DIFFERENTIABLE PIANO MODEL FOR MIDI-TO-AUDIO PERFORMANCE SYNTHESIS

Lenny Renault, Rémi Mignot and Axel Roebel

UMR 9912
STMS, IRCAM, Sorbonne Université, CNRS, Ministère de la Culture
Paris, France
renault@ircam.fr | mignot@ircam.fr | roebel@ircam.fr

ABSTRACT

Recent neural-based synthesis models have achieved impressive results for musical instrument sound generation. In particular, the *Differentiable Digital Signal Processing* (DDSP) framework enables the usage of spectral modeling analysis and synthesis techniques in fully differentiable architectures. Yet currently, it has only been used for modeling monophonic instruments. Leveraging the interpretability and modularity of this framework, the present work introduces a polyphonic differentiable model for piano sound synthesis, conditioned on *Musical Instrument Digital Interface* (MIDI) inputs. The model architecture is motivated by high-level acoustic modeling knowledge of the instrument which, in tandem with the sound structure priors inherent to the DDSP components, makes for a lightweight, interpretable and realistic sounding piano model. The proposed model has been evaluated in a listening test, demonstrating improved sound quality compared to a benchmark neural-based piano model, with significantly less parameters and even with reduced training data. The same listening test indicates that physical-modeling-based models still achieve better quality, but the differentiability of our lightened approach encourages its usage in other musical tasks dealing with polyphonic audio and symbolic data.

1. INTRODUCTION

The development of synthesizers and digital instruments has been prevalent in the evolution of music composition and production. Indeed, synthesizers led composers, musicians and sound designers to explore new sounds beyond those offered by acoustic instruments. Concurrently, the improvements made in instrument modeling have allowed for democratizing the usage of acoustic instruments to other musicians, by leveraging their sound in restricted contexts and with simpler controls. As such, instrument models need to be controllable to be usable in music creation workflows, while still being realistic by reproducing subtleties in the modeled instrument sounds.

Over the last several years, neural-based generative models have encountered success in producing convincing audio signals thanks to their expressiveness and the release of associated datasets. For monophonic and percussive sound synthesis, such models include autoregressive models [1], recurrent models [2], (variational) auto-encoders [3, 4], distillation model [5, 6] and *Generative Adversarial Network* (GAN) [7, 8, 9]. The more recently introduced DDSP method [10] modernized the spectral modeling paradigm [11] by implementing traditional synthesizers and digital signal processing operations as differentiable layers controlled by a *Deep Neural Network* (DNN). Those components are designed to exhibit characteristic properties of audio signals, such as periodicity and harmonicity. As these strong biases on the sound structure are introduced, the amount of training data and model parameters can be significantly reduced and the synthesis process is more interpretable. In the light of such results, other synthesizers and signal processing methods have been added to the set of DDSP components and were used in end-to-end neural models, such as wavetables [12], waveshapers [13] and infinite impulse response filters [14]. For better integration within music creation frameworks, there has been several improvements [15, 16, 17] to make the method compatible with the MIDI protocol. Polyphonic mixtures of audio signals can also be synthesized by combining multiple monophonic DDSP-based models [18, 19].

For the moment, all these previous methods have only been applied for monophonic instrument sound synthesis. While DDSP does not prevent its application for the polyphonic case, to the best of the authors' knowledge, no extension has been made to the DDSP components for handling simultaneous pitches on a single instrument. The piano, in particular, has been one of the most popular instruments through the history of western music, notably because of its versatility as a polyphonic instrument with a wide tessitura and dynamic range. The controls for playing it remain fairly simple despite its complexity, which inspired researchers for modeling it with different simulation systems for decades before the advent of deep generative models.

The contribution of this work if twofold. First, we propose an extension of a MIDI-compatible DDSP model to handle polyphonic input. Secondly, exploiting the modularity of the DDSP components, we conceive a model architecture that is motivated by high-level modeling knowledge, including several sub-models dealing with specificities of the piano sound, such as partials inharmonicity and beating. Trained on a publicly available dataset, the proposed approach of a fully differentiable piano synthesizer has been evaluated in a subjective evaluation. The results demonstrate that with significantly less parameters, and even with less training data, the proposed model achieves significantly better subjective evaluation than a state of the art neural-based model.

This paper is organized as followed. In section 2, different methods for piano sound synthesis are reviewed. The proposed model is then presented in section 3, while its training procedure is explained in section 4. Finally, model evaluation and comparison with other piano synthesis models are made in section 5. As a complement of this paper, audio samples and the source-code are provided online [1].

[1] https://github.com/lrenault/ddsp-piano

2. METHODS FOR PIANO SYNTHESIS

Piano sound synthesis can be achieved using different methods, with varying complexity, needs for data and overall quality.

The most common approach in the industry, which is also the most straightforward, is the concatenative synthesis, or sampling-based synthesis [20]. High-fidelity recordings of isolated notes are played back upon triggers from the MIDI input. The notes can be recorded at different velocities to cover the amplitude range, which can require significant memory storage, depending on the chosen resolution. While single notes can be perceived as realistic, mutual interactions between simultaneous notes (such a sympathetic resonances) cannot be reproduced with this approach and the user has very limited control over the piano model.

On the contrary, physical-modeling-based systems rely on explicitly modeling the sound generation and propagation processes in the physical instrument. These systems can achieve realistic, interpretable and controllable sound synthesis, but they require extensive modeling and precise measurements of physical components [21]. For practical usages, such approaches can be efficiently implemented with digital waveguides [22] or modal synthesis [23].

Signal-based methods can model the instrument by analysis and synthesis of audio examples with hand-crafted models. Underlying models include additive synthesis [24] and source-filter models [25]. These lightweight approaches are controllable and flexible as they can be directly applied to other instruments, but they often lack realism in the synthesis because of insufficient representation of physical details of the instrument or too simplistic controls.

Finally, data-driven neural-based systems train black-box models to synthesize audio from a large annotated dataset. Most of these models adapt successful text-to-speech techniques to the task of MIDI-to-audio synthesis for piano. The first works for this category of systems synthesize audio directly from the MIDI data using an auto-regressive WaveNet [1]. Others works make use of an acoustic model followed by a vocoder model to synthesize audio while usually predicting Mel-spectrograms as the intermediate audio representation [26, 27, 28]. The authors from [29] achieved better quality by predicting MIDI-filter-bank-based spectra instead: this time-frequency representation is a variant of the Mel-spectrogram where the filters for computing it from the *Short-Term Fourier Transform* (STFT) are centered around the MIDI note frequencies instead of the Mel frequencies. While differentiable, these neural-based systems require a significant amount of annotated recordings [1] and they do not explicitly model instrument properties. Controls on these systems are limited to the conditioning inputs provided during the model training.

3. PROPOSED APPROACH

The proposed synthesis model is a harmonic-plus-noise synthesizer [11] in a polyphonic context. It separately generates the inharmonic and noisy components $y^{additive}$ and y^{noise} of up to P simultaneous notes. The synthesized audio $\hat{y}$ is produced by summing all monophonic signals and by applying the estimated effect IR_i of the recording environment i:

$$\hat{y}(t) = (\mathrm{IR}_i * \sum_{p=1}^{P} (y_p^{additive} + y_p^{noise}))[t]. \tag{1}$$

The following sections detail the sub-modules composing the full model architecture, which is illustrated in Figure 1.

3.1. Inputs

The model is conditioned by all the controls a pianist has over its instrument, being the sequence of played notes, the pedals action and the recording environment.

By taking inspiration from the monophonic conditioning of DDSP [10, 17] and the polyphonic piano roll representation for piano synthesis and transcription [26, 30], we encode the played notes as a multi-channel sequence of active pitches and onset velocities $X(t)$:

$$X(t) = \{x_p^{pitch}(t), x_p^{vel}(t)\}_{p \leq P}. \tag{2}$$

For a monophonic channel p, the active pitch component $x_p^{pitch}(t)$ indicates the pitch (in the MIDI scale) of the note currently being active, taking the sustain pedal effect into consideration. A pitch of 0 implies that no note is played at time t. The force at which the note was played is given only at onset time in the onset velocity component $x_p^{vel}(t)$. The information for each note is contained within the same channel p in order for the model to more easily capture the monophonic string vibration. This encoding of the performance allows to disentangle sustained notes from repeated notes within an active sustain pedal, as in [26], while reducing the sparsity of regular piano rolls [31].

The una corda, sostenuto and sustain pedal signals are extracted from the MIDI 64, 66 and 67 *Control Change* (CC) channels at the same frame rate as the conditioning signal $X(t)$, and stacked into the pedals signal $\boldsymbol{x}^{ped}(t)$.

Finally, the piano model, the room reverberation and the microphones choice and placement are all entangled independently of the piano performance: each recording environment is provided as a one-hot encoding $i \in [1, I]$, for I different recording environments in the dataset.

3.2. Global model

One piano model can differ from another one because of its size and its tuning, which changes its inharmoncity profile over the tessitura [32] and its global detuning. We thus use an embedding layer, the *Z-Encoder*, to compute an embedding vector $\boldsymbol{z}_i$, an inharmonicity modifier b_i and an instrument specific detuning δf_i for each recording environment i.

Also, during a performance, the pedals activity and the interaction between simultaneous notes can change the timbre of an individual note [21]. This effect is modeled by a C-dimensional context signal $\boldsymbol{c}(t)$, which will be applied to all monophonic channels to influence the computation of the synthesizer controls. This context signal is computed by the *context network* $\mathcal{F}$, built upon a *Recurrent Neural Network* (RNN), from the piano embedding $\boldsymbol{z}_i$, the pedal signals $\boldsymbol{x}^{ped}(t)$ and the conditioning signal $X(t)$:

$$\boldsymbol{c}(t) = \mathcal{F}\{X(\tau), \boldsymbol{x}^{ped}(\tau), \boldsymbol{z}_i\}_{\tau \leq t}. \tag{3}$$

3.3. Monophonic string model

When a piano key is released, the damper mechanism is used to attenuate the string in order to stop it from vibrating. However, the total energy absorption is not instantaneous with this system, and higher notes do not even have dampers [21]. Therefore, the piano

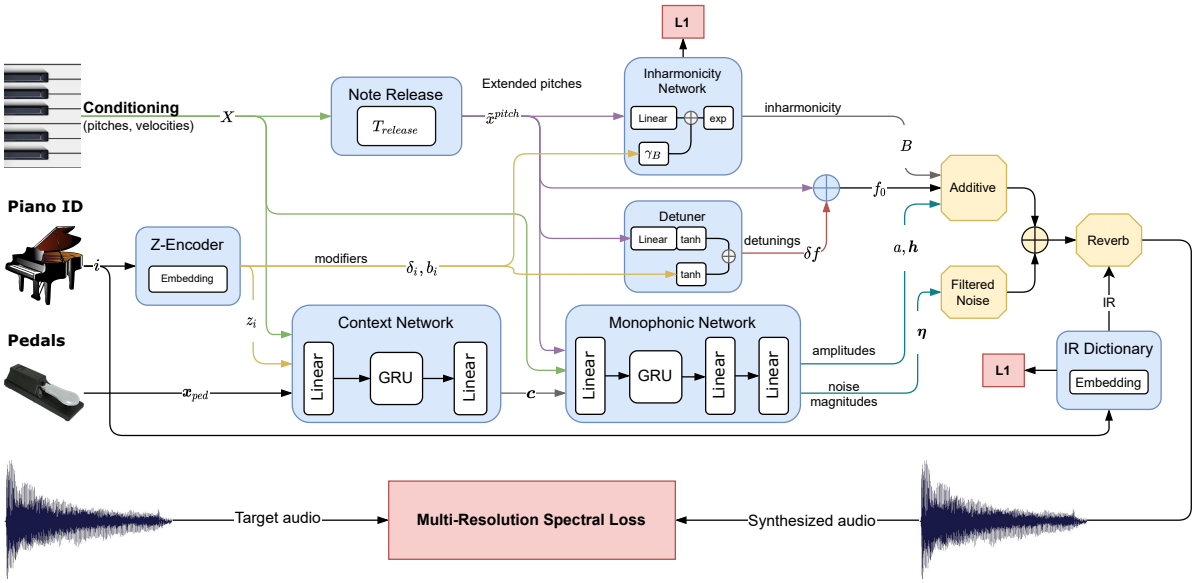

Figure 1: *Full architecture of the proposed piano sound synthesizer. The blue boxes represent the trained modules for the control of the synthesis. The synthesis modules from DDSP are represented by yellow boxes (*Additive, Filtered Noise*, and* Reverberation*). Finally, the* Multi-Resolution Spectral Loss *compares the input* target signal *(bottom left) and the output* synthesized sound *(bottom right).*

string still vibrates for a certain amount of time after the note offset. Taking inspiration from the *release* parameter of digital synthesizers, a *Note Release* module is used to generate an extended pitch signal $\tilde{x}^{pitch}(t)$ by prolonging the active pitch component of the conditioning signal $x^{pitch}(t)$ by a learned duration $T_{release}$. Note that the extended pitch conditioning signal $\tilde{x}^{pitch}(t)$ does not replace the original pitch conditioning $x^{pitch}(t)$ as we would lose the note offset information.

Furthermore, the stiffness of piano strings induces the partials of a piano note to not be pure harmonics of the fundamental frequency. Such characteristic is implemented with an explicit *inharmonicity model* over the piano tessitura, taken from [32]: the inharmoncity factor along the p-th channel $B_p(t)$ is computed from the extended pitch $\tilde{x}_p^{pitch}(t)$ and the instrument specific modifier b_i:

$$\begin{aligned} B_p(t) = &\exp\left(\alpha_T \tilde{x}_p^{pitch}(t) + \beta_T\right) \\ &+ \exp\left(\alpha_B \tilde{x}_p^{pitch}(t) + \beta_B + \gamma_B b_i\right), \end{aligned} \tag{4}$$

with $\{\alpha_T, \beta_T\}$ (resp. $\{\alpha_B, \beta_B\}$) the parameters of the linear asymptote in the treble (resp. bass) range. Treble asymptotes are the same for all pianos, according to [33], so b_i should only influence the bass asymptote, weighted by the parameter γ_B.

Another specificity of the piano sound is the duplication of higher strings to even out the loudness and the duration of notes across the whole tessitura. As the duplicated strings are slightly detuned from one another, partials beating can be perceived [34]. For n_{strings} sub-strings per note, a detuning factor δf of the fundamental frequency is computed by the *detuner* sub-model, by summing the global instrument-specific detuning δf_i with estimated detunings of the sub-strings by a time-distributed linear layer g_δ:

$$\boldsymbol{\delta f}_p(t) = \tanh(g_\delta(\tilde{x}_p^{pitch}(t)) + \tanh(\delta f_i). \tag{5}$$

The tanh activation function scales the detuning of each contribution to the semi-tone range, as in [17].

Finally, the spectral envelopes of notes and their evolution is predicted by the *monophonic network* $\mathcal{G}$. It is implemented as a causal RNN that computes the remaining synthesizers controls from the extended pitch $\tilde{x}^{pitch}(t)$, the conditioning vector $X(t)$ and the context vector $\boldsymbol{c}(t)$. As it is applied along each channel $p \leq P$, this recurrent network learns a monophonic string model to predict the notes amplitude $a(t)$, the energy distribution $\boldsymbol{h}(t)$ for K partials and noise filter magnitudes $\boldsymbol{\eta}(t)$:

$$a_p(t), \boldsymbol{h}_p(t), \boldsymbol{\eta}_p(t) = \mathcal{G}\{X_p(\tau), \tilde{x}_p^{pitch}(\tau), \boldsymbol{c}(\tau)\}_{\tau \leq t}. \tag{6}$$

3.4. Differentiable Synthesizers

The differentiable synthesizer layers convert the network controls into audio signals, in the spectral modeling paradigm [11]. Synthesizer controls are upsampled from frame rate to sample rate, with linear interpolation as in [10].

For a monophonic channel $p \leq P$, the *additive synthesizer* generates the (quasi-)harmonic audio component $y_p^{additive}(t)$ of the piano notes. It sums multiple sinusoids at frequencies computed from the extended pitch $\tilde{x}_p^{pitch}(t)$, inharmonicity $B_p(t)$ and detuning $\delta f_p(t)$ controls, and with amplitudes provided by the global amplitude $a_p(t)$ and harmonic distribution $\boldsymbol{h}_p(t)$:

$$y_p^{additive}(t) = \frac{a_p(t)}{n_{\text{strings}}} \sum_{n=1}^{n_{\text{strings}}} \sum_{k=1}^{K} h_{p,k}(t) \sin(\Phi_{p,n,k}(t)), \tag{7}$$

where $\Phi_{p,n,k}(t)$ is the instantaneous phase of the k-th partial given by:

$$\Phi_{p,n,k}(t) = 2\pi \sum_{\tau=0}^{t} f_{p,n,k}(\tau). \tag{8}$$

The inharmonic frequencies $\{f_{p,n,k}(t)\}_{k \leq K}$ are computed as in [32] from the fundamental frequency $f_{p,n,0}(t)$ and the inhar-

monicity coefficient $B_p(t)$:

$$f_{p,n,k}(t) = k f_{p,n,0}(t)\sqrt{1 + B_p(t)k^2}, \tag{9}$$

with the fundamental frequency $f_{p,n,0}$ deduced from the detuned pitch $\tilde{x}_p^{pitch} + \delta f_n$ with the MIDI note-to-frequency conversion formula:

$$f_{p,n,0}(t) = 440 \times 2^{\frac{1}{12}(\tilde{x}_p^{pitch}(t) + \delta f_n(t) - 69)}. \tag{10}$$

The *subtractive synthesizer* generates the residual noises that happen during a performance, mainly the hammer and key noise upon note onsets, the pedal noises and even the recording background noise. As in [10], a white noise $\boldsymbol{N}(t)$ is filtered in the frequency domain with the noise filter magnitudes $\boldsymbol{\eta}(t)$ computed by the model:

$$y_p^{noise}(t) = \mathrm{DFT}^{-1}(\boldsymbol{\eta}_p(t)\boldsymbol{N}(t)) \tag{11}$$

The room acoustics of the piano recordings is modeled by a differentiable convolutional reverberation. An impulse response IR_i is learned for each recording environment i and it is applied to the sum of audio signals from the bank of additive and subtractive synthesizers (equation 1).

4. EXPERIMENTAL SETUP

4.1. Dataset

The proposed model is trained and evaluated with performances from the MAESTRO v3.0.0 dataset [1]. This dataset contains almost 200 hours of professional piano performances spanning over $I = 10$ editions of the *International Piano-e-competition*. Pianists performed on Yamaha Disklaviers where MIDI data were recorded and aligned with the audio recordings. The ground-truth audio performances are downsampled to 16kHz and downmixed to mono, while the conditioning and pedals signals are extracted from the MIDI data with a frame rate of 250.

The model is trained with 3-second long segments, with a 50% overlap between two consecutive segments. Segments containing more simultaneous notes than the model polyphonic capacity P are removed from the training set.

4.2. Baseline systems

Our system is compared with other synthesis methods briefly presented in section 2. All samples synthesized with the following systems are also downsampled to 16kHz and converted to mono.

The open-source software `Fluidsynth` [2] is used as a sampling-based baseline system. The audio is synthesized by concatenating note recordings from the `YDP Grand Piano` [3] soundfont accordingly to the MIDI data.

As for the physical-modeling-based reference system, we chose the commercial software `Pianoteq 7`[4], which relies on modal synthesis [23]. Samples are generated using the default preset `NY Steinway D Classical`.

Finally, the *Text-to-Speech* (TTS)-inspired model from [29] is selected for the neural synthesis baseline. The `taco3-mfb-noi` variant exhibits the best quality according to their experiments. It is also trained on the MAESTRO dataset and it consists of a modified Tacotron-2 model [35] predicting MIDI-filter-bank-based spectra from active piano rolls. This time-frequency representation is then converted to audio with a simplified *Neural Source Filter* (NSF) model [36] with white noise as source.

[2] `https://www.fluidsynth.org/`

[3] `https://freepats.zenvoid.org/Piano/acoustic-grand-piano.html`

[4] `https://www.modartt.com/pianoteq`

Model	Parameters
TTS-baseline	31.4M
- Tacotron-2	30.6M
- NSF	736.3k
Default	521.5k
- Sub-models	281.5k
- Reverb	240k

Table 1: *Approximate number of trainable parameters for evaluated neural-based models and their sub-models.*

4.3. Model implementation details

Our proposed system is implemented with a polyphonic capacity of $P = 16$. The *Z-Encoder* outputs an embedding $\mathbf{z}_i$ of size 16 for each recording environment. The *context network* $\mathcal{F}$ is composed of a time-distributed dense layer of size 32 with leaky ReLU activation, followed by a causal *Gated Recurrent Unit* (GRU) layer of hidden size 64 and with layer normalization, then by a time-distributed linear layer outputting a context signal of size 32.

The *Note Release* module is initialized to extend the pitch conditioning signal by $T_{release} = 1$s. The *inharmonic model* is initialized with the parameters estimated in [32]: $\alpha_B^0 = -0.0847$, $\beta_B^0 = -5.82$, $\alpha_T^0 = 0.0926$ and $\beta_T^0 = -13.64$. We generate $n_{\text{strings}} = 2$ string signals per note, but their detuning are initialized to zero in the linear model g_δ of the *detuner*. The model-specific inharmonicity and detuning modifiers of the *Z-encoder* are also set first to zero, to learn a generic piano model during early training.

The *monophonic model* $\mathcal{G}$ input is processed by a 128-unit time-distributed dense layer with leaky ReLU activation, then by a 192-unit GRU layer and another dense layer of size 192 with leaky ReLU activation. Layer normalization is then applied before computing the note amplitude, $K = 96$ harmonic amplitudes and 64 noise filter coefficients with a linear layer.

Finally, the different reverb impulse responses are 1.5 seconds long at 16kHz (24k parameters for each recording environment), with the same random initialization as in [13] and the inference decay function from [17].

The total number of training parameters in this default configuration is given in Table 1, against *TTS*, the neural-based benchmark from Section 4.2.

4.4. Training

As with many recent neural-based audio synthesis methods [10, 13, 29], the model is trained to minimize the spectral difference between the target audio y and the synthesized audio $\hat{y}$ with a multi-resolution spectral loss. The component $\mathcal{L}_m$ of the spectral loss with resolution m compares the two audio signals by summing the L1 differences between both their spectrogram magnitudes and log

spectrograms, as in [10]:

$$\begin{aligned}\mathcal{L}_m(y,\hat{y}) = \quad & \||\mathrm{STFT}_m(y)| - |\mathrm{STFT}_m(\hat{y})|\|_1 \\ & + \|\log|\mathrm{STFT}_m(y)| - \log|\mathrm{STFT}_m(\hat{y})|\|_1,\end{aligned} \tag{12}$$

with $\|.\|_1$ the L1 norm and STFT_m the short-time Fourier transform with FFT size $m \in \{2048, 1024, 512, 256, 128, 64\}$.

In preliminary experiments, it has been observed that the reverb module tried to model the notes sustain and release behaviors, which resulted in abnormal reverberations and unrealistic unreverbed signals. As these behaviors should be learned by the *monophonic model* $\mathcal{G}$ instead, a L1 regularization loss $\mathcal{L}_{L1}$ is applied on the impulse response parameters.

Furthermore, the correct placement of partials in frequency is decisive for training stability, especially during early training. As partial frequencies in our system are deduced from explicit submodules, we propose a two-phase training procedure for separately optimizing the pure DNN components and the explicit sub-models.

During the first training phase, weights responsible for computing the partials frequencies are frozen to their initial values from Section 4.3, as they should be close to their optimal values: concerned weights are those from the *detuner*, the *inharmonicity model* and the model-specific detuning and inharmonicity modifiers of the *Z-encoder*. The other modules can then learn the notes spectral envelopes, residual noises and the reverb without displacing the note partials. The Adam algorithm [37] is used to optimize the model parameters, with a learning rate of 10^{-3} and a batch size of 6, with regard to the first loss function $\mathcal{L}_1$:

$$\mathcal{L}_1 = \sum_m \mathcal{L}_m(y,\hat{y}) + \lambda_{\mathrm{IR}}\mathcal{L}_{\mathrm{L1}}(\mathrm{IR}_i), \tag{13}$$

with λ_{IR} the balancing weight for the reverb regularization loss with regard to the spectral loss, here set to 0.01.

During the second training phase, the trainability of the model weights are reversed compared to the first training phase. In such manner, the system should match the learned partials frequency and beating to each piano specifically. For training stability purpose, a L1 regularization loss is applied on the *inharmonicity model* parameters deviation from their initial values. During this second training phase, the loss $\mathcal{L}_2$ can be expressed as:

$$\mathcal{L}_2 = \sum_m \mathcal{L}_m(y,\hat{y}) + \lambda_{\mathrm{B}} \sum_{\theta\in\{\alpha_B,\beta_B,\alpha_T,\beta_T\}} |\theta - \theta^0|, \tag{14}$$

with $\lambda_B = 0.1$ the weight on the *inharmonicity model* regularization loss with regard to the spectral loss. The weights of the *detuner*, *inharmonicity model* and *Z-Encoder* model-specific modifiers are fine-tuned by Adam with a learning rate of 10^{-5} and a batching size of 3.

The whole system can be optimized and fine-tuned by successively alternating between these two training phases. For our experiments, the system is trained with the first step for 2 full epochs on the training data until note partials are correctly generated by the additive synthesizer. It is then fine-tuned for 1 full epoch on the training data with the second training step. Finally, the first training step is applied again for further fine-tuning until the minimal validation loss value is reached. The full model training takes about 340k steps.

4.5. Ablation study

The relevance of our system sub-modules are evaluated by training and evaluating alternate versions of our approach. All following systems are trained with the procedure exposed previously in Section 4.4.

The *Deep Inharmonicity* variant replaces the explicit inharmonicity model from [32] by a DNN. As the deep *inharmonicity model* should ideally reproduce the equation 4, we use a *Multi-Layer Perceptron* (MLP) with sinusoidal activations as in [13]. The model takes the extended pitch conditioning and the model-specific inharmonicity modifier through 3 dense layers with sinusoidal activation and a hidden size of 8, then through a linear layer with ReLU activation for obtaining the note inharmonicity factor. The final output is scaled in order to keep the inharmoncity factor within a realistic range $B \in [0, 0.02]$.

The *Reduced-context* variant imitates sample-based synthesis by removing the conditioning input from the context vector computation. Since the synthesizer controls are computed on all monophonic channels independently, a monophonic note control would not have information on which other notes are also played, thus preventing mutual interaction between notes.

The *No Fine-tuning* variant is the default system without applying the fine-tuned values of the *inharmonicity model* and the *detuner*. This variant is an approximation of a model trained only with the first training phase of Section 4.4.

The DDSP components exploits sound structure priors that enables model training with low amount of data. Subsequently, we test the quality of our system on a simpler task but with reduced data by only modeling a single piano model. The *2009-only* system is trained by only keeping training and validation data from the year 2009 in the MAESTRO dataset, which amount for about 20 hours.

5. EVALUATION

The *default* configuration of our approach presented in Sections 3 and 4.3 is compared against its ablated variants from Section 4.5 and the other synthesis methods from Section 4.2.

5.1. Reconstruction quality

A first indicator of a model capacity to correctly reproduce its target data is the spectral loss value on unseen samples during training. The values of the spectral loss, as defined in Section 4.4, on the test recordings in the MAESTRO dataset are given in Table 2 for all evaluated systems.

	Spectral Loss	
Model	MAESTRO-all	MAESTRO-2009
Deep-Inharmonicity	5.89 (0.77)	5.93 (0.84)
Reduced-context	**5.49 (0.73)**	**5.53 (0.66)**
No Fine-Tuning	**5.48 (0.78)**	5.69 (0.83)
2009-only	-	5.61 (0.73)
Default	**5.48 (0.76)**	5.69 (0.83)

Table 2: Systems evaluation on the MAESTRO test set and the 2009 year test subset only. Measured by mean value of spectral loss difference with the original recordings (with standard deviation in parenthesis). Low loss value indicates better reconstruction quality.

For all variants, the loss values are higher on the 2009 subset than on the whole MAESTRO test set, which implies that this recording environment was one of the hardest to model. The system trained solely with data from the year 2009 obtains the same reconstruction quality as the *default* configuration on the 2009 subset. Therefore, if one wants to profile a single piano model, training on the full MAESTRO dataset is not necessary and smaller aligned datasets could be used instead [38].

The *Deep Inharmonicity* variant has higher loss values on both test sets than other variants relying on the explicit *inharmonicity model* from [32]: this explicit model proves to be beneficial for the system as it allows it to reproduce the spectral envelopes more faithfully with the additive synthesizer.

No statistical difference can be observed on the full test set by reducing the context. Reverb and the piano model embedding are enough to condition the global timbre in this classical repertoire. Hence, because of its reduced complexity, the *Reduced-context* variant converges faster than the *default* configuration and can achieve better reconstruction quality for the same number of training epochs, which can explain its better performances on the 2009 subset. Mutual interaction between notes (sympathetic resonances) is more exploited in contemporary music for instance: using such examples may help the system learning this specificity.

The approach also performs similarly without applying the fine-tuned inharmonicity and detuning parameters. Either the second training phase from Section 4.4 could not successfully fine-tune these parameters to their optimal values, or their initial values were already sufficient for allowing the whole system to converge.

In conclusion, one can train the proposed approach with a single piano model, with reduced context and by only applying the first training phase from Section 4.4, but keeping the explicit *inharmonicity model* is crucial for faster training while getting similar reconstruction quality.

5.2. Listening test

We conducted a listening test for gathering *Mean Opinion Score* (MOS) for all systems under evaluation. 11 performances were kept from the test data, covering all recording environments and with a diversity of composers, registers and note densities. The first 9 seconds of the performances were synthesized with all systems, which, with the real recordings, gives 99 audio samples to evaluate. Listeners were asked to rate their overall quality with a scale from 1 (very annoying) to 5 (real recording). In each trial, 2 samples from each of the 8 systems and 2 real recordings were randomly presented to the listener for rating. We gathered 52 participants that are musicians or audio professionals: 14 among them have notions of piano playing and 29 have been playing the instrument for several years. The results are reported in Figure 2.

The results from the previous reconstruction quality evaluation can be confirmed as the proposed system is rated similarly to its ablated versions, with the exception of the *Deep Inharmonicity* model having lower scores. Hence the quality improvement by using the explicit *inharmonicity model* is confirmed perceptually. For the tested samples, the natural beating between simultaneous notes in harmony may be sufficient for achieving realistic partials beating, and the mutual notes interaction is eventually negligible in front of the direct notes sound. Since single piano modeling is possible with our approach, it would be interesting to investigate the minimum amount of training data needed while preserving the perceived quality.

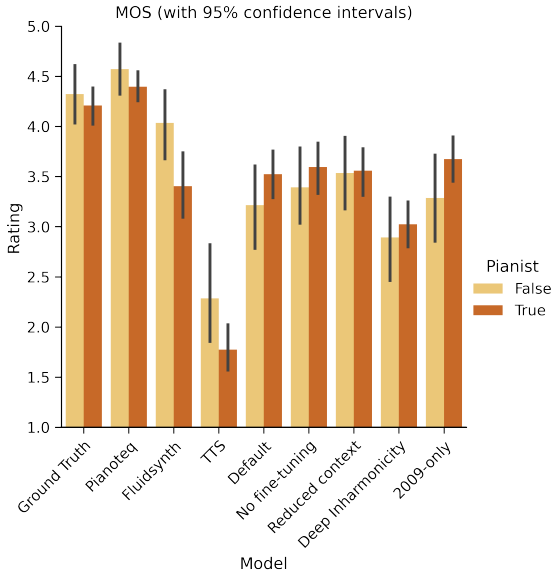

Figure 2: *MOS per system with 95% confidence intervals. Better sounding models have higher mean ratings. Scores are separated between non-pianists and people with notions of piano playing.*

All variants of our approach, including the model trained on a single piano, were preferred over the neural-based *TTS* benchmark. Even though the *TTS* baseline is more versatile for audio synthesis, as it was designed for speech synthesis, our approach is better suited for the task of piano sound synthesis: it achieves better sound quality with significantly less training parameters, as shown in Table 1.

The physical-modeling-based method has the best overall quality, even slightly better than the real recordings, as in [29]. The varying noise and recording quality in the real samples can be perceived as annoying compared to the clean synthesis offered by the `Pianoteq` software. As the quality of the training data is the upper bound limit, cleaner recordings can help the system achieving better synthesis quality. However, there is still improvements to be made on the system for actually reproducing the quality of the target data. As it stands, compared to the sampling-based system, the quality of our DDSP-based piano synthesizer is on par, according to the piano players, and slightly lower for the other listeners.

It is worth noticing that piano players are less convinced by the *Fluidsynth* concatenative synthesizer than non-piano players: they are more sensitive to an unrealistic feature in this synthesis approach, which can be the lack of interaction between notes, the constant use of the same samples for repeating notes or the lack of samples for correctly covering the velocity range.

6. CONCLUSIONS

This work presents an extension of the DDSP approaches to polyphonic instruments by designing a differentiable piano synthesizer motivated by high-level modeling knowledge. The approach relies, in particular, on an explicit model to account for the inharmonicity of the notes partials, which helps reproducing the spectral envelope of piano notes. In subjective evaluation, our model,

with significantly less parameters, obtains better synthesis quality than a state-of-the-art neural-based model. Its quality also does not decrease for the task of single piano modeling, which makes it usable for tasks with lower amount of training data. However, the proposed two-phase training does not improve the model performances compared to a simple training while fixing the explicit sub-models. Future works will investigate different approaches for training the explicit sub-models within this deep generative model.

The proposed approach does not outperform sampling-based and physical-modeling-based approaches, and quality can be improved by further integration of acoustic modeling knowledge for example. Still, thanks to its interpretability and its differentiability, our lightweight system can find its usage in other polyphonic music-related tasks, such as source separation [19] and self-supervised multi-pitch transcription [39].

7. ACKNOWLEDGMENTS

This work was supported by European Union´s Horizon 2020 research and innovation programme under grant number 951911 - AI4Media. We would like to thank Erica Cooper, among the authors from [29], for kindly sharing their test samples for our listening test. Part of this work was performed using HPC resources from GENCI-IDRIS (Grant 2022-AD011013202).

8. REFERENCES

[1] 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 *Proc. of the International Conference on Learning Representations (ICLR)*, 2019.

[2] Alexandre Défossez, Neil Zeghidour, Nicolas Usunier, Léon Bottou, and Francis Bach, "Sing: Symbol-to-instrument neural generator," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[3] 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 *Proc. of the International Conference on Machine Learning (ICML)*. PMLR, 2017, pp. 1068–1077.

[4] Philippe Esling, Axel Chemla-Romeu-Santos, and Adrien Bitton, "Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces.," in *Proc. of the International Society for Music Information Retrieval (ISMIR)*, 2018, pp. 175–181.

[5] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2021.

[6] Jinglin Liu, Chengxi Li, Yi Ren, Feiyang Chen, Peng Liu, and Zhou Zhao, "Diffsinger: Singing voice synthesis via shallow diffusion mechanism," *Association for the Advancement of Artificial Intelligence (AAAI)*, 2022.

[7] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts, "Gansynth: Adversarial neural audio synthesis," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2019.

[8] Javier Nistal, Stefan Lattner, and Gael Richard, "DrumGAN: Synthesis of drum sounds with timbral feature conditioning using generative adversarial networks," in *Proc. of the International Society for Music Information Retrieval (ISMIR)*, 2020.

[9] Antoine Lavault, Axel Roebel, and Matthieu Voiry, "StyleWaveGAN: Style-based synthesis of drum sounds with extensive controls using generative adversarial networks.," in *Proc. of the 19th Sound and Music Computing (SMC) Conference*, 2022, to appear, arXiv preprint arXiv:2204.00907.

[10] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts, "DDSP: Differentiable digital signal processing," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2020.

[11] Xavier Serra and Julius Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.

[12] Siyuan Shan, Lamtharn Hantrakul, Jitong Chen, Matt Avent, and David Trevelyan, "Differentiable wavetable synthesis," *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, to appear, arxiv preprint arxiv:2111.10003.

[13] Ben Hayes, Charalampos Saitis, and György Fazekas, "Neural waveshaping synthesis," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[14] Boris Kuznetsov, Julian D Parker, and Fabián Esqueda, "Differentiable IIR filters for machine learning applications," in *Proc. of the International Conference of Digital Audio Effects (eDAFx-20)*, 2020, pp. 297–303.

[15] Rodrigo Castellon, Chris Donahue, and Percy Liang, "Towards realistic MIDI instrument synthesizers," *NeurIPS Workshop on Machine Learning for Creativity and Design*, 2020.

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

[17] Yusong Wu, Ethan Manilow, Yi Deng, Rigel Jacob Swavely, Kyle Kastner, Tim Cooijmans, Aaron Courville, Anna Huang, and Jesse Engel, "MIDI-DDSP: Hierarchical modeling of music for detailed control," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2022, to appear, arXiv preprint arXiv:2112.09312.

[18] Masaya Kawamura, Tomohiko Nakamura, Daichi Kitamura, Hiroshi Saruwatari, Yu Takahashi, and Kazunobu Kondo, "Differentiable digital signal processing mixture model for synthesis parameter extraction from mixture of harmonic sounds," *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, to appear, arXiv preprint arXiv:2202.00200.

[19] Kilian Schulze-Forster, Clement SJ Doire, Gaël Richard, and Roland Badeau, "Unsupervised audio source separation using differentiable parametric source models," *arXiv preprint arXiv:2201.09592*, 2022.

[20] Diemo Schwarz, "Concatenative sound synthesis: The early years," *Journal of New Music Research*, vol. 35, no. 1, pp. 3–22, 2006.

[21] Juliette Chabassier, *Modélisation et simulation numérique d'un piano par modèles physiques*, Theses, Ecole Polytechnique X, Mar. 2012, Thèse sous la codirection de Antoine Chaigne, Unité de Mécanique, ENSTA ParisTech.

[22] Jukka Rauhala, Mikael Laurson, Vesa Välimäki, Heidi-Maria Lehtonen, and Vesa Norilo, "A parametric piano synthesizer," *Computer Music Journal*, vol. 32, no. 4, pp. 17–30, 2008.

[23] Balazs Bank and Juliette Chabassier, "Model-based digital pianos: From physics to sound synthesis," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 103–114, 2019.

[24] Julius O. Smith and Xavier Serra, "PARSHL: An analysis/synthesis program for non-harmonic sounds based on a sinusoidal representation," in *Proc. of the International Computer Music Conference (ICMC)*. International Computer Music Conference, 1987.

[25] Henrik Hahn and Axel Röbel, "Extended Source-Filter Model for Harmonic Instruments for Expressive Control of Sound Synthesis and Transformation," in *Proc. of the International Conference on Digital Audio Effects (DAFx)*, Maynooth, Ireland, Sept. 2013, p. 1.

[26] Jong Wook Kim, Rachel Bittner, Aparna Kumar, and Juan Pablo Bello, "Neural music synthesis for flexible timbre control," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 176–180.

[27] Hao-Wen Dong, Cong Zhou, Taylor Berg-Kirkpatrick, and Julian McAuley, "Deep performer: Score-to-audio music performance synthesis," *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, to appear, arXiv preprint arXiv:2202.06034.

[28] Yin-Jyun Luo Hao Hao Tan and Dorien Herremans, "Generative modelling for controllable audio synthesis of expressive piano performance," in *ICML Workshop on Machine Learning for Media Discovery Workshop (ML4MD)*, 2020.

[29] Erica Cooper, Xin Wang, and Junichi Yamagishi, "Text-to-Speech Synthesis Techniques for MIDI-to-Audio Synthesis," in *Proc. of the 11th ISCA Speech Synthesis Workshop (SSW 11)*, 2021, pp. 130–135.

[30] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck, "Onsets and frames: Dual-objective piano transcription," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[31] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan, "This time with feeling: Learning expressive musical performance," *Neural Computing and Applications*, vol. 32, no. 4, pp. 955–967, 2020.

[32] François Rigaud, Bertrand David, and Laurent Daudet, "A parametric model of piano tuning," in *Proc. of the International Conference on Digital Audio Effects (DAFx)*, September 2011, pp. 394–399.

[33] Robert W. Young, "Inharmonicity of plain wire piano strings," *The Journal of the Acoustical Society of America*, vol. 24, no. 3, pp. 267–273, 1952.

[34] Gabriel Weinreich, "Coupled piano strings," *The Journal of the Acoustical Society of America*, vol. 62, no. 6, pp. 1474–1484, 1977.

[35] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, Rif A. Saurous, Yannis Agiomvrgiannakis, and Yonghui Wu, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4779–4783.

[36] Xin Wang, Shinji Takaki, and Junichi Yamagishi, "Neural source-filter-based waveform model for statistical parametric speech synthesis," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5916–5920.

[37] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *Proc. of the International Conference for Learning Representations (ICLR)*, 2015.

[38] Valentin Emiya, Nancy Bertin, Bertrand David, and Roland Badeau, "MAPS - A piano database for multipitch estimation and automatic transcription of music," Research report, INRIA, July 2010.

[39] Jesse Engel, Rigel Swavely, Lamtharn Hanoi Hantrakul, Adam Roberts, and Curtis Hawthorne, "Self-supervised pitch detection by inverse audio synthesis," *ICML Workshop on Self-supervision in Audio and Speech*, 2020.