creating datasets (by recording their performances), the requirement is not too large of an obstacle as users can simply perform pieces for which they also have scores.