NEURAL SAMPLE-BASED PIANO SYNTHESIS

Riccardo Simionato

Dept. of Musicology
University of Oslo
Oslo, Norway
riccardo.simionato@imv.uio.no

Stefano Fasciani

Dept. of Musicology
University of Oslo
Oslo, Norway
stefano.fasciani@imv.uio.no

ABSTRACT

Piano sound emulation has been an active topic of research and development for several decades. Although comprehensive physics-based piano models have been proposed, sample-based piano emulation is still widely utilized for its computational efficiency and relative accuracy despite presenting significant memory storage requirements. This paper proposes a novel hybrid approach to sample-based piano synthesis aimed at improving the fidelity of sound emulation while reducing memory requirements for storing samples. A neural network-based model processes the sound recorded from a single example of piano key at a given velocity. The network is trained to learn the nonlinear relationship between the various velocities at which a piano key is pressed and the corresponding sound alterations. Results show that the method achieves high accuracy using a specific neural architecture that is computationally efficient, presenting few trainable parameters, and it requires memory only for one sample for each piano key.

1. INTRODUCTION

Physics-based sound synthesis methods can achieve remarkable realism in emulating the sound generation of acoustic instruments. However, the high-order nonlinear systems involved present a significant challenge for real-time computation. For this reason, even today, the vast majority of sound synthesizers for acoustic instruments utilize sample-based sound synthesis, also known as Pulse-Code Modulation (PCM) synthesis. For the piano, complete physical models that consist of the hammers, strings, soundboard, and the surrounding air have been proposed [1, 2]. Recent works have contributed knowledge on deeper aspects of piano sound generation, such as the coupling of transverse displacements [3], the origin of the so-called 'phantom partials' [4], and the radiation in the soundboard [5, 6]. With the recent advances in audio applications of machine learning, neural networks have been adopted for black-box modeling acoustic piano [7], where authors generate piano performance with an autoregressive model, conditioned by MIDI-based information, and more recently also for gray-box modeling [8, 9, 10], by using on the Differentiable Digital Signal Processing (DDSP) framework [11]. These approaches consider the sound signal to be synthesized as a combination of harmonic and noise components, similar to Spectral Modelling Synthesis [12] (SMS). The harmonic content is synthesized using a sum of sine waves, with parameters extrapolated directly from the target sound [8]. These piano models, which rely on machine learning, often entail a significant number of parameters, and their high computational complexity can limit their real-time applications. A similar approach, which focuses on synthesizing the sound of individual piano keys, enhances the model by integrating additional physics-based knowledge [9, 10]. This helps to reduce the required computations and takes into account the generation of the quasi-harmonic, percussive, and noise components, while an additional module enables the emulation of trichords. Comparable methods have been applied to other acoustic instruments, such as percussions [13] and guitars [14, 15].

The approach proposed in this paper further extends knowledge of how neural networks can be utilized for piano sound emulation, aiming to achieve better computational efficiency while maintaining the same level of emulation accuracy as state-of-the-art techniques. Our integration of neural modeling with sample-based synthesis enhances piano emulation and memory efficiency, as it enables the emulation of piano notes starting from just one example per key. Indeed, sample-based synthesis requires a trade-off between memory occupation and realism, as both are affected by the number of piano samples recorded and stored. In particular, a larger number of samples can cover more scenarios with respect to the stimuli that a player can impart to the piano keyboard and pedals. Ideally, covering all possible scenarios would offer perfect emulation, but this is not feasible in practice due to diverging memory requirements. When a specific stimulus for which sound needs to be reproduced is not associated with any recording, one or more processing techniques, such as interpolation, attenuation, and filtering, are used to cope with the lack of associated sound.

The rest of the paper is organized as follows. Section 2 details the methodology, including the neural architecture, dataset, training strategy, and the experiments we carried out to evaluate accuracy. Section 3 summarizes and discusses the results obtained, comparing the use of different sub-blocks within the neural architecture. Finally, Section 5 concludes the paper with a summary of our findings.

2. METHODOLOGY

The hybrid modeling approach we propose combines elements of sample-based synthesis and neural sound processing. In particular, the neural network is utilized to generate the sound of keys struck at any velocity, starting from a single recording at a given velocity. The neural network processes the audio samples one at a time, using the desired velocity as additional input information to condition the network's response. The neural network functions as a filter or an audio processing block, trained with input-output pairs that capture the dynamics of sound production. By doing so, our method allows a single recording of a piano key to be transformed

into a diverse set of nuanced sounds corresponding to different playing velocities.

We consider two training scenarios: one with a single network trained to cover all piano keys, and the other with separate networks trained for each individual piano key. The proposed model, shown in Figure 1, is based on a Selective State Space (S6) layer [16] inside a block such as in [17] and FiLM [18]-based network conditioning, which is placed just before the output layer [19]. We originally introduced this architecture for modeling virtual analog effects [17]. The bigger model presented in this work has 521 trainable parameters, thus utilizing an insignificant amount of memory compared to what is required for storing samples at different velocities.

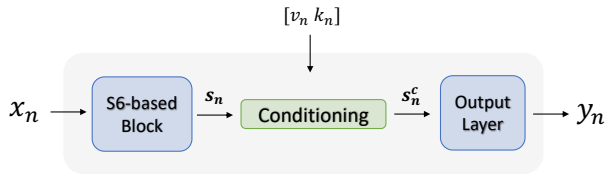

Figure 1: *High-level architecture of the proposed neural network, including the S6-based block, the FiLM-based conditioning block, and the output layer. x_n, y_n, v_n, and k_n are the input sample, output sample, velocity, and key value at time-step n.*

2.1. Architecture

The S6 layer is described by the following equations:

$$\begin{aligned} \boldsymbol{h_n} &= \boldsymbol{A}\boldsymbol{h_{n-1}} + \boldsymbol{B}(x_n)x_n \\ y_n &= \boldsymbol{C}(x_n)\boldsymbol{h_n} + \boldsymbol{D}x_n \end{aligned} \tag{1}$$

where x_n and y_n are the input and output samples at time n. $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ are real-valued matrices expressing linear mappings between x_n and $\boldsymbol{h_n}$ the states of the system. In our specific configuration, their dimensionalities are $(N \times N)$, $(N \times 1)$, $(1 \times N)$, and (1×1), respectively, where N represents the dimensionality of the state vector. The S6 layer is designed to capture long-range temporal dependencies within a sequence of inputs by using the state matrix $\boldsymbol{A}$. The matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ are dynamically computed based on the input x_n using a linear fully connected (FC) layer. This enables them to adapt to the different inputs. On the other hand, the coefficients of matrices $\boldsymbol{A}$ and $\boldsymbol{D}$ are learned during the training process and remain independent of the input, providing a stable foundation for encoding past system information. The S6 layer is integrated into a block structure illustrated in Figure 2, as in [17].

The S6-based block is structured as follows. The first FC layer contains twice the number of units as the length of the input channel dimension, thereby mapping its input into a higher-dimensional space. This projection is then split into two equal-sized vectors. The first vector is processed through a convolutional layer, followed by the Swish activation function [20], defined as

$$f(x) = x \cdot \text{sigmoid}(\beta x), \tag{2}$$

where β is a learnable parameter. This is followed by the S6 layer. The second vector is element-wise multiplied by the output of the S6 layer after it passes through the Swish activation function. The output, combined with the residual connection, is then fed into another FC layer that contains a number of units equal to the length of the block's input vector, which is followed by a Gaussian Error Linear Unit (GELU) [21] activation function

$$\text{GELU}\,(x) = x \cdot \frac{1}{2}\left[1 + \text{erf}(x/\sqrt{2})\right], \tag{3}$$

to produce the block's output.

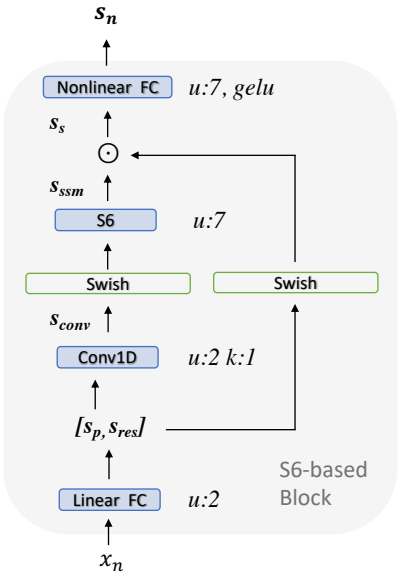

Figure 2: *Internal architecture of the S6-based block featured in the in architecture show in Figure 2. Details such as the number of units (u), activation functions, and kernel size (k) are provided next to the respective layers where they are applicable. S_p and S_{res} derived from the projection of the input signal x_n.*

2.2. Conditioning

The conditioning process consists of the FiLM method and the Gated Linear Unit (GLU), and it is illustrated in Figure 3. The FiLM method applies an affine transformation to a vector based on the conditioning value. Following the projection, a GLU determines the amount of information that should pass. The GLU employs a softsign activation function to regulate the flow of information.

The Conditioning block operates as follows: the velocity and key values v_n and k_n are first projected using a linear FC layer, which has a number of units equal to the double of the dimension of the output of the S6-based block s_n. The resulting vector is split into two coefficients, which are used to perform an affine transformation on s_n. After the affine transformation, the resulting vector is processed by another linear FC layer, which doubles the dimensionality. A softsign function is applied to one of the resulting coefficients, which is afterward multiplied by the other one to produce the output of the conditioning block.

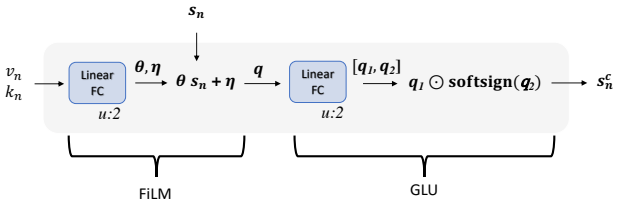

Figure 3: *Conditioning block consisting of FiLM and* softsign*-based GLU. The input velocity and key v_n and k_n are linearly transformed by an FC layer. The transformation results in an output vector with double the dimensionality of the output from the S6-based block, denoted as s_n. This vector is then utilized to apply an affine transformation to s_n. Subsequently, the transformed vector is processed through another fully connected layer, further doubling its dimensionality. Finally, a* softsign *function is applied to one of the resulting components, and this value is multiplied by the other component to yield the output of the conditioning block.*

2.3. Dataset

To evaluate the proposed approach, we have utilized recordings taken from the upright-closed and grand-piano from the BiVib dataset[1] [22]. The pianos used to collect the recordings are Yamaha Disklavier, which are controllable by MIDI messages. Pianos feature a high number of keys, spanning over seven octaves. The complex relationship between input velocity and the resulting sound, determined by the acoustic mechanism of the piano, exhibits significant differences between the low and high key registers.

We utilize all recordings at the ten different velocities available in the dataset, ranging from 12 to 111, values that correspond to the 7-bit MIDI velocity values. Recordings in the dataset are not associated with a specific velocity value but rather with a velocity range because small variations in velocity produce negligible changes in sound [22]. For our experiments, we have associated each recording with the lower bound of its respective velocity range, resulting in the following velocity values: $[12, 23, 34, 45, 56, 67, 78, 89, 100, 111]$. This resulted in a total of 880 recordings utilized in our experiments, each with a duration of 14 seconds, allowing the piano sound to fully decay after the key had been struck and held. The experiments were carried out with audio downsampled from 96 to 48 kHz, which is the rate at which our model generates audio samples.

To assess the effectiveness of our method, we conducted two different experimental trainings. First, we trained a set of key-specific models, choosing one key from each octave across different pitch classes: [$A0$, $B1$, $C2$, $D3$, $E4$, $F5$, $G6$, $A\#7$]. Next, we trained a single piano-specific model to predict arbitrary velocities for any of the 88 keys on the piano keyboard. This procedure was repeated for each dataset.

2.4. Experiments Setup

The models are trained using the Adam optimizer [23] with a gradient norm scaling of one [24] and the Mean Squared Error (MSE) as the loss function. Models were trained for up to 100 epochs with early termination if there was no reduction in validation loss for 50 epochs. The learning rate is reduced by 25% each epoch, starting from an initial learning rate set to $3 \cdot 10^{-4}$. The losses on the test set are computed using the model's weights that minimize the validation loss throughout the training epochs. The raw audio recordings are split into segments of 2048 samples and processed in batches of 8. The dataset is split into 90% for the training set and 10% for the validation set. Both piano and key-specific models were evaluated using a test set that included two specific velocities for the eight keys mentioned in the previous section, allowing for a fair comparison of the accuracy between both approaches.

When considering key-specific models, two different configurations have been investigated. The first, denoted as $T1$, involves training the neural network to generate sound at arbitrary velocities by processing a recording at the lowest velocity (12). This is evaluated on recordings at intermediate velocities 56 and 100. The other scenario, referred to as $T2$, evaluates the opposite situation, where the network processes recordings at the highest velocity (111). This is evaluated at intermediate velocities 56 and velocity at 23. These experiments allow us to evaluate the network's prediction ability on unseen velocities in the middle of the velocity range and near the other extreme of the range. When evaluating a single model intended to process recordings for all piano keys and produce sounds at arbitrary velocities, the assessment utilizes the configuration that yielded the best results in prior experiments with key-specific models, either $T1$ or $T2$. To assess the accuracy of the model, we computed the error-to-signal-ratio (ESR) and the multi-resolution STFT error defined as:

$$\frac{1}{N}\sum_m \left|\left| \frac{|STFT_m(\boldsymbol{y})| - |STFT_m(\hat{\boldsymbol{y}})|}{|STFT_m(\boldsymbol{y})|} \right|\right|_1 + \frac{1}{N}\sum_m ||\log(|STFT_m(\boldsymbol{y})|) - \log(|STFT_m(\hat{\boldsymbol{y}})|)||_1 \tag{4}$$

where $m = [256, 512, 1024]$.

To further assess the proposed neural architecture, we compared its accuracy in processing sound to match unseen velocities with against a long short-term memory (LSTM)-based model, which served as a baseline. Specifically, we replace the S6-based block with an LSTM layer with 8 units. The choice of units was made to match the number of trainable parameters of the proposed S6-based model. Audio examples, source code, and trained models described in this paper are available online [2].

2.5. Perceptual Evaluation

Perceptual evaluation is conducted using Mushra tests [25]. Specifically, we carried out listening tests to evaluate the unseen velocities at pitches [$A0, C2, D3, E4, F5, A\#7$] against the actual reference recording taken from the dataset. The test considers the specific-key models in both configurations ($T1$ and $T2$). The tests are conducted using the WebMUSHRA platform [25] and involved 20 participants with professional music expertise, including musicians and music producers. We utilize a standardized setup consisting of a laptop, a studio-grade audio interface, and high-end studio headphones. The test includes two sessions with twelve different listening examples each, conducted in a fixed order. In each example, the audio signal was limited to a duration of 7 seconds. We use a single anchor, a low-pass version of the reference with a cutoff frequency set to three times the fundamental frequency of the key. This approach deviates from the typical

[1] https://zenodo.org/records/2573232

[2] https://github.com/RiccardoVib/Neural-piano

cutoff frequencies used in MUSHRA tests, which generally utilize 7 kHz and 3.5 kHz low-pass filters. The adjustment ensures that the cutoff filter has an audible effect on test examples even at low pitches, which cannot be achieved with conventional fixed cutoff frequencies.

3. RESULTS

Table 1 summarizes the loss values for each model across scenarios $T1$ and $T2$. The proposed model demonstrates superior performance in both scenarios compared to the LSTM baseline. It exhibits lower MSE in scenario $T2$, although the STFT loss is marginally higher. Additionally, the proposed model shows less variance in results, offering more consistent performance across the octaves. When utilizing the grand piano dataset, the losses are generally higher, indicating a more challenging modeling task. Overall, the trend of the results is consistent with the upright dataset. In particular, the S6-based model is performing better than the LSTM-based model, and the $T2$ scenario results in a better configuration than the $T1$.

When evaluating performance, it's crucial to consider that the dataset consists of real recordings, which inherently include background noise. The relative impact of this noise is inconsistent across the dataset. At lower pitches, the piano's signal energy is higher than at higher registers, while the background noise level remains constant. Therefore, when presenting the test set results, we provide a rough estimate of the signal-to-noise ratio (SNR) for two velocity examples at each pitch. This estimate is based on the assumption that only noise is present in the last 5 seconds of each recording, while the rest of the duration contains both signal and background noise. However, this estimation approach has limitations, as the decay rate of notes in lower pitch registers is much slower than in higher ones. Consequently, for the two lowest pitches in the test set, a residual signal remains in the final 5 seconds of the recording, yet it fully decays within the segment. Additionally, velocity also influences the decay rate, albeit to a lesser extent, with lower velocities exhibiting faster decay.

Table 2 presents the losses of the proposed model for each key and training scenario, along with SNR estimates for the target examples in the test set. For both datasets, in the $T2$ scenario, the model excels in MSE and STFT losses but demonstrates poorer ESR performance compared to the $T1$ scenario. Further investigation indicates that these models struggle to accurately predict the tail part of each target recording, which is largely composed of background noise. However, matching sample-level noise is neither significant nor necessary for creating perceptually equal signals. In this final segment, a noticeable mismatch exists between the background noise level in the target and the softer noise predicted by the model, which was also suggested by listening tests. The model's apparent ability to reduce background noise levels can be seen as beneficial. Considering that SNR increases with velocity and decay rate is proportional to pitch, alongside the low relevance of prediction error over the noise segment, these aspects can enhance the interpretation and comparison of $T2$ versus $T1$ results. To facilitate comparison between different keys, a bar plot visualization of loss and metrics is provided in Figure 4. Models trained with the upright piano dataset display lower errors when examining the STFT and ESR metrics, particularly for lower keys. In contrast, models trained with the grand piano dataset find greater consistency across keys and present less variance in performances. Conversely, the MSE shows greater variation across both datasets due to the differing amplitudes among keys, making it a less intuitive indicator.

When considering a single model designed to process recordings for all 88 piano keys, the model's accuracy decreases as the complexity of the task increases, as evident in Table 3. In this case, the model performs better with the grand piano dataset. Figure 5 shows the results for the perceptual listening test, where the box plots display the median, lower, and upper quartiles, maximum and minimum values, and outliers of the responses. Of the 20 participants who participated in the listening tests, no one was removed in the post-screening because scoring the hidden reference below 90 in more than 15% of trial conditions. The plots reveal that the $T1$ and $T2$ configurations produce comparable perceptual outcomes across the entire range of velocities, with $T2$ demonstrating a marginally superior performance. The T-test yields a p-value of 0.013, indicating that this difference is statistically significant. Both scenarios exhibit a large variance in the ranking. Specifically, low-ranked examples are consistently associated with examples with higher pitches and lower velocities. In such cases, the faster decay rate reveals a longer segment of the evaluated example where the mismatch in background noise level from the target, as previously discussed, becomes more audible. Consequently, even a piano note with good sound quality might receive a lower rating when compared to a reference with higher background noise levels.

Additionally, from the listening test results, we specifically isolated the ratings of examples emulating only the velocity furthest away from the input recording, specifically 100 for the $T1$ case and 23 for the $T2$ case. When analyzing these extreme velocities, the results indicate that the $T2$ configuration is more perceptually accurate, providing a closer match to the reference. Here, the p-value is $1.71 \cdot 10^{-8}$, demonstrating statistical significance. This suggests that the $T2$ configuration is especially beneficial in these instances, offering enhanced performance compared to $T1$, particularly in emulating velocities that are most distant from the input recording. This scenario represents the most challenging situation for the models.

Figure 6 shows the spectrograms for the highest and lowest keys in the dataset, $A0$ and $A\#7$, and for the key in the middle, $D3$, for the $T2$ case. The spectrograms show a good match between predictions and targets. However, the sound generated by the model often exhibits more energy at high frequencies compared to the target, suggesting that the model does not sufficiently attenuate energy from the recordings at the highest velocity.

4. EFFICIENCY

The models proposed in this paper, namely the piano-specific and key-specific models, feature 521 and 505 trainable parameters, respectively—a relatively low number given the complexity of the task. These models require approximately 606 and 590 floating-point operations (FLOPs) per audio sample. Table 4 compares these models with a PCM synthesis approach, which utilizes separate recordings for each key and velocity. The PCM method represents a worst-case scenario in terms of memory requirements but offers the best possible accuracy. It's important to note that in PCM piano synthesis, multiple recordings are often stored for the same note-velocity pair to capture subtle variations that occur when repeatedly striking the same key with the same velocity—an aspect not considered in our current comparison.

To evaluate the memory requirements of our proposed mod-

Table 1: *Test set losses for the architectures considered in the experiments for each training scenario. Additionally, the table includes the error-to-signal-ratio (ESR) and STFT loss metrics. The reported errors are the mean and standard deviation calculated across all keys.*

Upright								
Model		MSE	ESR	STFT		MSE	ESR	STFT
Proposed	T1	$2.36 \cdot 10^{-4} (3.12 \cdot 10^{-4})$	$0.19 (0.21)$	$0.44 (0.30)$	T2	$4.83 \cdot 10^{-5} (2.23 \cdot 10^{-5})$	$0.22 (0.25)$	$0.47 (0.35)$
LSTM		$3.82 \cdot 10^{-4} (4.00 \cdot 10^{-4})$	$0.29 (0.26)$	$0.61 (0.41)$		$5.52 \cdot 10^{-5} (2.66 \cdot 10^{-5})$	$0.25 (0.25)$	$0.54 (0.35)$
Grand								
Model		MSE	ESR	STFT		MSE	ESR	STFT
Proposed	T1	$1.59 \cdot 10^{-3} (1.11 \cdot 10^{-3})$	$0.47 (0.29)$	$0.58 (0.41)$	T2	$6.07 \cdot 10^{-4} (6.38 \cdot 10^{-4})$	$1.06 (0.35)$	$1.06 (0.27)$
LSTM		$2.32 \cdot 10^{-3} (1.52 \cdot 10^{-3})$	$0.87 (0.10)$	$1.05 (0.04)$		$8.21 \cdot 10^{-4} (8.45 \cdot 10^{-4})$	$1.58 (0.70)$	$1.07 (0.55)$

Table 2: *Test set losses of the proposed model for each key in the test set under different training scenarios. The table includes ESR and STFT losses along with the estimated SNR in dB for two velocity examples at each pitch in the test set. In configuration $T1$, the test set velocities are 56 and 100, while in configuration $T2$, they are 56 and 23.*

Upright										
Key		∼SNR (dB)	MSE	ESR	STFT		∼SNR (dB)	MSE	ESR	STFT
$A0$	T1	$96 - 102$	$1.05 \cdot 10^{-3}$	0.19	0.96	T2	$96 - 87$	$6.50 \cdot 10^{-5}$	0.08	0.63
$B1$		$143 - 171$	$8.33 \cdot 10^{-5}$	0.02	0.37		$143 - 159$	$5.95 \cdot 10^{-5}$	0.05	0.52
$C2$		$192 - 202$	$3.29 \cdot 10^{-5}$	0.01	0.31		$192 - 164$	$2.26 \cdot 10^{-5}$	0.03	0.47
$D3$		$173 - 184$	$1.24 \cdot 10^{-4}$	0.03	0.42		$173 - 167$	$5.47 \cdot 10^{-5}$	0.05	0.42
$E4$		$177 - 187$	$1.66 \cdot 10^{-4}$	0.14	1.37		$177 - 167$	$4.64 \cdot 10^{-5}$	0.17	0.83
$F5$		$130 - 173$	$1.26 \cdot 10^{-4}$	0.18	1.60		$130 - 88$	$1.10 \cdot 10^{-5}$	0.15	0.94
$G6$		$119 - 158$	$1.59 \cdot 10^{-4}$	0.24	1.48		$119 - 81$	$4.07 \cdot 10^{-5}$	0.42	1.10
$A\#7$		$31 - 69$	$1.45 \cdot 10^{-4}$	0.70	1.67		$31 - 20$	$8.62 \cdot 10^{-5}$	0.83	1.91
Grand										
Key		∼SNR (dB)	MSE	ESR	STFT		∼SNR (dB)	MSE	ESR	STFT
$A0$	T1	$179 - 204$	$2.71 \cdot 10^{-3}$	0.79	1.86	T2	$179 - 122$	$6.79 \cdot 10^{-4}$	0.95	1.54
$B1$		$118 - 130$	$3.28 \cdot 10^{-3}$	0.49	1.05		$118 - 111$	$1.99 \cdot 10^{-3}$	1.45	1.40
$C2$		$172 - 205$	$2.29 \cdot 10^{-3}$	0.57	0.82		$172 - 163$	$1.22 \cdot 10^{-3}$	1.59	1.42
$D3$		$160 - 192$	$1.71 \cdot 10^{-3}$	0.71	1.73		$160 - 112$	$3.84 \cdot 10^{-4}$	0.97	1.45
$E4$		$162 - 208$	$1.11 \cdot 10^{-3}$	0.43	1.51		$162 - 111$	$2.18 \cdot 10^{-4}$	0.55	1.45
$F5$		$142 - 207$	$1.63 \cdot 10^{-3}$	0.78	2.28		$142 - 86$	$2.24 \cdot 10^{-4}$	0.84	1.72
$G6$		$111 - 183$	$1.46 \cdot 10^{-3}$	1.60	3.35		$111 - 7$	$1.10 \cdot 10^{-4}$	1.41	1.77
$A\#7$		$69 - 134$	$2.08 \cdot 10^{-3}$	0.93	1.97		$69 - 20$	$1.93 \cdot 10^{-5}$	0.74	2.24

els versus PCM synthesis, we consider audio samples encoded as 16-bit integers and trainable parameters represented as 32-bit floating-point numbers. For PCM synthesis, we consider the worst-case scenario where all pitch-velocity pairs are stored in memory. Therefore, accounting for 9 different velocity recordings for the 88 keys, each lasting 14 seconds, results in a total memory demand of $1,064$ gigabytes (GB).

In contrast, our proposed models significantly reduce memory usage. For the recordings associated with one velocity per key, only $118,272$ megabytes (MB) are required for the whole 88 keys of the piano. On the other hand, the proposed model requires storing trainable parameters. The piano-specific model utilizes $2,084$ bytes across all keys, while the key-specific model requires $2,020$ bytes per key, summing up to $177,760$ bytes in total. Overall, this approach enables substantial memory savings—approximately $946,173$ MB—when adopting either the piano-specific or key-specific model.

When considering polyphonic synthesis, which is typically used in piano synthesis, it's important to note that the proposed model is capable of providing only one voice. Therefore, multiple concurrent models are needed to achieve polyphony, making the overall computational requirements identical for both the piano-specific and key-specific approaches. However, given the low impact on memory requirements of the key-specific approach and its superior accuracy, we suggest this as the preferred method. Alternatively, a trade-off between the two approaches can be proposed, such as utilizing key-range-specific models. This is similar to PCM piano synthesis, where a specific sample is generally used over a given range of semitones. Lastly, while our focus is on small models, increasing the size of the neural network can enhance accuracy, though it comes at the cost of higher computational demands.

5. CONCLUSIONS

We presented a novel approach to piano emulation. A neural model is trained to learn how to process key recordings taken at a single velocity to emulate the sound of arbitrary velocities. Specifically, the neural network is trained on a collection of real piano recordings and acts as a filter that processes the available

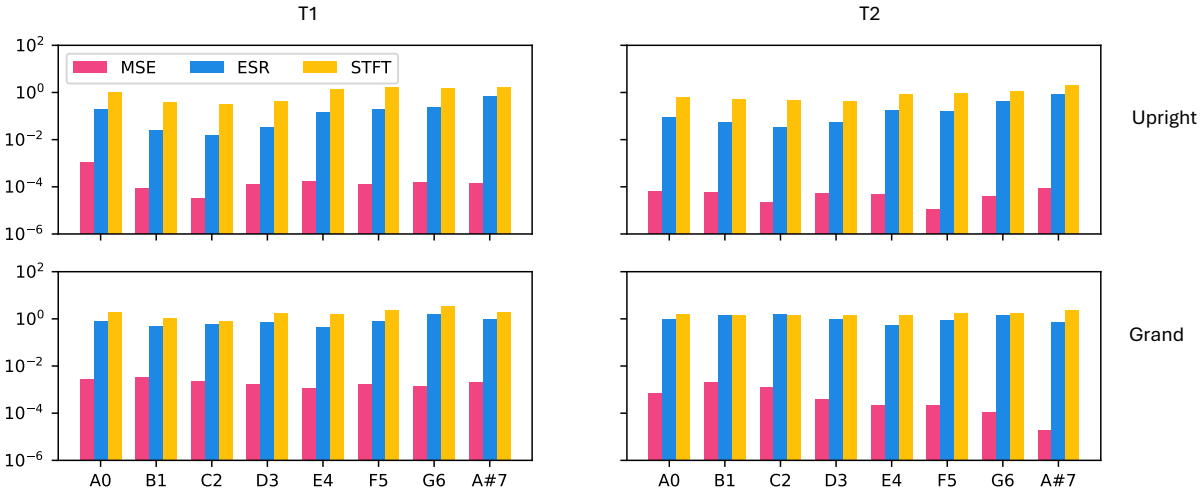

Figure 4: *Bar plots of the mean-squared-error (MSE), the error-to-signal-ratio (ESR), and STFT loss metrics from Table 1, for models trained with the the upright (top) and grand (bottom) piano dataset, and for the $T1$ (left) and $T2$ (right) configuration.*

Table 3: *Test set losses of the proposed model when considering all notes included in the datasets and each key in the test set. The table also reports the ESR and STFT loss.*

	Upright			Grand		
Key	MSE	ESR	STFT	MSE	ESR	STFT
$A0$	$1.02 \cdot 10^{-4}$	0.04	0.22	$2.36 \cdot 10^{-4}$	0.20	0.44
$B1$	$2.15 \cdot 10^{-3}$	1.36	1.20	$4.72 \cdot 10^{-3}$	2.16	1.02
$C2$	$3.17 \cdot 10^{-3}$	3.59	1.09	$3.97 \cdot 10^{-3}$	3.29	0.72
$D3$	$3.91 \cdot 10^{-3}$	2.01	0.63	$1.02 \cdot 10^{-3}$	1.56	0.80
$E4$	$5.55 \cdot 10^{-4}$	1.36	1.16	$3.81 \cdot 10^{-4}$	0.57	0.53
$F5$	$2.60 \cdot 10^{-4}$	1.14	1.29	$5.14 \cdot 10^{-4}$	1.09	0.79
$G6$	$7.03 \cdot 10^{-3}$	2.78	1.28	$4.03 \cdot 10^{-4}$	2.86	0.96
$A\#7$	$1.29 \cdot 10^{-4}$	1.62	1.69	$4.72 \cdot 10^{-5}$	1.30	2.13

Table 4: *FLOPs per sample, and memory requirements for the proposed models compared to PCM synthesis, where applicable. PCM synthesis uses 9 recordings for each key-velocity pair. Memory requirements for audio samples and model parameters are detailed separately. The requirements are based on a piano with 88 keys.*

Method	FLOPs per sample	Sample memory	Parameters memory	Total memory
PCM synthesis	n.a.	1,064448000 B	n.a.	1,064448000 B
Piano-specific	606	118,272000 B	2,084 B	118,274084 B
Key-specific	590	118,272000 B	177,760 B	118,449760 B

recordings to accurately emulate the sound associated with different stimuli. We focus on recreating the sound of individual keys at any velocity, which indicates the key press strength, starting from a single stored sample recorded at a single velocity.

We present two different scenarios: neural networks processing key-specific recordings utilizing velocity as an input to condition the sound processing, or a unique neural network for all the keys that utilize velocity and key number as input. In both cases, audio samples are processed one at a time, thereby achieving the lowest possible input-output latency. State-space models, particularly an S6 layer within a designed block, offer good emulation accuracy, especially when the network is used to generate the sound of lower velocities by processing the recording taken at the highest velocity. Both quantitative evaluation metrics and perceptual listening tests confirm this trend. On the other hand, the accuracy of this approach seems to diminish for keys in higher octaves because of the higher presence of background noise in the recordings. Utilizing recordings with higher SNR or performing specific noise reduction techniques may improve the performance. The paper focuses on small networks, with the expectation that increasing the network size will improve results, albeit at the cost of increased computational complexity. The method is evaluated using two datasets comprised of real recordings: one collected from an upright piano and the other from a grand piano. The key-specific models deliver better performance, likely due to the reduced problem complexity associated with modeling each key individually. Lastly, the paper discusses how the approach minimizes the number of recordings needed to be stored, and thus the

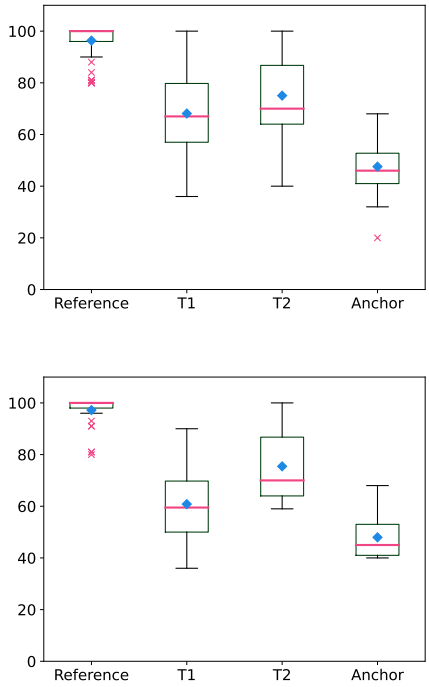

Figure 5: *Box plot illustrating the rating results from the listening tests (top) and the rating results when emulating only the velocity furthest away from the input recording (bottom). Magenta horizontal lines denote median values, and diamond markers represent the mean values. Outliers are displayed as crosses.*

associated memory requirements, by using a neural network.

The proposed method introduces a novel possibility for emulating individual piano notes without the need to store all audio samples and an approach that, in principle, can work with other acoustic musical instruments. This method can be integrated into the approach used in [10] to emulate trichords and other complex performance scenarios, such as re-stricken keys, arpeggios, and chords consisting of different numbers of notes and arbitrarily simultaneously played notes.

Although in our experiments, v_n is held constant throughout the duration of the played note, since the pressure applied to a key that is held down does not influence the sound generation of a piano, our proposed model theoretically allows v_n to be changed at every sample. This opens up possibilities for the design of piano synthesizers that incorporate features not found in their actual acoustic counterparts, such as modulating the synthesized sound by linking velocity to control parameters such as MIDI aftertouch, featured in high-end keyboards.

6. ACKNOWLEDGMENTS

We would like to acknowledge Stefano Zambon for his suggestions on this work.

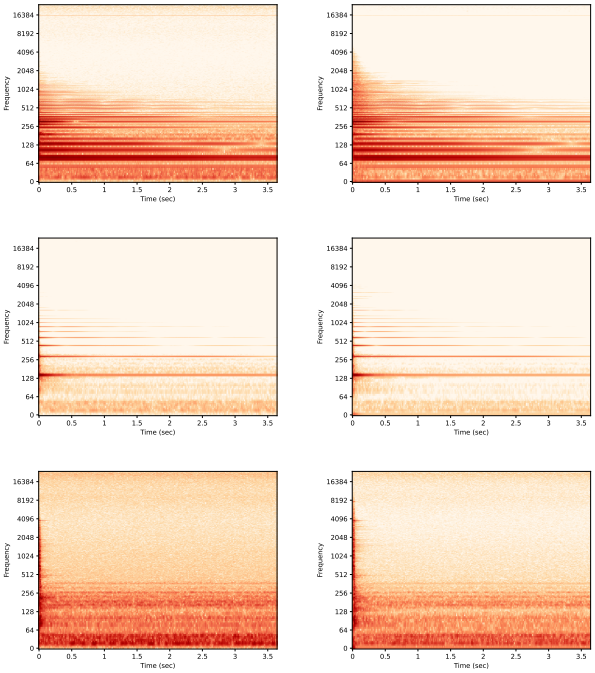

Figure 6: *Spectrograms of the $T2$ test set target (left) and prediction (right) for $A0$ (top), $D3$ (middle), and $A\#7$ (bottom) keys at velocity 34.*

7. REFERENCES

[1] N. Giordano and M. Jiang, "Physical modeling of the piano," *EURASIP Journal on Advances in Signal Processing*, vol. 2004, pp. 1–8, 2004.

[2] J. Chabassier, A. Chaigne, and P. Joly, "Modeling and simulation of a grand piano," *The Journal of the Acoustical Society of America*, vol. 134, no. 1, pp. 648–665, 2013.

[3] N. Etchenique, S. R. Collin, and T. R. Moore, "Coupling of transverse and longitudinal waves in piano strings," *J. Acoust. Soc. Am.*, vol. 137, no. 4, pp. 1766–1771, 2015.

[4] L. M. Neldner, "The origins of phantom partials in the piano," 2020.

[5] X. Boutillon and K. Ege, "Vibroacoustics of the piano soundboard: Reduced models, mobility synthesis, and acoustical radiation regime," *Journal of Sound and Vibration*, vol. 332, no. 18, pp. 4261–4279, 2013.

[6] B. Trévisan, K. Ege, and B. Laulagnet, "A modal approach to piano soundboard vibroacoustic behavior," *The Journal of the Acoustical Society of America*, vol. 141, no. 2, pp. 690–709, 2017.

[7] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C. A. Huang, S. Dieleman, E. Elsen *et al.*, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," *Int. Conf. on Learning Representations*, 2019.

[8] L. Renault, R. Mignot, and A. Roebel, "Differentiable piano model for MIDI-to-audio performance synthesis," in *Proc. of the Conf. on Digital Audio Effects (DAFx)*, 2022.

[9] R. Simionato, S. Fasciani, and S. Holm, "Physics-informed differentiable method for piano modeling," *Frontiers in Signal Processing*, vol. 3, p. 1276748, 2024.

[10] R. Simionato and S. Fasciani, "Sines, transient, noise neural modeling of piano notes," *Frontiers in Signal Processing*, vol. 4, p. 1494864, 2025.

[11] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," *Int. Conf. on Learning Representations*, 2020.

[12] X. Serra and J. 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.

[13] J. Shier, F. Caspe, A. Robertson, M. Sandler, C. Saitis, and A. McPherson, "Differentiable modelling of percussive audio with transient and spectral synthesis," *Forum Acusticum*, 2023.

[14] A. Wiggins and Y. Kim, "A differentiable acoustic guitar model for string-specific polyphonic synthesis," in *2023 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2023.

[15] N. Jonason, X. Wang, E. Cooper, L. Juvela, B. L. Sturm, and J. Yamagishi, "DDSP-based neural waveform synthesis of polyphonic guitar performance from string-wise MIDI input," *arXiv preprint arXiv:2309.07658*, 2023.

[16] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.

[17] R. Simionato and S. Fasciani, "Modeling time-variant responses of optical compressors with selective state space models," *Journal of Audio Engineering Society*, vol. 73, no. 3, p. 144–165, 2025.

[18] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[19] R. Simionato and S. Fasciani, "Conditioning methods for neural audio effects," in *Proceedings of the International Conference on Sound and Music Computing*, 2024.

[20] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[21] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2023.

[22] S. Papetti, F. Avanzini, F. Fontana *et al.*, "BIVIB: A multimodal piano sample library of binaural sounds and keyboard vibrations," in *International Conference on Digital Audio Effects*. Universidade de Aveiro, 2018, pp. DAFx–237.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Int. Conf. on Learning Representations*, 2014.

[24] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Int. Conf. on machine learning*, 2013.

[25] M. Schoeffler, S. Bartoschek, F. Stöter, M. Roess, S. Westphal, B. Edler, and J. Herre, "webMUSHRA—a comprehensive framework for web-based listening tests," *Journal of Open Research Software*, vol. 6, no. 1, p. 8, 2018.