G. A. Giudici, F. Caspe, L. Gabrielli, S. Squartini, and L. Turchet,
"Distilling DDSP: Exploring Real-Time Audio Generation on Embedded Systems,"
J. Audio Eng. Soc., vol. 73, no. 6, pp. 331–343 (2025 Jun.).
https://doi.org/10.17743/jaes.2022.0211.

Distilling DDSP: Exploring Real-Time Audio Generation on Embedded Systems

GREGORIO ANDREA GIUDICI,[1,*] **FRANCO CASPE,**[2] *AES Student Member*,
(gregorio.giudici@unitn.it) (f.s.caspe@qmul.ac.uk)
LEONARDO GABRIELLI,[3] *AES Member*, **STEFANO SQUARTINI,**[3] **AND**
(l.gabrielli@univpm.it) (s.squartini@univpm.it)
LUCA TURCHET,[1] *AES Member*
(luca.turchet@unitn.it)

[1]*Department of Information Engineering and Computer Science, University of Trento, Trento, Italy*
[2]*Centre for Digital Music, Queen Mary University of London, London, United Kingdom*
[3]*Department of Information Engineering, Università Politecnica delle Marche, Ancona, Italy*

This paper investigates the feasibility of running neural audio generative models on embedded systems, by comparing the performance of various models and evaluating their trade-offs in audio quality, inference speed, and memory usage. This work focuses on differentiable digital signal processing (DDSP) models, due to their hybrid architecture, which combines the efficiency and interoperability of traditional DSP with the flexibility of neural networks. In addition, the application of knowledge distillation (KD) is explored to improve the performance of smaller models. Two types of distillation strategies were implemented and evaluated: audio distillation and control distillation. These methods were applied to three foundation DDSP generative models that integrate Harmonic-plus-Noise, FM, and Wavetable synthesis. The results demonstrate the overall effectiveness of KD: the authors were able to train student models that are up to 100× smaller than their teacher counterparts while maintaining comparable performance and significantly improving inference speed and memory efficiency. However, cases where KD failed to improve or even degrade student performance have also been observed. The authors provide a critical reflection on the advantages and limitations of KD, exploring its application in diverse use cases and emphasizing the need for carefully tailored strategies to maximize its potential.

0 INTRODUCTION

In recent years, generative models have emerged as a cornerstone technology, providing the potential to create data with remarkable fidelity and diversity for various use cases. These models, e.g., variational autoencoders (VAE), generative adversarial networks (GAN), and diffusion models, have demonstrated unparalleled proficiency in generating images, text, and music that closely mirrors the distribution of their training data.

Within this landscape, neural audio synthesis leverages deep learning to synthesize and manipulate audio signals. Prominent examples include RAVE [1], a lightweight VAE model designed for real-time sound synthesis; ArchiSound [2] and DiffWave [3], two diffusion-based models known for their ability to refine noise into high-quality audio; and GANstrument [4], a GAN model able to interpolate between different sounds. In addition to these approaches, differentiable digital signal processing (DDSP) stands out for combining neural networks with traditional signal processing [5]. By offering intuitive control over musical parameters such as pitch, timbre, and loudness, DDSP is a versatile framework for tasks ranging from timbre transfer to voice synthesis [6].

Generative models have recently gained interest for embedded applications such as embedded systems, Internet of Things, and edge devices [7, 8], opening up exciting possibilities for personalization, interactivity, and reduced reliance on cloud-based services. For example, in real-time audio synthesis, embedded generative models can power devices such as Smart Musical Instruments [9] that can respond dynamically to the input of a musician. In this context, low latency and on-device processing is critical to ensuring a seamless and interactive user experience.

*To whom correspondence should be addressed, email: gregorio.giudici@unitn.it.

Deploying generative models on embedded systems presents significant challenges. High-quality models often require extensive computational resources, such as large memory footprints and high processing power, that exceed the capabilities of typical embedded hardware. Balancing the trade-off between model complexity and real-time performance is crucial to make generative models feasible in such constrained environments [10, 11]. To address these limitations, compression techniques like knowledge distillation (KD), quantization, and pruning have emerged as indispensable tools, offering a means to reduce the model's size while preserving performance. For instance, pruning and quantization have been successfully applied to real-time audio synthesis, including DDSP models [12–14]. In contrast, to the best of the authors' knowledge, KD has not been exhaustively explored for audio generation, particularly for real-time applications on embedded systems.

This paper takes a twofold approach. First, three DDSP models are analyzed based on foundation techniques for sound synthesis. This analysis evaluates the feasibility of implementing these models on real-time embedded systems while also comparing their performance with reduced versions (i.e., versions with fewer parameters). The promise of DDSP lies not only in its interpretability and controllability but also in its perceived efficiency. By leveraging DSP-inspired modules, DDSP models are often assumed to require fewer computational resources than purely neural counterparts, such as RAVE or DiffWave, especially for real-time applications. This notion has been echoed in works like [15] and [16], emphasizing DDSP's suitability for lightweight synthesis. However, this promise of efficiency has not been systematically validated, and it remains unclear whether DDSP models can consistently achieve efficient synthesis without compromising audio quality, particularly when scaled down for embedded systems.

Second, a framework for training audio generative models augmented by KD is presented. This approach explores two distinct types of distillation: audio distillation (AD), which emphasizes reproducing the teacher's audio quality, and control distillation (CD), which prioritizes capturing the structure of the teacher's generated control parameters. A thorough and informed analysis of the advantages and limitations of KD are provided, exploring its application in diverse use cases.

The motivations for the present study stem from the latency-quality trade-offs arising when deploying artificial intelligence (AI) models on single-board computers embedded in Smart Musical Instruments intended to foster novel kinds of real-time musician-AI interactions. These include real-time audio synthesis [9], collaborative improvisation [17], real-time detection of musical patterns to then control stage devices [18], or fast embedded retrieval of musical information for then querying online repositories [19]. In these contexts, it is crucial to reduce as much as possible the latency between an action of the performer and the response of the AI model while preserving the quality of the classified or generated content. This work addresses this by exploring the possibilities of deploying compressed neural synthesizers that perform comparably to their original counterparts.

1 BACKGROUND

1.1 Knowledge Distillation

KD was first defined and validated in [20], where it was shown to effectively train a smaller "student" neural network to mimic the behavior of a larger "teacher" network. This process leverages various forms of distilled knowledge directly from the teacher, rather than implicitly through an expanded dataset, enabling the student network to achieve performance comparable to or even exceeding that of the teacher.

Initially applied mostly for image classification [21–23], the use of KD has now extended to many other applications, such as speech recognition [24, 25] and text generation [26]. However, the application of this technique has not yet been exhaustively explored in the field of audio/music generation, particularly for real-time applications in embedded systems.

In the domain of audio synthesis, Oord et al. [27] introduced "Probability Density Distillation" to improve the efficiency of training the speech synthesis model WaveNet [28]. This technique distilled knowledge from a teacher model by transferring its probabilistic predictions to the student model. While effective for speech synthesis, the method was specifically designed for WaveNet and, additionally, was neither generalized to musical instrument synthesis nor analyzed for real-time performance or deployment on embedded systems.

Similarly, Nistal et al. [29] explored a GAN-based framework where a pretrained teacher model relabeled a categorical dataset with its predictions, effectively augmenting the dataset. A compact GAN model, acting as the student, was then trained using the teacher's predictions as targets. While this approach shares similarities with KD, it is closer to data augmentation than true distillation. Additionally, the method does not directly target real-time audio generation or embedded deployment.

In [30], KD techniques were applied to reduce the parameter count and inference time of Jukebox [31], a generative model for music composition. The authors employed distillation strategies to enhance the practicality of Jukebox for a faster generation. However, their focus was limited to reducing inference time for offline applications, without addressing real-time or embedded system constraints.

Despite the results reached in previous works, current studies on KD for audio and music generation are limited in scope. Most approaches, such as those discussed above, focus on specific architectures or offline tasks and fail to address the unique challenges of real-time generation in resource-constrained environments.

1.2 Differentiable Digital Signal Processing

DDSP paradigm merges traditional signal processing principles with machine learning. By reformulating classical synthesis techniques as differentiable operations, DDSP

enables models to learn directly from data while retaining interpretability and controllability, with applications spanning across music and speech synthesis. This section describes three fundamental synthesis techniques, i.e., Harmonic-plus-Noise (HpN), FM, and Wavetable synthesis, that have been adopted in the DDSP framework.

1.2.1 Harmonic-plus-Noise

HpN synthesis decomposes an audio signal into two complementary components: harmonic and noise. The harmonic component models periodic sounds as a sum of sinusoidal oscillators, while the noise component captures the nonperiodic, broadband content. A signal $x[n]$ is expressed as

$$x[n] = \underbrace{\sum_{k=1}^{N} A_k[n] \sin\left(2\pi f_k[n] nT + \phi_k[n]\right)}_{\text{Harmonic Component}} + \underbrace{e[n]}_{\text{Noise Component}}, \tag{1}$$

where T is the sampling period, N is the number of harmonics, and $A_k[n]$, $f_k[n]$, and $\phi_k[n]$ are respectively the amplitude, frequency, and phase of the kth harmonic. The noise component $e[n]$ can be modeled using subtractive synthesis:

$$e[n] = \mathcal{F}\big(\mathcal{N}[n]; \Theta\big), \tag{2}$$

where $\mathcal{N}[n]$ is an input noise (e.g., white noise or Gaussian noise), $\mathcal{F}$ is a filter function, and Θ are the parameters of the filter (e.g., cutoff frequency). In DDSP implementations, such as the one introduced in [5], a neural network predicts the overall amplitude of the audio signal, the normalized distribution of spectral variations among the various harmonics, and the coefficients of a LTV-FIR[1] filter used to model the noise component.

1.2.2 Linear FM

FM synthesis generates complex timbres by modulating the frequency of a carrier oscillator using another modulator oscillator. A signal $x[n]$ is expressed as

$$x[n] = A_c \sin\left(2\pi f_c nT + I \sin\left(2\pi f_m nT\right)\right), \tag{3}$$

where A_c and f_c are respectively the amplitude and frequency of the carrier, f_m is the modulator frequency, and I is the modulation index that determines the spectral complexity. FM synthesis gained prominence in the 1980s with Yamaha's DX7 synthesizer. It became a standard for generating bright, dynamic timbres in electric piano, bass, brass, and bell sounds. The DDX7 model [15] emulates DX7's FM synthesis with six sinusoidal oscillators, arranged in interconnected patches to simulate instrument-specific timbres (see Fig. 1). A convolutional decoder generates time-varying envelopes that control the amplitudes (A_c) and modulation indices (I) of the oscillators, while modulation (f_m) or carrier (f_c) frequencies are derived directly from the fundamental frequency passed as input to the model, according to the frequency ratios in the patches.

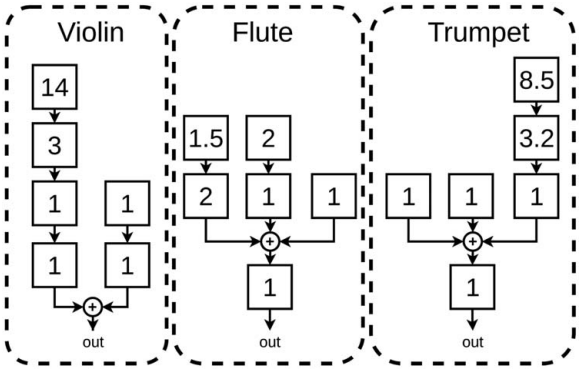

Fig. 1. Examples of FM patches used in the DDX7 [15]. Squares indicate sinusoidal oscillators and their frequency ratios, i.e., the ratio between the modulating frequency (f_m) and the carrier frequency (f_c).

1.2.3 Wavetable

Wavetable synthesis creates sound by cycling through predefined waveforms stored in memory. These waveforms are stored in tables, called wavetables, which capture the spectral characteristics of a sound.

An interpolation function $\mho(w; \phi[n])$ can be used to retrieve an arbitrary value of the stored waveform. For example, linear interpolation is defined as

$$\mho(w; \phi[n]) = w[i] + \alpha \cdot (w[i+1] - w[i]), \tag{4}$$

where the index i is calculated from the phase $\phi[n]$ as the integer part, $w[i]$ and $w[i+1]$ are consecutive values stored in the wavetable, and α is the interpolation factor that blends between these waveform values, typically equal to the fractional part of the phase. First introduced in the 1980s by PPG and later popularized by synthesizers like Waldorf's Wave, wavetable synthesis offers computational efficiency and rich timbral variety. By interpolating between multiple wavetables, this synthesis technique enables dynamic timbre transitions and richer generation. Based on this approach, a signal $x[n]$ can be expressed as a sum of multiple wavetable oscillators:

$$x[n] = \sum_{k=1}^{M} A_k[n] \mho(w_k; \phi_k[n]), \tag{5}$$

where $A_k[n]$ is the amplitude of the kth wavetable oscillator and M is the number of wavetables. Wavetable synthesis is a staple in digital synthesizers thanks to its dynamic sound design capabilities, versatility, and ease of use. The authors in [32] implemented a DDSP model that replaces the harmonic synthesizer with a computationally more efficient wavetable version. In addition, the values stored in the tables are learned as well during the training. While the model learns how to predict the amplitudes $A_k[n]$ and the waveform values of the wavetables, the phase accumulator ($\phi[n]$) is governed by the input pitch. This limits

[1] LTV-FIR filter refers to a combination of two concepts: linear time-varying (LTV) systems and finite impulse response (FIR) filters.

the model's ability to control phase dynamics explicitly but ensures computational efficiency and interoperability.

2 METHODOLOGY

This section describes the methodology employed to investigate the feasibility of deploying DDSP models for real-time audio synthesis on embedded systems. The selection and configuration of models, training procedures, KD schemes, and evaluation metrics are detailed. The code to implement and train all the models, both with and without the distillation schemes, is available online.[2] Additional material, such as audio examples and figures, is available at the accompanying page.[3]

2.1 Models

DDSP models are highly adaptable: The number of potential models is vast since each can be tailored to emulate different DSP techniques. All models in this work share the same DDSP decoder architecture [5], which features a decoder model conditioned on pitch and loudness extracted from input audio, with a differentiable synthesizer. A learnable reverb module, with a fixed duration of 1 s, is included as well to simulate acoustic effects of the input audio. The decoder is trained to predict the control parameters used by the synthesizer to emulate the input audio based on the pitch and loudness features.

Three foundational DDSP models, each representing a distinct approach to sound synthesis, are focused on:

1. HpN: As introduced in the original DDSP paper [5], the decoder is formed of fully connected and recurrent layers. For the differentiable synthesizer, the number of harmonics chosen for the sinusoidal synthesizer is 101, while the subtractive filter has 65 coefficients.
2. DDX7: Presented by Caspe et al. [15], this model features a convolutional-based decoder that generates envelope signals controlling the volume and modulation index of six oscillators interconnected in instrument-specific patches. The decoder contains various residual blocks, formed of temporal convolutional networks (i.e., one-dimensional convolutional networks with exponentially growing dilations that efficiently model long sequences).
3. Wavetable: This model is based on the work of Shan et al. [32]. It shares the same decoder architecture of the HpN model but replaces the harmonic synthesizer with a wavetable synthesizer comprising 20 wavetables: four fixed to fundamental harmonics and 16 for which the values are learned during the training phase. Each wavetable is 512 samples long. Instead of the harmonic distribution, the decoder predicts time-varying attention weights for mixing between the wavetables.

[2] https://github.com/gregogiudici/distilling-ddsp.
[3] https://gregogiudici.github.io/distilling-ddsp/.

Table 1. Number of trainable parameters for each DDSP model. The values comprise all learnable parameters.

Model	Full	Reduced
Harmonic-plus-Noise	4.3 million	23,500
Wavetable	4.3 million	32,400
DDX7	415,000	67,900

Each model was implemented in a "full" version, preserving the original architecture of the reference papers, and a "reduced" version, with fewer decoder parameters. The latter requires a lower computing power for inference, which makes it more suitable for embedded systems. The size of the various models in each version is presented in Table 1. Building on the initial comparison in the original DDSP paper [5], where a smaller version of the decoder of approximately 250,000 parameters was briefly presented, this work further pushes the compression to better satisfy resource constraints of embedded systems:

- For the HpN and Wavetable models, the size was reduced by two orders of magnitude (approximately 100× smaller), by choosing a value of 16 for the number of hidden states in the recurrent layers and neurons in the fully connected layers, significantly smaller than the original value of 512.
- For the DDX7, which was already significantly smaller, a one-order magnitude reduction was applied (approximately 10× smaller), by reducing the number of blocks that form the decoder from five to three and the number of hidden channels in each convolutional layer from 128 to 64.

2.2 Data and Training

All models were trained using four distinct musical instrument datasets: flute, trumpet, violin, and piano. These were chosen to investigate the model performances across a diverse set of timbres and signal features. The flute, trumpet, and violin samples were sourced from the University of Rochester Music Performance dataset [33], and the piano samples were drawn from the Jazznet dataset [34]. Between these datasets, there are considerable differences in terms of reverberation. Thus, separate models were trained for each of these datasets to mitigate the effect of varying reverberation on perceived audio quality. Additionally, it was ensured that the reverb module's size remained consistent between the full and reduced versions, minimizing its impact on the evaluation of model performance.

Preprocessing steps, following procedures in the reference papers [5, 15, 32], included resampling all audio samples to 16 kHz, detecting and removing silent regions, and segmenting the audio into 4-s chunks. Feature extraction involved the CREPE algorithm [35] for pitch estimation, and the computation of the A-weighted loudness as described in [36]. The hop size was fixed at 64 samples, resulting in pitch and loudness sequences of 1000 samples per 4-s segment, corresponding to a temporal resolution of 250 Hz.

Each dataset was divided into training, validation, and test sets with respective splits of 80%, 10%, and 10%.

All models were trained for 120,000 steps using a batch size of 16. The AdamW optimizer was employed with a weight decay of 0.01 and an initial learning rate of 0.001, which decayed by a factor of 0.98 every 10,000 steps. Gradient clipping was applied with a limit of three to prevent exploding gradients. Similarly to the reference paper on DDSP [5], the Multi-Scale Spectral loss was employed as reconstruction loss, with window lengths [2,048; 1,024; 512; 256; 128; 64] and overlap of 75% between windows.

2.3 Distillation Scheme

The KD scheme is illustrated in Fig. 2: Both the teacher and student models receive the same pitch and loudness sequences from which they predict the control parameters used by the differentiable synthesizer to generate the audio signal. The framework leverages two complementary losses, L_R and L_D, which are combined linearly to form the total loss $\mathcal{L}$ used to update the student's parameters:

$$\mathcal{L} = \alpha_R L_R + \alpha_D L_D. \tag{6}$$

Here, L_R represents the reconstruction loss, responsible for training the student model to recreate the characteristics of the target audio. In contrast, L_D is the distillation loss that guides the student's learning by prioritizing specific intrinsic features that the teacher model should capture better than the student. The weights α_D and α_R balance the contributions of the two losses.

The present KD framework incorporates two distinct distillation approaches:

1. AD: Matching the spectro-temporal characteristics of the teacher's output audio using audio-based loss functions. This distillation technique uses the teacher to prioritize specific audio features that the teacher should have learned to grasp more effectively than the student, thus guiding the student's learning.
2. CD: Minimizing the difference between the control parameters predicted by the teacher and the student (e.g., amplitudes). By shifting the focus from a one-dimensional domain such as audio to a multidimensional one, the student can better understand the intrinsic correlations between the various control parameters.

This second approach is inspired by the work presented in [37], which combines audio-based and control-based losses to improve the performance of the DDSP model. Unlike their work, which requires synthesizer-generated datasets, the present method distills teacher-generated controls, thus generalizing this method to any arbitrary audio datasets.

For all experiments, the Multi-Scale Spectral loss was employed for both reconstruction loss and AD, while mean absolute error was used for CD. The value of α_R was set to 0.5 for both distillation techniques, while α_D was set to 0.5 for AD and 1.0 for CD, due to the differing magnitudes of the loss values.

The framework has been designed with scalability and simplicity in mind. It is freely available online,[2] to encourage other researchers to expand its functionality starting from this work.

In the first series of experiments, the potential of KD was evaluated within the same DDSP model architecture, with the student being the reduced version and using the full version as the teacher. This setup provided a controlled environment to isolate and analyze the impact of distillation techniques on each specific DDSP model.

In addition, a series of specialized experiments was conducted:

- CD (Pretrained W.Table): Instead of retraining the student's wavetables from scratch, they were initialized using the pretrained wavetables of the teacher. The training process then focused solely on CD, teaching the student how best to utilize these pretrained wavetables without applying reconstruction loss.

For all experiments in this study, a reduced version of a DDSP model was consistently used as the student, and a full version of a DDSP model was consistently used as the teacher. Consequently, the terms "reduced" and "student," as well as "full" and "teacher," will be used interchangeably throughout the remainder of this paper.

2.4 Evaluation

To assess the performance of the proposed models, evaluation was conducted in two primary domains: *audio quality* and *real-time embeddability*.

2.4.1 Audio Quality Evaluation

Two metrics were employed to evaluate audio quality: Fréchet Audio Distance (FAD) and subjective listening tests. FAD is a widely used objective metric for assessing the quality and diversity of generated audio by measuring the statistical similarity between embeddings of generated and reference audio samples [38]. The embedding model chosen for FAD calculation significantly influences the results, as highlighted in [39]. Thus, FAD was computed using two distinct embedding models: VGGish [40] and EnCodec [41]. VGGish, trained on a large-scale audio dataset, is commonly used in DDSP-related research [15, 42] and captures broad audio features. EnCodec, on the other hand, was selected for its higher sensitivity to acoustic variations [39], making it complementary to VGGish and offering a more nuanced evaluation of model differences. The FAD was computed using the `frechet-audio-distance` library,[4] and the calculation was performed over the entire corpus, including the training set.

Subjective evaluation was performed using a web-MUSHRA [43] listening test, designed to assess differences in perceived audio quality between models. The test involved 18 participants (aged between 24 and 71 years, mean

[4] https://github.com/gudgud96/frechet-audio-distance.

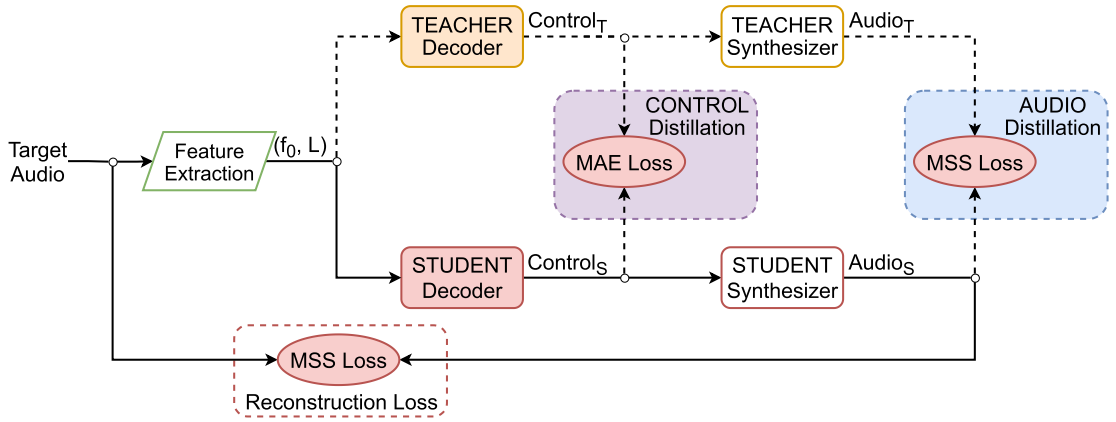

Fig. 2. Overview of the KD scheme applied in this work. The teacher and student models receive the same pitch (f_0) and loudness (L) sequences to predict the control parameters (Control$_T$ and Control$_S$) that will be used to synthesize the audio signals (Audio$_T$ and Audio$_S$). Components highlighted in red are involved in training the smaller model (i.e., for computing the loss used for updating the student's parameters). The pretrained teacher model with frozen parameters is highlighted in orange. The reconstruction loss ensures the student model's audio output resembles also the target audio. Solid lines indicate workflow elements present in all experiments (typical of the standard training of a DDSP model), while dotted lines represent workflows specific to the distillation process. In this paper, the authors have alternately used AD and CD, although the framework allows several distillations to be combined simultaneously. MAE = mean absolute error; MMS = Multi-Scale Spectral.

age = 33 years, standard deviation = 10 years). All participants reported normal hearing conditions. Nine participants were professional musicians (eight having a Conservatory diploma), while the others included audio researchers, music producers, and amateur musicians.

Participants were asked to rate the quality of audio generated by each model on a scale of 0 to 100, compared to a hidden reference and an intentionally degraded anchor. Results were averaged to obtain the mean opinion score (MOS), a standard metric for subjective evaluation. The anchor was generated using linear predictive coding, a widely used method for analyzing and synthesizing speech and other audio signals [44]. Linear predictive coding corresponds to decomposing an audio signal into two components: the excitation signal (source) and the filter (vocal tract or instrument resonance). In this setup, the filter was an all-pole filter with ten coefficients, and the excitation signal was an impulse train. To ensure the reliability of results, the responses from two participants were excluded from the statistical analysis for rating the hidden reference below 75 in more than two trials. Audio examples are available at the accompanying page.[3]

2.4.2 Real-Time Embeddability Evaluation

To evaluate the suitability of the models for real-time applications on embedded systems, three key metrics were assessed:

- Memory usage: measuring the memory footprint both during model initialization (i.e., loading into RAM) and during inference pass to ensure compatibility with devices with limited RAM.
- Floating-point operations per second (FLOPS): determining computational complexity during inference to estimate processing demands on embedded processors. When the required giga FLOPS (GFLOPS) exceeds the processing power of the embedded system, real-time performance is not achievable, even if sufficient RAM is available.
- Real-time factor (RTF): calculating the ratio of the time taken to generate audio (t_p) to the duration of the generated audio (t_b), with RTF values ($\frac{t_p}{t_b} < 1$) indicating real-time capability. Unlike memory usage and FLOPS, which provide theoretical estimates, RTF reflects the combined effects of computational complexity, model optimization, memory access patterns, bus speed, and other embedded system characteristics. It is calculated through empirical testing directly on the target device.

Table 2. Relevant specifications of currently available Raspberry Pi models for real-time audio processing tasks.

Model	Clock Rate (GHz)	GFLOPS	RAM (GB)
RPi 3B+	1.4	1.4	1
RPi 4B	1.5	1.5	[1, 2, 4, 8]
RPi 5B	2.4	2.4	[4, 8]

Note. GFLOPS = clock rate (GHz) × FLOPS per cycle.

The embeddability evaluation was conducted using a Raspberry Pi 4 with 2 GB of RAM as a representative embedded platform. The Raspberry Pi family was focused on due to its popularity, widespread use, and comprehensive documentation. Table 2 summarizes the theoretical specifications and limitations of the Raspberry Pi models currently on the market, including CPU architecture and available RAM. Multicore models (such as RPi 4B) have higher total FLOPS values; however, the implementations of DDSP models used in this work did not exploit any multiprocessing. Thus, the primary CPU constraint is GFLOPS per core.

Table 3. Comparison of each DDSP model, regarding GFLOPS and memory usage during initialization and forward phases.

		Memory	
DDSP Model	GFLOPS	Initialize (MB)	Forward (MB)
HpN-full	1.667	16.45	43.88
HpN-reduced	0.003	0.09	29.21
DDX7-full	0.100	1.59	8.45
DDX7-reduced	0.013	0.26	5.10
Wavetable-full	1.658	16.34	55.88
Wavetable-reduced	0.002	0.12	41.21

Note. The reported GFLOPS and memory (forward) correspond to the generation of 1 s of audio (i.e., 16,000 samples).

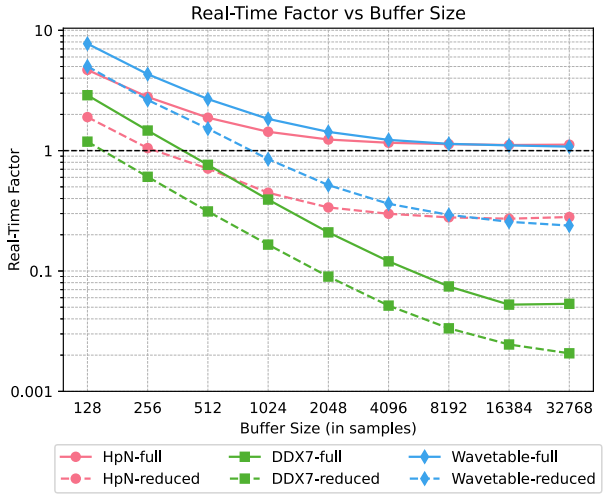

Fig. 3. Plot of the mean RTF against buffer size for all DDSP models for each version. Mean computed over 100 runs on a single thread of a Raspberry Pi 4 with 2 GB of RAM.

Three Python libraries were used for evaluating the footprint of each model: `PyTorch profiler`,[5] `PyTorch benchmark`,[6] and `fvcore`.[7] These libraries allow accurate profiling of memory consumption, FLOPs, and inference speed. The RTF values were obtained by averaging 100 inference runs for each model.

3 RESULTS

3.1 Embeddability

The results summarized in Table 3 highlight how HpN-full and Wavetable-full are incompatible with real-time applications on devices with constrained resources and would theoretically require the computational power of a more capable system. These models require over 1.6 GFLOPS, which exceeds the capabilities of Raspberry Pi up to model 4B. As proof of this, the plot in Fig. 3 shows that HpN-full and Wavetable-full fail to achieve real-time performance (RTF < 1) across all tested buffer sizes. In contrast, the reduced versions of these models lower the computational demands. For instance, HpN-reduced and Wavetable-reduced require less than 5 mega FLOPS, theoretically enabling them to run even on systems with lower resources than the RPi 4. The HpN-reduced model achieves real-time performance up to buffers of 512 samples, while the Wavetable achieves up to 1,024 samples.

The DDX7 models stand out for their real-time performances. Being already an order of magnitude smaller than other full models, DDX7-full requires only 0.1 GFLOPS, less than a tenth as much as the other models. This allows it to achieve real-time performance with buffer sizes of up to 512 samples, as shown in Fig. 3, and even outperform the reduced versions of other models with buffers greater than 1,024 samples.

Of all the models under review, the DDX7-reduced model is the only one to achieve real-time performance even with buffers of 256 samples, despite being the largest model among the reduced versions, both in terms of GFLOPS and in terms of memory required for initialization. This capability can be directly linked to the lower memory usage required by DDX7 models during the forward phase.

The memory required by a model can significantly impact its execution speed on embedded systems, as these devices often have limited RAM and cache memory. If a model exceeds the available memory, the system may resort to swapping or paging operations, which drastically degrade performance. Starting with buffers of 1,024 samples, the DDX7-full achieves better real-time performance than even the reduced versions of the other two models, which require lower computational cost in terms of GFLOPS.

This can be attributed to DDX7's architectural characteristics. Unlike the HpN and Wavetable models, which use recurrent neural network–based decoders, DDX7 employs convolutional layers that are computationally intensive but more memory-efficient. Convolutional neural networks leverage local computation patterns that are better optimized for cache performance on modern hardware, whereas recurrent neural networks rely on sequential computations that are more memory-intensive. As a result, DDX7 minimizes cache misses and eliminates the need for costly disk-based memory operations such as paging or swapping, especially with larger buffer sizes. In addition, the models analyzed in this work generated only monophonic audio (i.e., one note at a time). The synthesis of polyphonic melodies multiplies the computational and memory requirements. In these scenarios, memory-efficient models such as the DDX7 offer clear advantages, enabling polyphonic generation without exceeding memory limits.

Although the evaluation was conducted on a Raspberry Pi 4, the authors point out that the results may not be generalizable to other embedded architectures with slower memory buses or different processing capabilities, which may encounter additional bottlenecks not observed in this study. Furthermore, the models were tested in PyTorch, which is a realistic inference environment but does not reflect optimized production settings. In practice, the models can be further optimized for embedded systems by adopt-

[5] https://pytorch.org/docs/stable/profiler.html.
[6] https://pytorch.org/docs/stable/benchmark_utils.html.
[7] https://github.com/facebookresearch/fvcore.

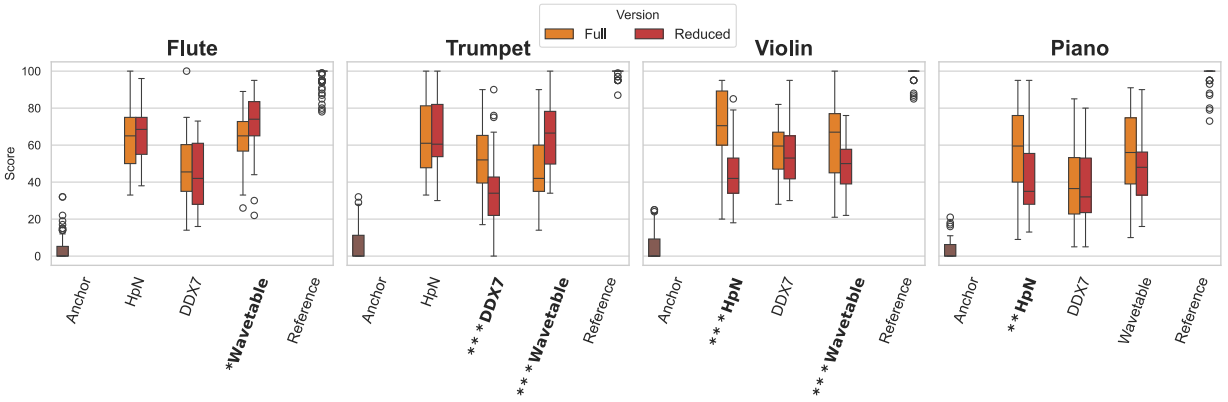

Fig. 4. Boxplots of audio quality scores given to each version of the synthesis model during each trial in the listening test. Bold labels in the x axis indicate significant differences between the full and reduced versions of the DDSP model in pairwise Wilcoxon signed-rank tests with false discovery rate correction. * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$.

ing dedicated inference engines, such as TorchScript or ONNX Runtine, which can run in real-time and are less resource-hungry. These optimizations could significantly improve performance but are beyond the scope of this study. Therefore, comparative trends between models, which have emerged in this work, should remain consistent across other implementations.

3.2 Audio Quality

FADs were computed using embeddings extracted from VGGish and EnCodec, as detailed in Table 4. Listening test results are visualized in Fig. 4, which highlights pairwise comparisons between the DDSP model versions, divided by musical instrument dataset. Statistical analysis was performed using Wilcoxon's signed-rank test with false discovery rate for p value correction.

It was observed that across all models and datasets, neither VGGish nor EnCodec consistently aligned with subjective listening test results (see Table 4). This highlights the limitations of relying solely on FAD for audio quality evaluation, as embedding models may inadequately capture all perceptual nuances. Therefore, the results obtained from listening tests were prioritized to evaluate the audio quality of the various models. To assess the statistical power of this study, a post hoc analysis was performed. The findings indicate that the experiment had an average power of 0.878 and an average effect size of 0.473 across all tests. When considering only statistically significant comparisons, the average effect size was 0.422, and the average statistical power was 0.98, demonstrating that the sample size of 18 participants was sufficient to detect meaningful differences. Below, the performance of each architecture and dataset is discussed.

3.2.1 Harmonic-plus-Noise

No statistically significant difference (i.e., $p < 0.05$) was found between the HpN-full and HpN-reduced models in the flute and trumpet datasets during listening tests, despite the FAD differentials (see Table 4). This implies that the reduced model, which achieves better real-time performance, could be used without substantial audio quality loss for these instruments. In contrast, the HpN-full model obtained significantly higher subjective audio quality for violin and piano ($p = 4 \times 10^{-8}$ and $p = 1 \times 10^{-3}$, respectively).

This discrepancy can be attributed to the different dynamic range of the instruments. Flute and trumpet, which have relatively low dynamic ranges, are simpler to synthesize, allowing the reduced decoder to effectively control the synthesizer as well as the full version. In contrast, violin and piano have more complex dynamics, making the difference in size and capacity between the two models more noticeable in their ability to synthesize high-quality audio.

3.2.2 DDX7

The full model performed significantly better than the reduced model with the trumpet dataset, as shown by both FAD scores and listening tests (with $p = 6 \times 10^{-5}$). On the contrary, no significant differences were observed between

Table 4. MOS and FAD, computed using VGGish and EnCodec embeddings, for all models and instruments.

	FAD (VGGish)				FAD (EnCodec)				MOS			
DDSP Model	Flute	Trumpet	Violin	Piano	Flute	Trumpet	Violin	Piano	Flute	Trumpet	Violin	Piano
HpN-full	**2.80**	**0.54**	1.51	2.21	87.55	**18.57**	**19.33**	**49.61**	63.8	64.3	**69.5**	**58.2**
HpN-reduced	2.91	3.85	4.84	1.94	95.31	31.08	49.78	53.08	65.3	65.6	45.3	42.8
DDX7-full	4.29	2.72	6.11	3.78	114.87	33.54	36.51	61.09	48.4	52.1	56.8	38.3
DDX7-reduced	6.70	4.89	7.33	4.23	132.78	55.94	39.55	66.05	44.0	35.5	53.9	36.9
Wavetable-full	4.23	2.73	**1.40**	**1.41**	**86.60**	41.51	19.40	57.54	62.6	45.3	62.9	55.0
Wavetable-reduced	3.48	3.51	3.91	3.12	97.60	28.52	49.88	56.79	**71.4**	**66.3**	48.2	48.7

Note. Best values in each column are bold; second-best values are underlined.

Table 5. Effects of KD in students' listening tests results.

	FLUTE			TRUMPET			VIOLIN			PIANO		
KD configuration	HpN	DDX7	Wavetable	HpN	DDX7	Wavetable	HpN	DDX7	Wavetable	HpN	DDX7	Wavetable
AD	–	–	↘*	–	↗***	↘**	–	–	–	↘*	–	↘*
CD	↗**	–	↘***	–	↗**	–	–	↘**	–	↗***	↘***	–
CD (Pretrained W.Table)			↘***			↘**			↗**			–

Note. Statistically significantly better teachers (green background) improve students (↗) in most cases. Similarly, teachers worse than students (red) decrease students' performance (↘). When teachers and students show similar MOS (white), KD generally shows no improvement (–) or even decreases it. ↗, ↘ and – indicate statistically significant improvement, degradation, or no change, respectively. * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$.

the full and reduced versions with the other dataset in the listening tests, despite differences in FAD.

The pronounced difference for the trumpet dataset can be explained by the complexity of the corresponding FM patch. As noted in the original DDX7 paper [15], the trumpet patch required more intricate manipulation, even benefiting from using simpler patches with fewer oscillators. The full model, with its larger capacity, was better equipped to control this more complex patch, whereas the reduced model struggled to achieve the same level of performance. Overall, the DDX7 models consistently had lower audio quality compared to other DDSP architectures.

3.2.3 Wavetable

The Wavetable models exhibited unexpected behavior. Surprisingly, the Wavetable-reduced model outperformed the full version in listening tests for both flute and trumpet datasets ($p = 0.016$ and $p = 1.4 \times 10^{-5}$, respectively). This behavior is likely tied to the training process, where the wavetables are learnable. The authors hypothesize that, for simpler instruments such as flute and trumpet, the full model appears to overfit early to suboptimal wavetables, neglecting their refinement during training. Conversely, the reduced model, with fewer parameters, places greater emphasis on optimizing wavetable quality, resulting in better-perceived audio. For more complex instruments like violin and piano, the wavetables of the full version continue improving throughout training, ultimately surpassing those of the reduced model. Indeed, the Wavetable-full model obtained significantly higher subjective audio quality for violin ($p = 8 \times 10^{-5}$).

3.3 Distillation Results

The application of KD yielded mixed results across different models and datasets, with students improving over their counterpart trained without KD generally when guided by well-performing teachers but degrading with poor teachers. Table 5 summarizes the results of applying the different distillation methods for all models and datasets. More details on the results obtained in the various distillation experiments can be found on the accompanying page.[3] Below, the effect of KD for each DDSP model is discussed.

3.3.1 DDX7

KD produced the most striking positive outcome with the trumpet dataset across all KD configurations under examination. This improvement was reflected in listening tests, see Fig. 5(a), showing incremental gains in the student's ability to synthesize audio with the similar quality as the teacher. For all other instruments, where the difference between the teacher and student models without KD was not significant, the application of KD often led to a deterioration in the DDX7 student's performance. In these cases, KD can amplify inconsistencies or inefficiencies in the student's control of the synthesizer patch, particularly for complex instruments. This highlights the importance of the teacher

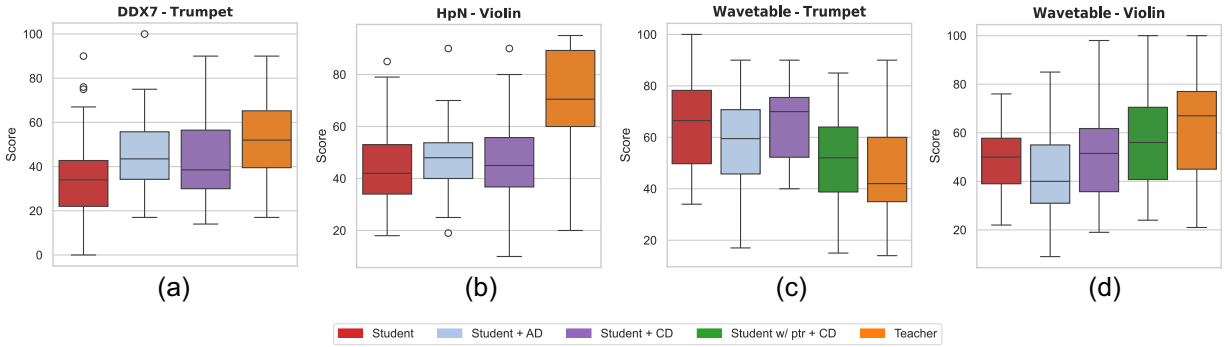

Fig. 5. Boxplots illustrating the effects of KD on different instruments and models: (a) DDX7-Trumpet, (b) HpN-Violin, (c) Wavetable-Trumpet, and (d) Wavetable-Violin. (a, d) Positive impact of KD when the teacher outperforms the student without KD. (b) Limited efficacy of KD, likely due to the restricted capacity of the student model. (c) Detrimental impact of KD when the teacher is inferior to the student without KD.

having clear superiority over the student for KD to be effective.

3.3.2 Harmonic-plus-Noise

HpN models presented varied results depending on the dynamic range of the instrument, in a similar way that was already observed in the previous subsection. For the trumpet dataset, KD failed to improve student performance, as the teacher and student models already produced similar results. In contrast, for the flute dataset, the CD strategy improves the MOS of the student even when the teacher does not perform better. It is suspected that CD forces the student to mimic the teacher's control envelopes, while also smoothing them, resulting in a more pleasant audio output from the student with KD. With the violin dataset, KD was ineffective because the reduced student model likely reached its capacity limit, preventing it from benefiting from the teacher's guidance, see Fig. 5(b). In contrast, the piano dataset demonstrated significant improvements with CD. The relative simplicity of the Jazznet dataset, which lacks complex articulations, likely allowed the KD process to enhance the student's ability to model the piano's dynamics effectively.

3.3.3 Wavetable

Results show that KD can also work in reverse: When the teacher's performance is inferior to that of a student, KD tends to reduce the students' MOS, as evidenced by the flute and trumpet models where the application of KD caused a degradation in the student's performance, as seen in Fig. 5(c). For the violin model, CD can guide the student to leverage pretrained teacher's wavetables, achieving a similar performance comparable to the teacher, see Fig. 5(d). In this case, by initializing the student's wavetables with the teacher's, the student model is induced to focus mainly on emulating the teacher's control parameters, rather than simultaneously optimizing both the wavetables and their usage. It is important to note that this strategy can also work in reverse. As observed in the Wavetable-Trumpet case, if the teacher wavetables themselves are suboptimal, this limitation will propagate to the student.

Overall, the results emphasize the dual-edged nature of KD. When the teacher model significantly outperforms the student, it can close the performance gap, as demonstrated with the DDX7-Trumpet and HpN-Piano cases. However, when the teacher model underperforms or exhibits specific weaknesses, KD can propagate these deficiencies to the student, as seen in the Wavetable with the trumpet dataset. Importantly, CD emerged as the most impactful technique, yielding both the most significant improvements and the most notable degradations. To ensure favorable outcomes with KD, it is imperative that the teacher model outperforms the student.

4 DISCUSSION

This study provides valuable insights into the strengths and challenges of deploying DDSP models on embedded systems, particularly when combined with KD for model compression. One of the key takeaways is that small models, can often generate audio of comparable or even superior quality to their larger counterparts. This phenomenon suggests that small models may leverage the inductive bias inherent in DDSP more effectively, avoiding the potential overfitting of larger models, as illustrated by several models, such as HpN-Flute and Wavetable-Trumpet. However, in many cases, there is a clear trade-off between computational efficiency and audio quality, particularly when modeling complex dynamics of the piano or violin datasets. Smaller decoders, constrained by fewer parameters, often fail to represent the subtle amplitude modulations and timbral shifts required to convincingly model such instruments. In these cases, DDSP models face inherent challenges in capturing the nuanced variations required for their articulations and tonal characteristics.

In addition to this general challenge, each synthesis architecture introduces unique constraints that further influence the performance of reduced models. For example, in the FM-based DDX7 architecture, the complexity of the patch configuration can affect the model's ability to generate audio, as evidenced by the DDX7-Trumpet model. In contrast, patches with fewer modulators did not exhibit the same performance gap, highlighting how patch-specific complexity can amplify the limitations of smaller models.

Similarly, in Wavetable synthesis, the quality of the learned wavetables likely plays a pivotal role. For instruments like the violin and piano, it seems that larger models might continue to refine the wavetables throughout training, potentially leading to superior performance. This observation suggests that the balance between model complexity and wavetable quality could be critical.

In these scenarios, KD emerges as a valuable tool, particularly when using the CD, as it shows the more consistent results. However, this holds as long as the teacher outperforms the student; otherwise, the reverse effect occurs where the student worsen its listening test results. Second, the student must possess sufficient capacity to benefit from the distilled knowledge; it is suspected that this was not the case for the HpN-Violin experiments, where the limited capacity of the student constrained the effectiveness of KD.

From these experiments, it was found that CD generally works best with FM and HpN. CD with pretrained wavetables is best for wavetable synthesis. This means that there is no one-size-fits-all approach, and KD can be tailored to the specific requirements of different students.

Future work could explore hybrid strategies that integrate KD with other model compression techniques, such as pruning or quantization, to further optimize models for real-time synthesis on embedded systems. Another promising avenue is exploring the effect of KD across a broader range of neural architectures, such as VAEs, as well as designing new forms of distillation. Integrating VAEs in the DDSP paradigm allows manipulating latent representations, enabling a more expressive and fine-grained control over timbre variations [45, 46]. For instance, a dual-layered approach could apply CD to enhance the generative capabilities of the decoder, while a form of feature distillation

is simultaneously used to optimize the encoder's representation learning. Integrating advanced perceptual loss functions into the distillation process represents another compelling research direction. Perceptual metrics, such as the perceptual-neural-physical loss [47], could be leveraged to align the distilled models more closely with human auditory perception.

5 CONCLUSION

This study investigates the feasibility of deploying DDSP-based generative models on embedded systems, focusing on their embeddability and audio quality. Additionally, it explores the impact of KD as a model compression strategy through an in-depth analysis of three foundational DDSP models: HpN, FM, and Wavetable.

In several instances, it was observed that lightweight models can perform similarly well to their full-sized counterparts; however, a clear trade-off exists between model size and audio quality, particularly when capturing the nuanced dynamics of complex instruments. Moreover, synthesis architectures such as FM and Wavetable introduce additional constraints: FM models depend heavily on the complexity of the patch configuration, while Wavetable models are sensitive to the quality of learned wavetables. For these cases, it has been shown that applying KD can bridge the performance gap between full and reduced models. However, the effectiveness of KD was highly dependent on both the quality of the teacher model and the specific characteristics of the dataset. This underscores the importance of careful teacher selection and task-specific tuning of the KD strategy for specific model architectures to maximize its benefits.

6 ACKNOWLEDGMENT

G. A. Giudici's contributions are funded by the European Union under NextGenerationEU (M.D. 118/2023). Views and opinions expressed are those of the authors only and do not necessarily reflect those of the European Union or the European Research Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. F. Caspe's contributions are supported by UK Research and Innovation (grant number EP/S022694/1).

7 REFERENCES

[1] A. Caillon and P. Esling, "RAVE: A Variational Autoencoder for Fast and High-Quality Neural Audio Synthesis," *arXiv preprint arXiv:2111.05011* (2021 Dec.).

[2] F. Schneider, *ArchiSound: Audio Generation With Diffusion*, Master's thesis, ETH Zurich, Zurich, Switzerland (2023 Jan.).

[3] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "DiffWave: A Versatile Diffusion Model for Audio Synthesis," presented at the *International Conference on Learning Representations* (Online) (2021 May).

[4] G. Narita, J. Shimizu, and T. Akama, "GANStrument: Adversarial Instrument Sound Synthesis With Pitch-Invariant Instance Conditioning," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (Rhodes Island, Greece) (2023 Jun.). https://doi.org/10.1109/ICASSP49357.2023.10097250.

[5] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable Digital Signal Processing," in *Proceedings of the International Conference on Learning Representations*, pp. 12010–12028 (Addis Ababa, Ethiopia) (2020 Jan.).

[6] B. Hayes, J. Shier, G. Fazekas, A. McPherson, and C. Saitis, "A Review of differentiable Digital Signal Processing for Music and Speech Synthesis," *Front. Signal Process.*, vol. 3, paper 1284100 (2024 Jan.). https://doi.org/10.3389/frsip.2023.1284100.

[7] J. Wen, J. Nie, J. Kang, et al., "From Generative AI to Generative Internet of Things: Fundamentals, Framework, and Outlooks," *IEEE Internet Things Mag.*, vol. 7, no. 3, pp. 30–37 (2024 May). https://doi.org/10.1109/IOTM.001.2300255.

[8] A. Karapantelakis, P. Alizadeh, A. Alabassi, K. Dey, and A. Nikou, "Generative AI in Mobile Networks: A Survey," *Ann. Telecommun.*, vol. 79, no. 1–2, pp. 15–33 (2023 Aug.). https://doi.org/10.1007/s12243-023-00980-9.

[9] L. Turchet, "Smart Musical Instruments: Vision, Design Principles, and Future Directions," *IEEE Access*, vol. 7, pp. 8944–8963 (2019 Jan.). https://doi.org/10.1109/ACCESS.2018.2876891.

[10] N. Devis and P. Esling, "Neurorack: Deep Audio Learning in Hardware Synthesizers," presented at the *Workshop on Human Factors in DH* (Lausanne, Switerland) (2021 Dec.).

[11] D. Stefani and L. Turchet, "Real-Time Embedded Deep Learning on Elk Audio OS," presented at the *International Symposium on the Internet of Sounds* (Pisa, Italy) (2023 Oct.). https://doi.org/10.1109/IEEECONF59510.2023.10335204.

[12] N. Kalchbrenner, E. Elsen, K. Simonyan, et al., "Efficient Neural Audio Synthesis," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, pp. 2410–2419 (Stockholm, Sweden) (2018 Jul.).

[13] P. Esling, N. Devis, A. Bitton, et al., "Diet Deep Generative Audio Models With Structured Lottery," in *Proceedings of the 23rd International Conference on Digital Audio Effects* (Vienna, Austria) (2020 Sep.).

[14] D. Südholt, A. Wright, C. Erkut, and V. Välimäki, "Pruning Deep Neural Network Models of Guitar Distortion Effects," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 31, pp. 256–264 (2022 Nov.). https://doi.org/10.1109/TASLP.2022.3223257.

[15] F. Caspe, A. McPherson, and M. Sandler, "DDX7: Differentiable FM Synthesis of Musical Instrument Sounds," in *Proceedings of the 23rd International Society for Music Information Retrieval Conference*, pp. 608–616 (Bengaluru, India) (2022 Dec.).

[16] D.-Y. Wu, W.-Y. Hsiao, F.-R. Yang, et al., "DDSP-Based Singing Vocoders: A New Subtractive-Based Synthesizer and a Comprehensive Evaluation," in *Proceedings of the 23rd International Society for Music Information*

Retrieval Conference, pp. 76–83 (Bengaluru, India) (2022 Dec.).

[17] G. Assayag, L. Bonnasse-Gahot, and J. Borg, "Cocreative Interaction: Somax2 and the REACH Project," *Comput. Music J.*, vol. 46, no. 4, pp. 7–25 (2022 Dec.). https://doi.org/10.1162/comj_a_00662.

[18] N. S. Silva and L. Turchet, "Real-Time Pattern Recognition of Symbolic Monophonic Music," in *Proceedings of the 19th International Audio Mostly Conference: Explorations in Sonic Cultures*, pp. 308–317 (Milan, Italy) (2024 Sep.). https://doi.org/10.1145/3678299.3678329.

[19] L. Turchet, J. Pauwels, C. Fischione, and G. Fazekas, "Cloud-Smart Musical Instrument Interactions: Querying a Large Music Collection With a Smart Guitar," *ACM Trans. Internet Things*, vol. 1, no. 3, paper 15 (2020 Jun.). https://doi.org/10.1145/3377881.

[20] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv preprint arXiv:1503.02531* (2015 Mar.).

[21] Z. Li and D. Hoiem, "Learning Without Forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947 (2018 Dec.). https://doi.org/10.1109/TPAMI.2017.2773081.

[22] J. Wang, L. Gou, W. Zhang, H. Yang, and H.-W. Shen, "DeepVID: Deep Visual Interpretation and Diagnosis for Image Classifiers via Knowledge Distillation," *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 6, pp. 2168–2180 (2019 Mar.). https://doi.org/10.1109/TVCG.2019.2903943.

[23] W.-C. Chen, C.-C. Chang, and C.-R. Lee, "Knowledge Distillation With Feature Maps for Image Classification," in C. V. Jawahar, H. Li, G. Mori, and K. Schindler (Eds.), *Computer Vision – ACCV 2018*, Lecture Notes in Computer Science, pp. 200–215 (Springer, Cham, Switzerland, 2019). https://doi.org/10.1007/978-3-030-20893-6_13.

[24] Y. Chebotar and A. Waters, "Distilling Knowledge From Ensembles of Neural Networks for Speech Recognition," in *Proceedings of the 17th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3439–3443 (San Francisco, CA) (2016 Sep.). https://doi.org/10.21437/Interspeech.2016-1190.

[25] J. H. M. Wong and M. J. Gales, "Sequence Student-Teacher Training of Deep Neural Networks," in *Proceedings of the 17th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 2761–2765 (San Francisco, CA) (2016 Sep.). https://doi.org/10.21437/Interspeech.2016-911.

[26] Y.-C. Chen, Z. Gan, Y. Cheng, J. Liu, and J. Liu, "Distilling Knowledge Learned in BERT for Text Generation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7893–7905 (Online) (2020 Jul.). https://doi.org/10.18653/v1/2020.acl-main.705.

[27] A. Oord, Y. Li, I. Babuschkin, et al., "Parallel WaveNet: Fast High-Fidelity Speech Synthesis," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, pp. 3918–3926 (Stockholm, Sweden) (2018 Jul.).

[28] A. Oord, S. Dieleman, H. Zen, et al., "WaveNet: A Generative Model for Raw Audio," in *Proceedings of the 9th ISCA Workshop on Speech Synthesis*, p. 125 (Sunnyvale, CA) (2016 Sep.).

[29] J. Nistal, S. Lattner, and G. Richard, "DarkGAN: Exploiting Knowledge Distillation for Comprehensible Audio Synthesis With GANs," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference*, pp. 484–492 (Online) (2021 Nov.).

[30] M. Pezzat-Morales, H. Perez-Meana, and T. Nakashika, "Fast Jukebox: Accelerating Music Generation With Knowledge Distillation," *Appl. Sci.*, vol. 13, no. 9, paper 5630 (2023 May). https://doi.org/10.3390/app13095630.

[31] P. Dhariwal, H. Jun, C. Payne, et al., "Jukebox: A Generative Model for Music," *arXiv preprint arXiv:2005.00341* (2020 Apr.).

[32] S. Shan, L. Hantrakul, J. Chen, M. Avent, and D. Trevelyan, "Differentiable Wavetable Synthesis," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4598–4602 (Singapore) (2022 May). https://doi.org/10.1109/ICASSP43922.2022.9746940.

[33] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, "Creating a Multitrack Classical Music Performance Dataset for Multimodal Music Analysis: Challenges, Insights, and Applications," *IEEE Trans. Multimed.*, vol. 21, no. 2, pp. 522–535 (2019 Feb.). https://doi.org/10.1109/TMM.2018.2856090.

[34] T. Adegbija, "Jazznet: A Dataset of Fundamental Piano Patterns for Music Audio Machine Learning Research," presented at the *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (Rhodes Island, Greece) (2023 Jun.). https://doi.org/10.1109/ICASSP49357.2023.10096620.

[35] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "CREPE: A Convolutional Representation for Pitch Estimation," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 161–165 (Calgary, Canada) (2018 Apr.). https://doi.org/10.1109/ICASSP.2018.8461329.

[36] B. C. Moore, B. R. Glasberg, and T. Baer, "A Model for the Prediction of Thresholds, Loudness, and Partial Loudness," *J. Audio Eng. Soc.*, vol. 45, no. 4, pp. 224–240 (1997 Apr.).

[37] N. Masuda and D. Saito, "Synthesizer Sound Matching With Differentiable DSP," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference*, pp. 428–434 (Online) (2021 Nov.).

[38] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet Audio Distance: A Reference-Free Metric for Evaluating Music Enhancement Algorithms," in *Proceedings of the 20th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 2350–2354 (Graz, Austria) (2019 Sep.). https://doi.org/10.21437/Interspeech.2019-2219.

[39] A. Gui, H. Gamper, S. Braun, and D. Emmanouilidou, "Adapting Fréchet Audio Distance for Generative Music Evaluation," in *Proceedings of the*

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1331–1335 (Seoul, South Korea) (2024 Apr.). https://doi.org/10.1109/ICASSP48485.2024.10446663.

[40] S. Hershey, S. Chaudhuri, D. P. Ellis, et al., "CNN Architectures for Large-Scale Audio Classification," presented at the *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 131–135 (New Orleans, LA) (2017 Mar.). https://doi.org/10.1109/ICASSP.2017.7952132.

[41] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High Fidelity Neural Audio Compression," *Trans. Mach. Learn. Res.*, (2023 Sep.). https://openreview.net/forum?id=ivCd8z8zR2.

[42] B. Hayes, C. Saitis, and G. Fazekas, "Neural Waveshaping Synthesis," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference*, 254–261 (Online) (2021 Nov.).

[43] M. Schoeffler, S. Bartoschek, F.-R. Stöter, et al., "webMUSHRA — A Comprehensive Framework for Web-Based Listening Tests," *J. Open Res. Softw.*, vol. 6, no. 1, paper 8 (2018 Feb.). https://doi.org/10.5334/jors.187.

[44] D. O'Shaughnessy, "Linear Predictive Coding," *IEEE Potentials*, vol. 7, no. 1, pp. 29–32 (1988 Feb.). https://doi.org/10.1109/45.1890.

[45] Y. Liu, C. Jin, and D. Gunawan, "DDSP-SFX: Acoustically-Guided Sound Effects Generation With Differentiable Digital Signal Processing," in *Proceedings of the 27th International Conference on Digital Audio Effects*, pp. 216–221 (Guilford, UK) (2024 Sep.).

[46] Y. Liu and C. Jin, "Simi-SFX: A Similarity-Based Conditioning Method for Controllable Sound Effect Synthesis," *arXiv preprint arXiv:2412.18710* (2024 Dec.).

[47] H. Han, V. Lostanlen, and M. Lagrange, "Perceptual–Neural–Physical Sound Matching," presented at the *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (Rhodes Island, Greece) (2023 Jun.). https://doi.org/10.1109/ICASSP49357.2023.10095391.

THE AUTHORS


Gregorio Andrea Giudici


Franco Caspe


Leonardo Gabrielli


Stefano Squartini


Luca Turchet

Gregorio Andrea Giudici is a Ph.D. student at the Department of Information Engineering and Computer Science of the University of Trento, Italy. He is a member of the Creative, Intelligent & Multisensory Interactions Laboratory (CIMIL), and his research focuses on smart musical instruments. He is currently working on lightweight AI models for embedded systems for interactive live performance.

•

Franco Caspe is a Ph.D. student at the Centre for Digital Music (Queen Mary University of London) and the Augmented Instruments Lab (Imperial College). His Ph.D. research aims to bridge the gap between acoustic instruments and synthesizers, by using deep learning both as an analytical tool to capture performance characteristics from instrumental audio and as a generative tool for synthetic sound rendering.

•

Leonardo Gabrielli is an Assistant Professor at the Department of Information Engineering, Università Politecnica Marche. His main research topics are related to audio signal processing and machine learning with application to sound synthesis, computational sound design, networked music performance and several audio classification tasks. He is part of the board of the Associazione di Informatica Musicale Italiana.

•

Stefano Squartini is a Full Professor of Electrical Engineering at the Università Politecnica delle Marche. His research interests primarily focus on DSP and AI techniques, with applications across various domains, including digital audio. His research group have been actively developing AI-based algorithms for speech processing, environmental sound analysis, and audio restoration.

•

Luca Turchet is an Associate Professor at the Department of Information Engineering and Computer Science of the University of Trento, Italy. He is the Chair of the IEEE Emerging Technology Initiative on the Internet of Sounds and the founding president of the Internet of Sounds Research Network.