F. Caspe, J. Shier, M. Sandler, C. Saitis, and A. McPherson,
"Designing Neural Synthesizers for Low-Latency Interaction,"
J. Audio Eng. Soc., vol. 73, no. 5, pp. 240–255 (2025 May).
https://doi.org/10.17743/jaes.2022.0204.

Designing Neural Synthesizers for Low-Latency Interaction

FRANCO CASPE,[1,*] *AES Student Member*, **JORDIE SHIER,**[1] **MARK SANDLER,**[1] *AES Fellow*,
(f.s.caspe@qmul.ac.uk) (j.m.shier@qmul.ac.uk) (mark.sandler@qmul.ac.uk)
CHARALAMPOS SAITIS,[1] **AND ANDREW MCPHERSON**[2]
(c.saitis@qmul.ac.uk) (andrew.mcpherson@imperial.ac.uk)

[1] *Centre for Digital Music, Queen Mary University of London, London, UK*
[2] *Dyson School of Engineering, Imperial College, London, UK*

Neural audio synthesis (NAS) models offer interactive musical control over high-quality, expressive audio generators. While these models can operate in real time, they often suffer from high latency, making them unsuitable for intimate musical interaction. The impact of architectural choices in deep learning models on audio latency remains largely unexplored in the NAS literature. In this work, the authors investigate the sources of latency and jitter typically found in interactive NAS models. They then apply this analysis to the task of timbre transfer using the RAVE model (Realtime Audio Variational autoEncoder), a convolutional variational autoencoder for audio waveforms introduced by Caillon and Esling in 2021. Finally, an iterative design approach for optimizing latency is presented. This culminates with a model the authors call BRAVE (Bravely Realtime Audio Variational autoEncoder), which is low-latency and exhibits better pitch and loudness replication while showing timbre modification capabilities similar to RAVE. It is implemented in a specialized inference framework for low-latency, real-time inference, and a proof-of-concept audio plugin compatible with audio signals from musical instruments is presented. The authors expect the challenges and guidelines described in this document to support NAS researchers in designing models for low-latency inference from the ground up, enriching the landscape of possibilities for musicians.

0 INTRODUCTION

In recent years, interest in interactive neural audio synthesis (NAS) algorithms has increased within the music research community. NAS algorithms utilize deep neural networks (DNNs), trained on audio datasets, and recent advancements have paved the way for high-quality synthesis [1]. The flexibility of these architectures has enabled a variety of control modalities, ranging from MIDI input [2] to audio-based control [3, 4] and has inspired entirely new interfaces [5]. The use of variational autoencoders (VAEs) [6] for NAS has been particularly fruitful, with the RAVE model [4] facilitating a range of real-time interactive musical applications and inspiring the development of novel interaction modalities [5, 7].

Many NAS algorithms support real-time inference, but they are generally unsuitable for live instrumental performance because they often introduce latencies of several hundred milliseconds [8]. This is an important factor in music performance; systems that offer low latency can lead to richer and better-perceived experiences [9–11]. However, latency in NAS models has yet to be formally investigated, which hinders the development of low-latency musical NAS models.

This work first presents an analysis of primary latency sources in interactive, audio-to-audio NAS algorithms. These are (1) buffering delay, (2) cumulative delay (3) representation delay (4) data-dependent latency, and (5) positional uncertainty (jitter). As a case study, RAVE's architecture is analyzed for these sources of latency, highlighting how this is unsuitable for low-latency interaction. To support this analysis, an open-source toolkit is designed that is used to empirically evaluate latency during inference.

Building on the initial analysis, a design approach is presented where the RAVE architecture is iteratively modified to achieve the target latency and jitter goals for musical interaction, as well as computing requirements for real-time inference on a consumer CPU. The iterative design

*To whom correspondence should be addressed, email: f.s.caspe@qmul.ac.uk

culminates with a model the present authors call BRAVE (Bravely Realtime Audio Variational autoEncoder), which is released in a proof-of-concept audio plugin for musicians to experiment with. See the accompanying website for training source code and audio examples.[1]

The model variants are evaluated on an audio timbre transfer task (i.e., modifying the timbre of an audio input while preserving its musical content), and it is shown that BRAVE improves upon RAVE in preserving musical content (in terms of pitch and loudness) and presents similar capabilities for reproducing the timbre of the training dataset. The results of this analysis and evaluation point to the importance of the model's receptive field in enabling rich musical affordances such as timbre transfer. The ability of the decoder to consider time-varying trajectories of latent representations, supported by a suitable receptive field, is critical in the emergence of useful musical interactions. These findings suggest practical methods for future development of low-latency NAS systems beyond RAVE.

In summary, this work puts latency at the center of NAS design. A design framework is provided that leverages analytical and empirical observations grounded in the authors' musical goals, which motivate and inform architectural decisions, ensuring that both performance and responsiveness are prioritized. The authors hope that this perspective, along with the results of this research, will not only support the improvement of existing NAS models but also support the development of novel systems that enrich the landscape of Digital Musical Instruments (DMIs) for musicians.

1 BACKGROUND

1.1 Real-Time Constraints for Musical Interaction

Low action-to-sound latency and timing stability are crucial in DMIs for supporting control intimacy, time-keeping, and developing performance skills and personal style [12]. Wessel and Wright [13] suggest an upper bound on latency of 10 ms and a jitter (variation of latency) of ± 1 ms for musical interaction. Furthermore, Jack et al. [11] conducted an empirical study that supported this recommendation for percussive instruments, demonstrating that 10 ms of latency was acceptable to performers in the absence of jitter; however, 20 ms of latency or 10 ± 3 ms of latency significantly degraded the experience of performance. Lester and Boley [14] performed subjective tests with practicing musicians to investigate the effect of latency under different monitoring scenarios. Their results highlight the context-dependent nature of latency perception with sensitivity among musicians varying based on the instrument and type of monitoring device. In general, however, their results reinforce the 10-ms bound for minimal impact on musical performance. This work aims to reach strict latency constraints that approximate those of 10 ± 1 ms reported in previous work focused on instrumental interaction.

[1] https://fcaspe.github.io/brave.

1.2 Interactive Neural Audio Synthesis

NAS is a technique that employs DNNs as learned audio synthesizers. Neural synthesis has its origins in offline generation; however, increased computation capabilities have enabled real-time inference, which has opened the doors for NAS use in live performance systems with control schemes that span from MIDI inputs [2, 15] to music information retrieval features [16], latent spaces [17, 18], and other audio signals [4, 19]. Despite this, latency is typically only addressed after models are trained [8]. A common solution in neural audio plug-in development is to report latency to the audio workstation for it to be compensated by realigning other audio tracks, which is incompatible with this live use case.

Recently, a set of NAS models such as RAVE and the Differentiable DSP (DDSP) Decoder [3] have gained popularity due to their ability to transform audio in real time according to learned timbral and dynamic characteristics of a dataset. These models can supplement musical instruments with new timbral possibilities; however, such models are not optimized for real-time interaction with the audio of musical instruments, due to their high latency.

Low-latency exceptions to this case are specific models designed for emulating traditional audio effects [20] and also systems for audio-driven control of traditional synthesizers like the Envelope Learning [21], Timbral Remapping [22], and guitar percussive technique [23] models, designed with low-latency interaction with musical instruments in mind. However, these are restricted to specific DSP synthesis techniques such as monophonic FM synthesis, percussive 808-style emulation, or modal synthesis. In certain contexts, the base latency of NAS models like RAVE may be deemed acceptable, as in with novel musical interfaces where a fast action-to-sound response is not necessarily expected [24], or in audio plugins for production use [8]. This paper focuses on audio-driven interactions, specifically timbre transfer, where audio from an instrument is transformed in real time. RAVE was chosen over DDSP due to its flexibility in handling a broader range of source material, without being constrained to monophonic or harmonic instruments.

1.3 Real-Time Factor

Algorithms, including neural nets, are assessed for real-time operation using the real-time factor (RTF), defined as the ratio between processing time t_p and the duration of the input t_i, $RTF = t_p/t_i$. It follows that a model requires an $RTF < 1$ for a specific block size to process it in real time with specific hardware. Low-latency models require processing short audio blocks at a high rate, which limits the available processing time for a single inference pass.

1.4 Streaming NAS Algorithms

Audio algorithms that support real-time interaction must process sequential blocks of one or more audio samples at a time instead of the entire duration of an input in a single pass. This approach is known as *block processing*. Generative models that support this scheme are called *streaming*

models. In this case, block processing has to be supported by all the layers of a model [8]. Each layer is required to keep track of its internal state between subsequent forward passes to guarantee a continuous output signal between blocks. For instance, convolutional layers are reconfigured with a *cached padding* mechanism [25] where the end of the input tensor is retained and used to pad the next input.

The outputs from causal systems do not depend on future inputs. This is a strict requirement for streaming models. Nevertheless, noncausal systems with finite *lookahead*, that is, systems that process a finite number of future input timesteps, can be reconfigured for causal operation by introducing delays that allow all required samples to be acquired before processing. Although noncausal models typically afford better reconstruction quality [8, 26], in some applications it is desirable to avoid this additional cumulative delay by training in a causal way [27, 28, 23].

Designers of streaming models can achieve tolerable latency for several tasks simply by employing causal training and processing relatively short audio blocks. The authors of the EnCodec neural codec [26] introduced a streaming and causal architecture, reporting a *theoretical minimum latency* of 13 ms, determined by the shortest block supported. However, given the model size (23.3 M parameters), it is unlikely that inference can be performed on a consumer CPU at the frame rate that would support such latency. In a plausible streaming scenario, longer audio windows are accumulated before processing by the coder; this improves its RTF but increases latency, although it remains within tolerable limits for the task.

Similarly, the spoken dialog framework Moshi [29] uses causal models to encode and decode audio frames of 80 ms, performing temporal aggregation in parallel with a causal language model, yielding a final latency of 200 ms. This demonstrates how, for several applications, tolerable latency can be achieved by optimizing specific system components in isolation. Nevertheless, to support instrumental interaction (10 ms total latency, 3 ms jitter), that authors argue that a more comprehensive analysis of potential latency sources is necessary before design.

1.5 Timbre Transfer

Timbre transfer involves transforming a musical audio signal so that timbre is altered while preserving key performance parameters such as melodies, accents, and rhythm. Most research in timbre transfer is grounded in the classic (but disputed [30]) American National Standards Institute definition, which describes timbre as an attribute of auditory sensation that differentiates sounds of the same pitch and loudness [31]. In this context, timbre is typically treated as a global attribute encompassing the sonic quality of a specific instrument.

Recent approaches in timbre transfer, such as DDSP [3, 32], build on this by combining explicit pitch and loudness controls with deep learning models that implicitly model timbre through a spectral loss function. These models have demonstrated real-time timbre transfer capabilities [33] but remain constrained by their application to monophonic and harmonic instruments.

NAS provides more flexible, source-agnostic methods for modeling musical audio and learning disentangled representations [34]. Engel et al. pioneered an autoencoder architecture [35], which enabled interpolations of instrumental timbres. Mor et al. [36] and Alinoori and Tzerpos [37] extended this architecture for timbre transfer by using instrument-specific WaveNet decoders and by applying a teacher forcing technique with paired audio data, respectively. Building on techniques from neural style transfer in the image domain [38], timbre transfer has also been tackled using various architectures, including autoencoders [39–41], generative adversarial networks [42, 43], attention-based methods [44], vector-quantized VAEs [45, 46], and, more recently, diffusion models [47–49].

This work focuses on timbre transfer using RAVE, an autoencoder-based NAS architecture. To the authors' knowledge, RAVE is the only NAS-based timbre transfer model to enable real-time, streaming operation; its architecture—specifically the information bottleneck of its autoencoder design—and training on datasets with few instruments have enabled timbre transfer affordances. This exploration centers on how RAVE's architectural design influences latency and, critically, how this impacts timbre transfer results.

2 LATENCY IN NEURAL AUDIO SYNTHESIS

This section first presents an analysis of the sources of latency and jitter that can affect interactive NAS models. Then, the focus of this work is set on RAVE, a successful interactive NAS algorithm, and its sources of latency are analyzed, highlighting how its design decisions hinder its ability to achieve low action-to-sound latency.

2.1 Latency Sources

Here, the architectural sources of latency that result in a model's delayed response to an input are presented. We identify four sources, namely, buffering, cumulative and representation delay, and data-dependent latency.

2.1.1 Buffering

Real-time audio systems typically work in a pipeline employing a *double buffering* approach, where one audio block is computed while the next is being captured. In a double buffering scenario, the total output buffering latency is of two block lengths: one for sample acquisition and one for computing [12]. Therefore, the block size determines the upper limit for computation time. This latency source is not visible during model training.

2.1.2 Cumulative Delay

Cumulative delay arises from reconfiguring noncausal convolutional layers for streaming by adding a delay line that the noncausal layers use for looking ahead in time. This lag, which accumulates through successive layers, is called *cumulative delay*. It can increase latency in noncausally

trained models by hundreds of milliseconds [8] compared to the same architecture trained causally. This delay appears after training, when the model is reconfigured for streaming. Its final value depends on the configuration of all the layers and can be computed by propagating the delay of the first layer throughout the network. The PyTorch package `cached_conv` presented by Caillon and Esling [8] can compute this delay during model building.

2.1.3 Representation

Representation latency is denoted as the number of samples it takes a feature extractor to produce an expected output. NAS models use audio features as input, such as fundamental frequency (F0) and loudness [3], filter banks such as Pseudo-Quadrature Mirror Filters (PQMF) [4], short-time Fourier transforms [50], or Mel filterbanks [51], to name a few. For instance, linear phase finite impulse response (FIR) filterbanks typically exhibit a group delay D_g of approximately half of the filter length N_f [52]. On the other hand, other audio feature extractors, such as F0 trackers can exhibit varying latency, depending on their analysis window and the nature of the signal being measured [53].

2.1.4 Data-Dependent Latency

Convolutional layers are FIR filters, and their coefficients depend ultimately on training data. Models based on convolutional networks can exhibit different degrees of latency depending on the learned coefficients. However, this has not been analyzed by previous literature. SEC. 3 analyzes the effect of the training data on the latency, comparing models with the same architecture trained on different datasets.

2.2 Positional Uncertainty (Jitter)

Lossy representations computed using block processing encode the position of an event with a temporal resolution of up to a block size. The authors call this *positional uncertainty*. In this case, the exact position in samples of an event is lost, and the model's response will depend not only on the input and training method but also on the relative position of the event within an input window, which generates jitter. As mentioned earlier, jitter has to be ideally kept under ± 1 ms to support intimate control with musical instruments. This requirement suggests that smaller block sizes are desirable to minimize jitter.

2.3 Case Study: RAVE

The design of the RAVE model [4] is addressed because of its extensive application in real-time synthesis and wide interaction possibilities [7, 5, 54]. This work is based on the `v1` version, available at the official repository,[2] and presented in the original paper. For the analysis, the authors assume models operating at a sample rate of 44.1 kHz.

RAVE is a convolutional VAE featuring a compressing encoder and decompressing decoder with strided convolutional layers. For efficiency purposes, it does not work directly with raw audio waveforms and instead encodes and decodes an audio representation down-sampled and split into 16 bands using PQMF [55].

2.3.1 Compression Ratio

The compression ratio determines how many audio samples are compressed into a single latent timestep. A higher compression ratio typically makes audio generation more efficient, with latent vectors representing high-level information [26, 56]. The encoder receives a PQMF representation sequence and progressively down-samples it to generate latent timesteps, each one a vector of size 128. The model's compression ratio can be defined as $C_r = N_b \cdot \prod_{j=1}^{M} s_j$, where $N_b \in \mathbb{N}$ is the number of bands in the PQMF representation and $S = \{s_n\}_{n=1}^{M}$; $s_n \in \mathbb{N}$ a sequence of strides along M convolutional blocks in the encoder. RAVE features $N_b = 16$, $M = 4$, and $S = [4, 4, 4, 2]$ and therefore $C_r = 2{,}048$. The decoder receives a latent sequence and upsamples it through M transposed convolution [57], upsampling layers with stride configuration S, interleaved with M residual stacks.

2.3.2 Receptive Field

The encoding and decoding processes have a much larger temporal context than the compression ratio. This context constitutes the *memory* of the system and is denoted *receptive field* [27], which ensures temporal continuity across audio blocks.

The receptive field of RAVE's modules is computed. Using backpropagation, the temporal length of the gradient between inputs and a single output timestep is assessed. The encoder features a receptive field $R_{fe} = 15{,}449$ samples, enabled by the kernel length and striding of its convolutional layers.

On the other hand, RAVE's decoder processes previous latent timesteps due to its dilated residual stacks. Each stack of K residual layers features a dilation configuration $D = \{d_n\}_{n=1}^{K}$; $d_n \in \mathbb{N}$ with residual layers becoming progressively more dilated to increase the number of latent timesteps that are taken into account during the upsampling process. RAVE features $K = 3$ residual layers with dilations $D = [1, 3, 5]$ repeated in $M = 4$ residual blocks for a total receptive field $R_{fd} = 16$ latent timesteps. RAVE's total receptive field (R_f) can be computed by considering a sliding encoder receptive field R_{fe} that spans across R_{fd} latent space timesteps [Eq. (1)]. This accounts for a total of 46,169 audio samples (1.04 s).

$$R_f = R_{fe} + (R_{fd} - 1) * C_r. \tag{1}$$

2.3.3 Waveform Generation

RAVE generates audio samples with a waveform synthesizer and a noise generator. The waveform synthesizer projects the decoder's output and generates a loudness envelope and multiband signal (with *tanh* activation) that is inverted back into 2,048 samples using PQMF. The noise generator is based on a strided convolutional stack that recompresses the decoder's output using a ratio of 1,024 and then performs Fast Fourier transform (FFT) convolution

[2] https://github.com/acids-ircam/RAVE.

on white noise to generate another multiband signal that is summed with the waveform synthesizer before PQMF inversion.

2.3.4 Training

Models are first trained using *representation learning*, which uses a multiresolution spectral reconstruction loss as a metric to learn a suitable compressed representation, and later with an *adversarial fine-tuning* to improve the decoder's audio quality. The noise generator is only active during adversarial fine-tuning. This "slightly increases the reconstruction naturalness of noisy signals" [4, p. 6]. Finally, RAVE can be trained in a *causal* or *noncausal* fashion. Noncausal training allows the model to look ahead half of its receptive field into the future. This increases reconstruction quality at the expense of additional delay. On the other hand, a causal training procedure ensures that the model produces output only by accounting for current and past input. The original RAVE model has 17.6 million trainable parameters.

2.4 Sources of latency in RAVE

This section offers a critical analysis of the latency sources present in RAVE's architecture:

1. **Buffering Delay:** This is determined by two audio blocks of 2,048 samples, equaling a total of 4,096 samples (92.9 ms).
2. **Representation Delay:** The PQMF module features a minimum interband attenuation of 100 dB, implemented with 16 polyphase FIR filters for encoding and 16 for decoding. This requires filter lengths of 513 and 33 for the encoding and decoding process, respectively. This yields a group delay of 256 samples for encoding and 16 samples for decoding. However, each sample at the decoder is then up-sampled by the number of channels. This yields an equivalent of 256 samples of delay at the decoder process, for a total of 512 samples of representation delay (12 ms).
3. **Cumulative Delay:** It manifests when enabling models for streaming that were trained using a noncausal approach such as the original RAVE model. The cached_conv package is employed to measure the theoretical cumulative delay in both encoder and decoder and find that it corresponds to a total of 566 ms at 44.1 kHz, which is enough to cover RAVE's half-second lookahead.
4. **Jitter:** RAVE's encoder acts as a learned feature extractor that processes input audio in a block-by-block fashion. Using sliding blocks would break the temporal coherency of the latent space, which, during training, encodes an input signal without overlap. Because of this, streaming RAVE has to operate with a minimum block size of the compression ratio $C_r = 2{,}048$ samples, which is compressed into a single latent vector timestep of size 128. This can heavily restrict an event's temporal resolution, resulting in jitter when decoding due to the positional uncertainty of the latent representation. For example, consider a trained model that requires an incoming event to be fully captured by the input to decode it. In this case, the jitter could easily cover the range of an audio block, ± 23.2 ms, for a total jitter span of 46.4 ms.

In summary: RAVE's design decisions prioritize reconstruction quality at an elevated compression ratio, which is useful for latent space manipulation and automatic synthesis through prior networks [4].

Considering the goal of interaction with musical instruments, it is unclear what architectural features enable RAVE's timbre transfer capabilities. A valid hypothesis could be that a high compression ratio affords a high-level representation of input, capturing complex temporal structure and allowing the system to learn to track melodies and harmonies. However, RAVE's long receptive field could also aggregate such information.

Interestingly, the features that make this system attractive for automatic music generation make them highly unsuitable for low-latency interaction: a high compression ratio, present in both causal and noncausal versions, introduces unacceptable buffering delay and jitter but enables efficient inference with prior networks, whereas a noncausal mode of operation introduces cumulative delay but improves quality and does not affect offline generation. In the next section, the design goals are reversed, aimed to develop a system for low-latency musical performance.

3 ARCHITECTURE REDESIGN

RAVE's autoencoding architecture works well for the timbre transfer case but cannot support low-latency inference. This section discusses RAVE's design choices and examines their effect on RTF and *sound-to-sound latency*. The latter is closely related to action-to-sound latency, representing the delay between a captured sound-producing action performed on an instrument and the model's resulting output. This analysis is performed by systematically and progressively modifying RAVE's architecture towards a low-latency variant.

Firstly, the datasets used for training the design iterations are presented. Then, an empirical latency and RTF measurement strategy used to guide the design space exploration is introduced. Next, variations of RAVE are trained with different C_r, *PQMF attenuation*, and R_f. This allows for assessing their impact on latency and RTF. This section closes by presenting BRAVE, a model suitable for low-latency sound-to-sound interaction, which achieves adequate latency <10 ms and low total jitter of about 3 ms. Fig. 1 shows its architecture compared to RAVE's.

3.1 Training Details and Datasets

All models are trained in a causal approach to minimize the cumulative delay, following RAVE's two-phase training schedule over a total of 1.5 million steps: 1 million steps of representation learning and 500,000 steps of adversar-

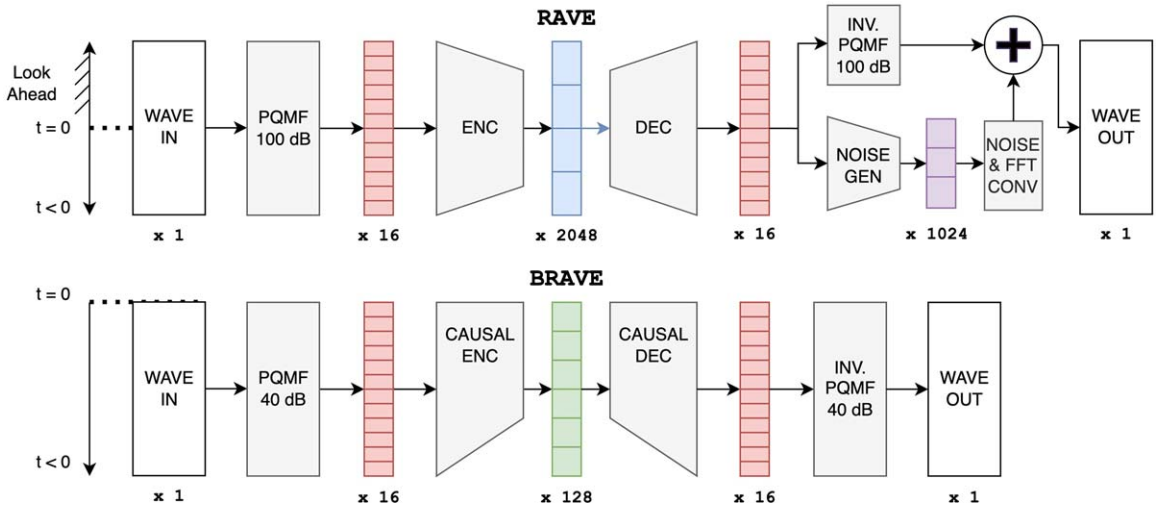


Fig. 1. Simplified architectural comparison. BRAVE achieves adequate latency (<10 ms) and jitter (3 ms) by removing RAVE's noise generator and using a smaller encoder compression ratio, PQMF attenuation, and causal training, reducing its buffering, representation, and cumulative delays respectively. The number of parameters is also reduced to improve its RTF (see Table 2). Numbers in `monospace` denote the compression ratio of intermediate results.

ial training with the original RAVE's adversarial model. Models that encode and decode a single audio channel are trained. Furthermore, RAVE's noise generator is removed from all models except the RAVE reference, because of its independent compression ratio, which should be optimized separately, obscuring the analysis of latency and its sources.

The different variations of RAVE are trained using percussive and harmonic datasets to analyze whether training data affects the models' latency:

- Filosax: An excerpt from the Filosax dataset [58] containing 4.5 h of solo saxophone, extracted from "Participant 1." The last four takes of the same participant are held and used as the test set.
- Drumset: A subset of the Expanded Groove MIDI Dataset [60]. This includes 2 h 50 min of a drummer performing on an electronic drum kit. The selected subset comprises "drummer1" playing the "Acoustic kit," and samples from the denoted training set are used. This dataset includes separated test samples, which are employed for the test set.

The average loudness of each dataset is computed according to ITU-R BS.1770-4 across all instances, obtaining –18.5 and –36.0 LUFS for Filosax and Drumset, respectively. This measurement is used to balance loudness on the test datasets described in SEC. 4.2. All datasets are sampled at 44.1 kHz. No postprocessing is applied to avoid altering the instruments' relationship between timbre and loudness.

3.2 Latency and RTF Evaluation

To guide the design exploration, latency and jitter are empirically measured by testing trained models with a large set of input signals. The system response time is measured as determined by the difference between input and output onset. RTF is also measured by collecting the time taken for multiple forward passes using different block sizes. This evaluation methodology is released as an open-source Python package to support NAS developers (see accompanying website).

3.2.1 Synthetic Excitation Signals

By synthesizing different excitation signals, the authors can precisely control the onset times and generate a wide range of spectral characteristics to assess the models' response under diverse circumstances. White noise is included, a *dense sinusoidal* containing frequencies increasing logarithmically by a half semitone filling a randomly selected bandwidth and a harmonic signal with a random pitch and number of harmonics. All signals are enveloped with an exponential function that decays the amplitude by 60 dB over the duration of the signal.

Once signals are generated, they are front-padded with a random duration of silence $N_s \sim$ ð'° $\{2{,}048, ..., 44{,}100\}$and are back-padded with a second of silence. A full grid search over signal type, length $N_l \in \{4{,}096, 44{,}100\}$ and amplitude $A \in \{0$ dB, -6 dB$\}$ (equaling 12 different signals) was conducted for each model and each test repeated 500 times. All presented results were selected using the signal that leads to the best average latency results for each model and dataset, representing a best-case scenario.

3.2.2 Measuring Latency

The models were set for streaming operation, which introduces cumulative delay. Then, latency is measured as the delay between the start of the excitation signal and the first onset measured in the output signal. The goal is to measure the time required for a model to respond to an input. Two different onset detectors were used, and the first detected onset between the two is reported. The first is a time-domain method, based on the *AmpGate* algorithm implemented in the FluCoMa library [60], which responds quickly to percussive onsets, and the second is the spectral-flux method proposed by [61], which is more robust to different sig-

nal characteristics such as slower attacks observed in wind instruments.

The latency and jitter in samples are obtained and converted to milliseconds using the datasets' sampling rate (44.1 kHz). To this value, the authors add the required buffering delay, equal to two blocks of size equal to the respective compression ratio, therefore showing the minimum sound-to-sound action the model can exhibit. While the error range of the onset detectors may vary depending on the data the models render, impacting the jitter results, the latency test is used as a guide for the designs; even though the informed jitter ranges cannot be guaranteed to be exact, models with better jitter are expected to show so in the results.

3.2.3 Measuring RTF

The RTF measurement process starts by processing the PyTorch models into scripted compute graphs (i.e., the sequence of blocks and layers of a model) using the tools provided in the RAVE repository. The scripting process generates a TorchScript[3] file that can be loaded in Python or C++ using Libtorch, Torch's C++ API. Once loaded, the models are compiled just in time before execution, optimizing the inference process and improving their RTF in comparison with a standard PyTorch inference pass. Scripted models have been integrated into neural audio plugins without additional optimizations [62, 63, 8]. Therefore, this is considered a faithful approximation of performance in a typical real-time use case.

For this process, cached convolutions are enabled in all models to ensure they support streaming operations. However, the latent projection layers the original tool appends to the models' compute graph are skipped because of their additional computational requirements. This does not modify the models' performance and serves for latent space manipulation, which is outside of the scope of this work.

A basic C++ application is designed that loads the scripted models and times their execution time for different audio block sizes. The models are executed 1,100 times, with the first 100 executions employed as a "warm-up" for the operating system scheduler to find an optimal way to handle the inference workload and the rest registered for computing statistics. Then, the application calculates the RTF of each run taking 44.1 kHz as a reference sampling rate and reports the mean and standard deviation. All measurements are performed on a MacBook with an M1 Pro CPU and 16 GB RAM, running MacOS Sonoma 14.3.

3.3 Compression Ratio Latency

This subsection includes empirical analysis of the effects of compression ratio in the model, which directly impacts buffering latency and jitter. RAVE versions that operate at different compression ratios are trained, without altering other model features such as PQMF attenuation, receptive field, and number of parameters.

[3] https://pytorch.org/docs/stable/jit.html.

Table 1. Latency evaluation for different compression ratios. Reported as the best average latency [best jitter] in milliseconds.

	Filosax	Drumset
RAVE v1	244.83 [136.60]	439.89 [84.56]
`c2048_r10`	144.44 [116.28]	130.62 [42.97]
`c1024_r10`	82.56 [47.05]	70.40 [41.36]
`c512_r10`	43.42 [21.16]	39.80 [12.65]
`c256_r10`	30.63 [28.80]	25.77 [5.71]
`c128_r10`	20.50 [8.34]	18.50 [3.27]

The compression ratio of models is reduced by modifying the strides at both the encoder and decoder. However, this also directly affects their receptive field. Since the generator has a fixed receptive field of latent vectors, halving the number of samples encoded in each latent vector halves the encoder's receptive field. The authors compensate for this by increasing the generator's receptive field.

The generator's residual blocks' dilations are modified, manipulating D. Since the residual layers feature a kernel size of 5, the authors chose to exponentially increase the dilation factor by a maximum rate of 3; this avoids holes in the decoder's receptive field. Furthermore, K is also modified in models with small compression ratios, adding more layers to the generator's residual block. This allows for working with residual blocks with higher dilations at a relatively small increase in the total number of parameters. In doing so, models are tested with progressively smaller compression ratios but with similar sizes and receptive fields.

Following this approach, models are designed by progressively halving the original compression ratio down to 128. This last model can operate with a small buffering delay of 5.8 ms at a jitter of 2.9 ms, or $\pm$ 1.45 ms, which the authors deem close enough to their initial latency and jitter targets. However, other sources of latency are still present.

RAVE v1 and all model variations are trained on both datasets. It is observed that all training processes converge similarly. Latency tests are run against all the trained models, and results are shown in Table 1. Please refer to Table 2 for the complete architecture characteristics of all trained models. It is observed that latency improves greatly with causal training and drastically improves with smaller compression ratios. As expected, the model iteration `c128_r10`, with its compression ratio of 128, can provide an acceptable jitter; however, results vary with the training dataset, and all models exceed the 10-ms latency requirement.

3.4 Multiband Decomposition Latency

Having lowered the latency and achieved an acceptable jitter, the authors turn to the PQMF multiband decomposition module to find other sources of latency they can tune. It is possible to reduce the group delay D_g of the filters by relaxing the interband attenuation requirements, obtaining shorter FIR filters with shorter delay. Although greater interband attenuation is typically desired in DSP applications, leading to a better reconstruction with less sub-band

Table 2. Summary of models implemented. All models have a latent vector size of 128. All of them are causal and do not have a noise generator, with exception of RAVE. The receptive field assumes a sample rate of 44.1 kHz.

Model	Hidden Sizes	S: Strides	D: Dilations	PQMF Attenuation (dB)	C_r: Compression Ratios	R_f: Recaptive Field (ms)	No. Parameters (M)
RAVE v1 (noncausal)	[64, 128, 256, 512]	[4, 4, 4, 2]	[1,3,5]	100	2048	1047	17.6
`c2048_r10`	[64, 128, 256, 512]	[4, 4, 4, 2]	[1,3,5]	100	2048	1047	17.5
`c1024_r10`	[64, 128, 256, 512]	[4, 4, 2, 2]	[3,9,27]	100	1024	1070	16.9
`c512_r10`	[64, 128, 256, 512]	[4, 2, 2, 2]	[3,9,18,36]	100	512	960	18.4
`c256_r10`	[64, 128, 256, 512]	[2, 2, 2, 2]	[3,9,27,36]	100	256	973	16.2
`c128_r10`	[64, 128, 256, 512]	[2,2,2,1]	[3,9,27,45,63]	100	128	955	17.3
`c128_r10_p70`	[64, 128, 256, 512]	[2,2,2,1]	[3,9,27,45,63]	70	128	947	17.3
`c128_r10_p40`	[64, 128, 256, 512]	[2,2,2,1]	[3,9,27,45,63]	40	128	941	17.3
`c128_r05_p40`	[64, 128, 256, 512]	[2,2,2,1]	[3,9,27,36]	40	128	517	15.2
BRAVE	[32. 64, 128, 256]	[2,2,2,1]	[3,9,27,36]	40	128	517	4.9

Table 3. Latency evaluation for different PQMF attenuations. All modes have the same C_r. Reported as the best average latency [best jitter] in milliseconds. `c128_r05_p40` and BRAVE are optimized for RTF.

	Attenuation (dB)	Filosax	Drumset
`c128_r10`	100	20.50 [8.34]	18.50 [3.27]
`c128_r10_p70`	70	13.58 [10.86]	13.19 [4.22]
`c128_r10_p40`	40	10.46 [6.30]	9.92 [3.36]
`c128_r05_p40`	40	10.22 [7.57]	9.67 [4.31]
BRAVE	40	10.08 [7.80]	9.75 [2.47]

aliasing, the impact is less clear in NAS where the leakage between sub-bands could be leveraged by the model or compensated for during training.

Variations of `c128_r10` are trained with attenuations of 70 and 40 dB, which are denoted as `c128_r10_p70` and `c128_r10_p40`. They feature PQMF group delays of 256 and 128 samples, respectively, while featuring the same C_r and R_f of `c128_r10`. Their architectural description is shown in Table 2. Next, their latency response is evaluated. Results are shown in Table 3, where a reduction in latency with less PQMF attenuation is observed. The latency target has nearly been reached with the `c128_r10_p40` model, which exhibits latencies around 10 ms, though it shows slightly higher-than-desired jitter values. Nonetheless, these results are considered suitably close to the goal to consider it for real-time implementation.

3.5 Optimizing for RTF

Real-time execution can be exceptionally challenging for low-latency models where inference is performed many times per second (i.e., at a high inference frame rate). This is partially due to the increasing number of scaffolding operations conducted by the inference framework such as memory allocation for inputs, outputs, and intermediate results [64]: a model can exhibit a higher RTF if the inference is computed at a smaller block size. On the other hand, the lower C_r models have a similar number of weights compared to the original RAVE but generate smaller audio blocks at each pass, meaning it is less efficient in terms of computing requirements per sample.

The initial low-latency model, `c128_r10_p40`, cannot run in real time on the reference CPU. The authors set out then to reduce its computational requirements. Firstly, the layers with higher dilation in the generator are removed, effectively halving its receptive field. This model is referred to as `c128_r05_p40`. Next, the hidden sizes of both the encoder and decoder are halved to achieve a model roughly one-third the original size, which can be computed at a suitable RTF. In line with this, the number of channels of its discriminator is also halved. This model is denoted as BRAVE. See Table 2 for architectural details.

The lightweight models' latency is evaluated, shown in Table 3 finding that the modifications do not substantially impact latency. Then, the RTF test is run on these models. Results are shown in Table 4. All models are observed to drastically improve their RTF with larger block sizes. For instance, BRAVE can be run in real time but only at a larger block size of 256, which increases buffering latency. The authors suspect this is due to scaffolding operations within scripted models that require a fixed time, such as memory allocation [65]. This prompts evaluation of an implementation using different tools to avoid that temporal cost.

3.6 Low-Latency Implementation

A custom C++ inference developed by the first author is presented to support the BRAVE autoencoding architecture and some variations. The engine is based on RTNeural [66], a DNN inference library for real-time DNN inference for audio applications such as audio plug-ins. RTNeural is chosen for its design, which transparently supports causal, block-based inference, prompting the designer to think of the model inference process as a streaming operation. Furthermore, it preallocates all memory before inference; this is critical for working at the high frame rates required for low latency. RTNeural is extended to support strided and transposed convolutions and implement all model components.

Three different causal models are implemented in this architecture: a version of RAVE v1 without noise generator (`c2048_r10`), a low-latency and high-capacity model (`c128_r05_p40`), and BRAVE. The fastest implementation is achieved in the reference M1 Pro CPU with the Stan-

Table 4. RTF for selected block sizes shown as mean (standard deviation).

Model	128	256	512	2,048
RAVE v1	...	...	...	0.16 (0.02)
`c2048_r10`	...	...	...	0.15 (0.02)
`c128_r10`	2.62 (0.07)	1.43 (0.03)	0.74 (0.05)	0.24 (0.04)
`c128_r10_p70`	2.61 (0.06)	1.42 (0.03)	0.73 (0.02)	0.24 (0.05)
`c128_r10_p40`	2.60 (0.06)	1.42 (0.03)	0.73 (0.02)	0.24 (0.04)
`c128_r05_p40`	2.20 (0.06)	1.22 (0.03)	0.62 (0.01)	0.20 (0.04)
BRAVE	1.18 (0.07)	0.65 (0.03)	0.37 (0.08)	0.10 (0.003)

dard Template Library backend of RTNeural, using Apple Clang 15. Table 5 compares the RTF between the RTNeural and Libtorch implementations, showing that the latter can execute BRAVE comfortably at the lowest block size and latency, unlike Libtorch. However, it does not improve its RTF with longer block sizes for any of the models, with Libtorch becoming a more efficient option for longer block sizes. It is suspected that Libtorch may employ computing algorithms optimized for different block sizes. This work's models are employed in a proof-of-concept audio plugin (see the accompanying webpage).

Table 6. FAD computed on resynthesis of test set.

	Filosax	Drumset
RAVE v1	43.95	1.48
`c2048_r10`	41.54	1.55
`c128_r10`	6.99	1.19
`c128_r10_p40`	7.58	1.11
`c128_r05_p40`	**6.04**	**1.10**
BRAVE	9.03	2.21
Test Set	*0.22*	*0.28*

Values in bold indicate the model with best FAD for the chosen dataset.

4 EVALUATION

This section sets out to determine whether the design variations can support timbre transfer, testing (1) the models' audio quality, (2) their timbre modification, and (3) their content preservation capabilities, comparing them to RAVE's. Models that progressively accumulate modifications are selected to assess how causality (`c2048_r10`), compression ratio (`c128_r10`), PQMF attenuation (`c128_r10_p40`), receptive field (`c128_r05_p40`), and model capacity (BRAVE) affect timbre transfer.

Firstly, their audio quality is assessed using a simple resynthesis task. Then, the datasets employed are presented as sources for timbre transfer. Next, the authors evaluate whether a high compression ratio is necessary to perform timbre transfer or it can be achieved using a small compression ratio but a sizable receptive field. To this end, the *maximum mean discrepancy* (MMD) test is introduced, similar to that of Bitton et al. [39], designed to quantify differences in timbre distributions between the original and transferred audios. Finally, the authors investigate if working with smaller block sizes can improve the temporal resolution of rendered musical events. The models' capabilities are compared to follow musical dynamics and melodies in timbre transfer by evaluating loudness curves and fundamental frequency rendering.

4.1 Audio Quality Assessment

The widely used Fréchet Audio Distance (FAD) [66] is employed to evaluate audio quality using the VGGish model. The background embeddings are computed using both the Drumset and Filosax training datasets. Evaluation embeddings are computed on audio from their corresponding test datasets, which have been resynthesized by each model variant. The original test set is added as an additional evaluation embedding for reference. FAD is computed between background and evaluation embeddings for each model variant. Results are shown in Table 6.

All the models are observed to score similarly on their respective datasets, which indicates similar audio quality, with a slight degradation in BRAVE possibly due to its smaller capacity. The notable exception is the high compression ratio models trained on Filosax. Such variations in score are suspected to be due to unstable adversarial training, which negatively impacts the capability of the models to replicate pitched sounds. This suggests that such models may require larger datasets to stabilize training, as evi-

Table 5. RTF comparison of Libtorch and RTNeural (denoted with †) models at different block sizes.

Model	128	256	512	2,048
`c2048_r10`	...	...	...	0.15 (0.02)
`c2048_r10`†	...	...	...	0.30 (0.007)
`c128_r05_p40`	2.20 (0.06)	1.22 (0.03)	0.62 (0.01)	0.20 (0.04)
`c128_r05_p40`†	1.02 (0.005)	1.01 (0.010)	1.01 (0.029)	1.00 (0.008)
BRAVE	1.18 (0.07)	0.65 (0.03)	0.37 (0.08)	0.10 (0.003)
BRAVE†	0.29 (0.006)	0.29 (0.005)	0.29 (0.064)	0.29 (0.020)

denced by the size of datasets used in the original RAVE implementation, which are about one order of magnitude bigger than the present one. Please refer to the accompanying website for audio examples.

4.2 Testing Datasets

Audio corpora are selected for testing the timbre transfer capabilities of the models. Different test sets are employed, depending on whether the models were trained on Filosax (melodic test set) or Drumset (percussive test set).

The melodic test set includes "Viola," 26 min of solo viola recordings extracted from the URMP dataset [67], and "Svoice," 16.5 min of female singing—participant 1 and participant 12 [68]. Each track of these datasets is loudness-normalized to –18 LUFS, to match the Filosax dataset loudness.

The percussive test set includes "Candombe," presented in Nunes et al. [69], for which a single recording is selected for each artist totaling 19 min, and "Beatbox," employing the complete Amateur Vocal Percussion Dataset [70] and totaling 18 min. Each track is loudness normalized to –36 LUFS, to match that of the Drumset dataset.

4.3 Timbre Transfer Evaluation

A numerical analysis of timbre transfer performance is sought to compare the capabilities of this model to the original RAVE model. As ground truth audio signals are not available in this evaluation, distributions of timbre features are considered, and timbre transfer is viewed as transforming from one distribution to another [71]. The authors turn to the MMD [72], a two-sample statistical test to determine whether samples are drawn from different distributions, which has been successfully applied to evaluate audio timbre transfer [39].

Given independent samples $\{x_i\}_{i=1}^{n}$ from random variable X following distribution P, $X \sim P$, and $\{y_j\}_{j=1}^{m}$ from $Y \sim Q$, the unbiased empirical estimator of the squared MMD is

$$\begin{aligned}\mathrm{MMD}_u^2[X, Y] = &\frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} k(x_i, x_j) \\ &+ \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \neq i}}^{m} k(y_i, y_j) \\ &- \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} k(x_i, y_j), \end{aligned} \tag{2}$$

where k is a Gaussian kernel $k(x, x') = \exp(-\frac{1}{2\sigma}\|x - x'\|^2)$ and σ is the bandwidth parameter. σ is selected independently for each test as the median L_2 distance between all pairwise sample combinations [72].

Timbre distributions are generated for each audio corpus using Mel-frequency cepstral coefficients (MFCCs). MFCCs are computed with FFTs of size 2,048 with 75% overlap and 128 Mel-bands; MFCCs 2–13 are then selected [46]. Texture windows are created from MFCC frames by average pooling using a window of 40 frames (≈0.5 s) with 50% overlap. The set of texture windows for each audio corpora is used for MMD.

Timbre transfer is performed by running inference over the testing datasets using selected models and storing the results of each model and test set as separate corpora. Furthermore, the held-out Filosax and Drumset sets are kept as references: they are employed as anchors to evaluate target similarity. For each timbre transferred corpus, two MMD distances are calculated: one from the test set (input instrument) and the other from the reference set (target instrument). Additionally, the cross-similarity (distance between the test and reference sets) and the self-similarity (distances obtained by sampling from the same test or reference distributions) are assessed. Successful transfer results should yield MMD scores lower than the cross-similarity, demonstrating that the timbre of the converted audio becomes more aligned with that of the reference dataset. Results are shown in Fig. 2.

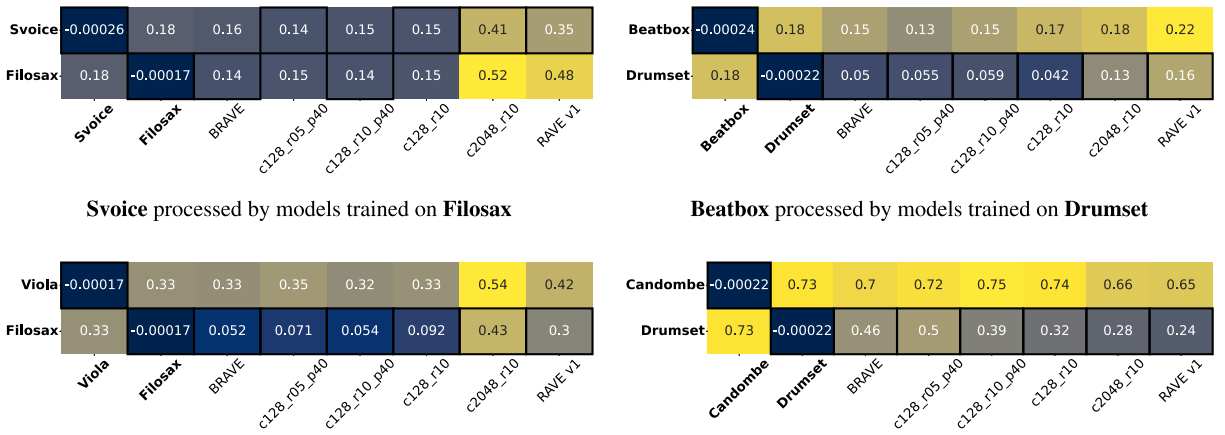
It is observed that the MMD scores do not vary significantly across the models. This indicates that the resulting MFCC distributions are not affected by model causality, PQMF attenuation, or compression ratio, suggesting that the timbre modification capabilities of all models are afforded by their receptive field, which even for BRAVE, covers a sizable 0.5 s. The exceptions are the high compression ratio models trained on Filosax, which can be explained due to their inability to render pitched sounds, as discussed in SEC. 4.1.

4.4 Content Preservation Evaluation

This section presents an analysis of the second requirement for successful timbre transfer: content preservation. A loudness similarity is computed between the test and timbre-transferred corpora, which can account for dynamics, accents, and rhythm information. Additionally, the models' capability of preserving the fundamental frequency of the melodic test sets is evaluated, accounting for pitched and melodic content.

Pitch evaluation utilizes pitch tracking computed on input and transferred audio and evaluates differences using pitch accuracy. Following [73], CREPE [74] is used for pitch tracking, which produces pitch estimates at a frame rate of 200 Hz. Results are filtered using a pitch confidence threshold of 0.85 to account for instabilities in pitch estimates. Accuracy is measured using the overall accuracy metric described by Salamon et al. [75]. This measures the proportion of frames that are accurately labeled as voiced (i.e., have a pitch confidence above the threshold) and are within 0.5 semitones of the input if the frame is voiced. Pitch evaluation results are shown in Table 7.

Loudness error is the L_1 distance between loudness envelopes computed on the input and output audio. Following Hantrakul et al. [73], loudness is calculated by A-weighting the power spectrum, which models the frequency-dependent hearing sensitivities of the human ear by de-emphasizing low and high frequencies. A-weighed power spectrums are then converted to decibel units by log-scaling. Results are computed using an FFT size of 2,048



Svoice processed by models trained on **Filosax**

Beatbox processed by models trained on **Drumset**

Viola processed by models trained on **Filosax**

Candombe processed by models trained on **Drumset**

Fig. 2. The MMD distance between the timbre transfer results of each model and the input (top row) and target instrument (bottom row) datasets are computed. Along the horizontal, entries in bold show dataset cross-similarity and self-similarity. Following this, MMD results for timbre transfer of all model variants. Lower values for the target instrument indicate closer alignment to the target distribution and therefore, a more successful timbre transfer.

Table 7. Pitch accuracy evaluated on melodic test sets. Higher values are better.

	Svoice	Viola
RAVE v1	0.29	0.53
`c2048_r10`	0.28	0.53
`c128_r10`	0.64	0.65
`c128_r10_p40`	0.62	0.63
`c128_r05_p40`	0.63	0.67
BRAVE	**0.68**	**0.69**

Values in bold indicate the model with best pitch content preservation.

Table 8. Loudness L_1 evaluated on both the percussive and melodic test sets. Lower values are better.

	Beatbox	Candombe	Svoice	Viola
RAVE v1	14.24	13.75	33.63	24.69
`c2048_r10`	13.28	8.21	34.24	31.83
`c128_r10`	9.84	4.09	12.65	9.02
`c128_r10_p40`	10.97	4.08	13.68	10.02
`c128_r05_p40`	7.99	**4.04**	12.73	7.66
BRAVE	**7.89**	4.31	**9.94**	**7.63**

Values in bold indicate the model exhibiting the best loudness preservation for each dataset.

samples and a hop size of 256 samples. Loudness L_1 distance results are shown in Table 8.

It is observed that all content metrics improve with a smaller compression ratio. This is attributed to the higher temporal resolution afforded by all models of $C_r = 128$, which permits them to follow closely fast-varying input signal characteristics related to pitch variations and accents. Interestingly, the BRAVE model outperforms their higher-capacity counterparts on many test sets, with minimal audio quality loss. Being a lower-capacity model, it is suspected to learn to rely much more on the encoder to generate plausible output, allowing it to relay more content to the timbre-transferred output.

4.5 A Comment on the Interaction Capabilities of BRAVE

Several BRAVE models are deployed on the demo plugin, which can be downloaded from the accompanying webpage. They are played using different instruments such as guitar, congas, and voice. The models demonstrate fast response on transients, resulting in very low perceived latency. However, in models like Drumset, the transients appear a bit soft or smeared. Despite this, they perform well in percussive transformations. For pitched audio, the results are mixed. While the Filosax model also exhibits low latency, it often struggles with correct pitch rendering, which is essential for supporting natural interaction with many instruments. In contrast, other models tested show better performance in this regard; further investigation on pitch performance is left for future work.

4.6 Discussion of Results

4.6.1 Data-Dependent Latency and Jitter

In NAS systems that leverage convolutional layers, the entire system can be viewed as a complex, nonlinear filter where filter coefficients are learned through a data-driven process. The response of this system, and thereby the overall group delay, is characterized by the underlying convolutional filters. While the analytic determination of the group delay in such systems is complicated due to nonlinearities, it was empirically observed that the nature of the data used to train these models had an impact on latency and jitter, consistently measuring lower latency and jitter with the Drumset dataset as compared to the Filosax. This suggests that signals with strong transients resulted in the models prioritizing temporal accuracy, which contributes heavily to the error in the multiresolution spectral loss in bands with higher temporal resolution. Some variation of the jitter in the results could be attributed to a different error range of the onset detector between datasets.

Overall, the authors note that there is a complex relationship between the training data, loss functions, and model latency, which they highlight as an area for future investigation. For example, time-domain audio loss functions may be incorporated to improve temporal precision and phase reconstruction of the model [76]; however, this may be incompatible with the VAE objective.

4.6.2 Timbre Transfer Affordances

The timbre transfer test results indicate that timbre modification can be afforded by a large receptive field, evidenced by the similar MMD score between BRAVE and RAVE. Furthermore, content rendering improves when working at smaller compression ratios. This is in line with the typical timbre transfer design assumptions that assume that musical content varies over time while timbre is a global attribute. Finally, it is surprising to see similar performance between this lower-capacity BRAVE model and its low-compression ratio counterparts. This suggests that (1) further performance gains can be obtained by a careful revision of BRAVE's structure toward a smaller model and (2) there may be potential for increasing audio fidelity in bigger models.

4.6.3 Design Challenges for Low-Latency NAS

Designing NAS algorithms for low latency requires putting this issue at the center of the design problem and carefully inspecting the audio representations and the architecture to ensure these do not increase response time. It is found that a representation learned with a VAE, rendered as a *time-varying trajectory* progressively constructed over short audio windows in a temporally causal manner by a compressing encoder and aggregated by a decoder with a considerable receptive field, works well for this timbre transfer task.

The authors believe this approach can work for other low-latency interactive NAS tasks, provided all modules within the network are designed to support such latent trajectories. For instance, a low-latency redesign of RAVE's noise generator for low compression ratio operation could improve BRAVE's audio quality. One possibility would be the addition of low-latency, differentiable infinite impulse response filters [77] controlled by an additional decoder submodule with an appropriate receptive field. The authors expect their design experience can guide other researchers in the design of complex low-latency NAS algorithms, while leveraging recent inference tools that can simplify implementation [65, 64].

5 CONCLUSION

This paper advocated for designing interactive NAS models from the ground up, considering latency at each step. To support this design process, an overview of common sources of latency and jitter was presented, along with a method for empirically measuring it. It was found that RAVE, a model with extensive interactive applications, incurs a level of latency that makes it unsuitable for audio-driven interaction with traditional musical instruments.

The redesign process puts latency at the center of the problem and allows for designing architectures that are suitable for real-time interaction with musical instruments. This can enable timbre transfer in use cases that as of today have been relegated to production environments or specially designed musical interfaces.

The low-latency timbre transfer system holds the potential to foster intimate control of instruments and interfaces with an extended timbral palette. However, its data-dependent performance may limit interaction possibilities, and further work should look into how its affordances are affected by data and its corresponding latent representations.

6 ACKNOWLEDGMENT

This work is supported by the Centre for Doctoral Training in Artificial Intelligence and Music, Engineering and Physical Sciences Research Council, UK Research and Innovation (EP/S022694/1). This research utilized Queen Mary's Apocrita High-Performance Computing facility, supported by Queen Mary University of London Research-IT (http://doi.org/10.5281/zenodo.438045).

7 REFERENCES

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

[2] G. Narita, J. Shimizu, and T. Akama, "GANStrument: Adversarial Instrument Sound Synthesis With Pitch-Invariant Instance Conditioning," *arXiv preprint arXiv:2211.05385* (2023 Mar.).

[3] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable Digital Signal Processing," in *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, pp. 1210–1228 (Addis Ababa, Ethiopia) (2020 Apr.).

[4] A. Caillon and P. Esling, "RAVE: A Variational Autoencoder for Fast and High-Quality Neural Audio Synthesis," *arXiv preprint arXiv:2111:05011* (2021 Dec.).

[5] N. Privato, T. Magnusson, and E. T. Einarsson, "Magnetic Interactions as a Somatosensory Interface," in *Proceedings of the 23rd International Conference on New Interfaces for Musical Expression (NIME)*, paper 3 (Mexico City, Mexico) (2023 May).

[6] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv preprint arXiv:1312.6114* (2014 May).

[7] H. Scurto and L. Postel, "Soundwalking Deep Latent Spaces," in *Proceedings of the 23rd International Conference on New Interfaces for Musical Expression (NIME)*, paper 23 (Mexico City, Mexico) (2023 May).

[8] A. Caillon and P. Esling, "Streamable Neural Audio Synthesis With Non-Causal Convolutions," *arXiv preprint arXiv:2204.07064* (2022 Apr.).

[9] A. Schmid, M. Ambros, J. Bogon, and R. Wimmer, "Measuring the Just Noticeable Difference for Audio Latency," in *Proceedings of the 19th International Audio Mostly Conference: Explorations in Sonic Cultures*, pp. 325–331 (Milan, Italy) (2024 Sep.). https://doi.org/10.1145/3678299.3678331.

[10] F. Morreale, A. Guidi, and A. McPherson, "Magpick: An Augmented Guitar Pick for Nuanced Control," in *Proceedings of the 19th Conference on New Interfaces for Musical Expression (NIME)*, pp. 65–70 (Porto Alegre, Brazil) (2019 Jun.).

[11] R. H. Jack, T. Stockman, and A. McPherson, "Effect of Latency on Performer Interaction and Subjective Quality Assessment of a Digital Musical Instrument," in *Proceedings of the 16th International Audio Mostly Conference (AM)*, pp. 116–123 (Norrköping, Sweden) (2016 Oct.). https://doi.org/10.1145/2986416.2986428.

[12] A. P. McPherson, R. H. Jack, and G. Moro, "Action-Sound Latency: Are Our Tools Fast Enough?" in *Proceedings of the 16th International Conference on New Interfaces for Musical Expression (NIME)*, pp. 20–25 (Brisbane, Australia) (2016 Jul.).

[13] D. Wessel and M. Wright, "Problems and Prospects for Intimate Musical Control of Computers," *Comput. Music J.*, vol. 26, no. 3, pp. 11–22 (2002 Sep.). https://doi.org/10.1162/014892602320582945.

[14] M. Lester and J. Boley, "The Effects of Latency on Live Sound Monitoring," presented at the *123rd Convention of the Audio Engineering Society* (2007 Oct.), paper 7198.

[15] Y. Wu, E. Manilow, Y. Deng, et al., "MIDI-DDSP: Detailed Control of Musical Performance via Hierarchical Modeling," presented at the *International Conference on Learning Representations (ICLR)* (Online) (2022 Apr.).

[16] M. Pasini and J. Schlüter, "Musika! Fast Infinite Waveform Music Generation," in *Proceedings of the 23rd International Society for Music Information Retrieval Conference (ISMIR)*, pp. 543–550 (Bengaluru, India) (2022 Dec.).

[17] G. Vigliensoni and R. Fiebrink, "Steering Latent Audio Models Through Interactive Machine Learning," in *Proceedings of the 14th International Conference on Computational Creativity (ICCC)*, pp. 393–397 (Waterloo, Canada) (2023 Jun.).

[18] P. Esling, A. Chemla–Romeu-Santos, and A. Bitton, "Generative Timbre Spaces: Regularizing Variational Auto-Encoders With Perceptual Metrics," *arXiv preprint arXiv:1805.08501* (2018 Oct.).

[19] M. Pasini, M. Grachten, and S. Lattner, "Bass Accompaniment Generation Via Latent Diffusion," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1166–1170 (Seoul, South Korea) (2024 Apr.). https://doi.org/10.1109/ICASSP48485.2024.10446400.

[20] A. Carson, A. Wright, J. Chowdhury, V. Välimäki, and S. Bilbao, "Sample Rate Independent Recurrent Neural Networks for Audio Effects Processing," in *Proceedings of the 27th International Conference on Digital Audio Effects (DAFx)*, pp. 17–24 (Guildford, UK) (2024 Sep.).

[21] F. Caspe, A. McPherson, and M. Sandler, "FM Tone Transfer With Envelope Learning," in *Proceedings of the 18th International Audio Mostly Conference (AM)*, pp. 116–123 (Edinburgh, UK) (2023 Oct.). https://doi.org/10.1145/3616195.3616196.

[22] J. Shier, C. Saitis, A. Robertson, and A. McPherson, "Real-Time Timbre Remapping With Differentiable DSP," in *Proceedings of the 24th International Conference on New Interfaces for Musical Expression (NIME)*, pp. 377–385 (Utrecht, Netherlands) (2024).

[23] A. Martelloni, A. P. McPherson, and M. Barthet, "Real-Time Percussive Technique Recognition and Embedding Learning for the Acoustic Guitar," in *Proceedings of the 24th International Society for Music Information Retrieval Conference (ISMIR)*, pp. 121–128 (Milan, Italy) (2023 Nov.).

[24] N. Privato, G. Lepri, T. Magnusson, and E. T. Einarsson, "Sketching Magnetic Interactions for Neural Synthesis," in *Proceedings of the International Conference on Technologies for Music Notation and Representation (TENOR)*, pp. 89–97 (Zurich, Switzerland) (2024 Apr.).

[25] O. Rybakov, N. Kononenko, N. Subrahmanya, M. Visontai, and S. Laurenzo, "Streaming Keyword Spotting on Mobile Devices," in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 2277–2281 (Shanghai, China) (2020 Oct.). https://doi.org/10.21437/Interspeech.2020-1003.

[26] A. Défossez, J. Copet, G. Synnaeve, and Y. Adi, "High Fidelity Neural Audio Compression," *arXiv preprint arXiv:2210.13438* (2022 Oct.). https://doi.org/10.48550/arXiv.2210.13438.

[27] C. J. Steinmetz and J. D. Reiss, "Efficient Neural Networks for Real-Time Analog Audio Effect Modeling," *arXiv preprint arXiv:2102.06200* (2021 Feb.).

[28] M. Radfar, P. Lyskawa, B. Trujillo, et al., "Conmer: Streaming Conformer Without Self-Attention for Interactive Voice Assistants," in *Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 2198–2202 (Dublin, Ireland) (2023 Aug.). https://doi.org/10.21437/Interspeech.2023-2228.

[29] A. Défossez, L. Mazaré, M. Orsini, et al., "Moshi: A Speech-Text Foundation Model for Real-Time Dialogue," *arXiv preprint arXiv:2410.00037* (2024 Oct.).

[30] K. Siedenburg and S. McAdams, "Four Distinctions for the Auditory 'Wastebasket' of Timbre," *Front. Psychol.*, vol. 8, paper 1747 (2017 Oct.). https://doi.org/10.3389/fpsyg.2017.01747.

[31] ANSI, "American National Standard Psychoacoustical Terminology," *Standard S3.20-1973* (1973 Dec.).

[32] 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.

[33] M. Carney, C. Li, E. Toh, N. Zada, P. Yu, and J. Engel, "Tone Transfer: In-Browser Interactive Neural Audio Synthesis," presented at the *ACM Conference on Intelligent*

User Interfaces (ACM IUI): 2nd Workshop on Human-AI Co-Creation With Generative Models (HAI-GEN) (Online) (2021 Apr.).

[34] Y.-J. Luo, K. W. Cheuk, W. Choi, et al., "DisMix: Disentangling Mixtures of Musical Instruments for Source-Level Pitch and Timbre Manipulation," *arXiv preprint arXiv:2408.10807* (2024 Aug.).

[35] J. Engel, C. Resnick, A. Roberts, et al., "Neural Audio Synthesis of Musical Notes With WaveNet Autoencoders," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 1068–1077 (Sydney, Australia) (2017 Jul.).

[36] N. Mor, L. Wolf, A. Polyak, and Y. Taigman, "A Universal Music Translation Network," *arXiv preprint arXiv:1805.07848* (2018 May).

[37] M. Alinoori and V. Tzerpos, "Music-STAR: A Style Translation System for Audio-Based Re-Instrumentation," in *Proceedings of the 23rd International Society for Music Information Retrieval Conference (ISMIR)*, pp. 419–426 (Bengaluru, India) (2022 Dec.).

[38] L. A. Gatys, A. S. Ecker, and M. Bethge, "A Neural Algorithm of Artistic Style," *arXiv preprint arXiv:1508.06576* (2015 Sep.).

[39] A. Bitton, P. Esling, and A. Chemla-Romeu-Santos, "Modulated Variational Auto-Encoders for Many-to-Many Musical Timbre Transfer," *arXiv preprint arXiv:1810.00222* (2018 Sep.).

[40] R. S. Bonnici, M. Benning, and C. Saitis, "Timbre Transfer With Variational Auto Encoding and Cycle-Consistent Adversarial Networks," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8 (Padua, Italy) (2022 Jul.). https://doi.org/10.1109/IJCNN55064.2022.9892107.

[41] Y. Wu, Y. He, X. Liu, Y. Wang, and R. B. Dannenberg, "Transplayer: Timbre Style Transfer With Flexible Timbre Control," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5 (Rhodes Island, Greece) (2023 Jun.). https://doi.org/10.1109/ICASSP49357.2023.10096233.

[42] S. Huang, Q. Li, C. Anil, et al., "TimbreTron: A WaveNet(CycleGAN(CQT(Audio))) Pipeline for Musical Timbre Transfer," in *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 1508–1524 (New Orleans, LA) (2019 May).

[43] C.-Y. Lu, M.-X. Xue, C.-C. Chang, C.-R. Lee, and L. Su, "Play as You Like: Timbre-Enhanced Multi-Modal Music Style Transfer," *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 1, pp. 1061–1068 (2019 Jul.). https://doi.org/10.1609/aaai.v33i01.33011061.

[44] D. K. Jain, A. Kumar, L. Cai, S. Singhal, and V. Kumar, "ATT: Attention-Based Timbre Transfer," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6 (Glasgow, UK) (2020 Jul.). https://doi.org/10.1109/IJCNN48605.2020.9207146.

[45] A. Bitton, P. Esling, and T. Harada, "Vector-Quantized Timbre Representation," *arXiv preprint arXiv:2007.06349* (2020 Jul.).

[46] O. Cífka, A. Ozerov, U. Şimşekli, and G. Richard, "Self-Supervised VQ-VAE for One-Shot Music Style Transfer," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 96–100 (Toronto, Canada) (2021 Jun.). https://doi.org/10.1109/ICASSP39728.2021.9414235.

[47] L. Comanducci, F. Antonacci, and A. Sarti, "Timbre Transfer Using Image-to-Image Denoising Diffusion Implicit Models," *arXiv preprint arXiv:2307.04586* (2023 Jul.). https://doi.org/10.48550/arXiv.2307.04586.

[48] V. Popov, A. Amatov, M. Kudinov, V. Gogoryan, T. Sadekova, and I. Vovk, "Optimal Transport in Diffusion Modeling for Conversion Tasks in Audio Domain," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5 (Rhodes Island, Greece) (2023 Jun.). https://doi.org/10.1109/ICASSP49357.2023.10094854.

[49] M. Mancusi, Y. Halychanskyi, K. W. Cheuk, et al., "Latent Diffusion Bridges for Unsupervised Musical Audio Timbre Transfer," *arXiv preprint arXiv:2409.06096* (2024 Sep.).

[50] T. Kaneko, K. Tanaka, H. Kameoka, and S. Seki, "iSTFTNet: Fast and Lightweight Mel-Spectrogram Vocoder Incorporating Inverse Short-Time Fourier Transform," *arXiv preprint arXiv:2203.02395* (2022 Mar.).

[51] D.-Y. Wu1, 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 (ISMIR)*, pp. 76–83 (Bengaluru, India) (2022 Dec.).

[52] B. Chandra Garai, P. Das, and A. K. Mishra, "Group Delay Reduction in FIR Digital Filters," *Signal Process.*, vol. 91, no. 8, pp. 1812–1825 (2011 Aug.). https://doi.org/10.1016/j.sigpro.2011.02.005.

[53] A. De Cheveigné and H. Kawahara, "YIN, a Fundamental Frequency Estimator for Speech and Music," *J. Acoust. Soc. Am.*, vol. 111, no. 4, pp. 1917–1930 (2002 Apr.).

[54] J. Armitage, N. Privato, V. Shepardson, and C. B. Gutierrez, "Explainable AI in Music Performance: Case Studies From Live Coding and Sound Spatialisation," presented at the *37th Annual Conference on Neural Information Processing Systems (NeurIPS): XAI in Action: Past, Present, and Future Applications Workshop* (New Orleans, LA) (2023 Dec.).

[55] T. Nguyen, "Near-Perfect-Reconstruction Pseudo-QMF Banks," *IEEE Trans. Signal Process.*, vol. 42, no. 1, pp. 65–76 (1994 Jan.). https://doi.org/10.1109/78.258122.

[56] M. Pasini, S. Lattner, and G. Fazekas, "Music2Latent: Consistency Autoencoders for Latent Audio Compression," *arXiv preprint arXiv:2408.06500* (2024 Aug.). https://doi.org/10.48550/arXiv.2408.06500.

[57] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional Networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2528–2535 (San Francisco, CA) (2010 Jun.). https://doi.org/10.1109/CVPR.2010.5539957.

[58] D. Foster and S. Dixon, "Filosax: A Dataset of Annotated Jazz Saxophone Recordings," in *Proceedings of the*

22nd International Society for Music Information Retrieval Conference (ISMIR), pp. 205–212 (Online) (2021 Nov.).

[59] L. Callender, C. Hawthorne, and J. Engel, "Improving Perceptual Quality of Drum Transcription With the Expanded Groove MIDI Dataset," *arXiv preprint arXiv:2004.00188* (2020 Dec.).

[60] P. A. Tremblay, O. Green, G. Roma, et al., "Fluid Corpus Manipulation Toolbox," *Zenodo* (2022 Jul.). https://doi.org/10.5281/zenodo.6834643.

[61] S. Böck and G. Widmer, "Maximum Filter Vibrato Suppression for Onset Detection," in *Proceedings of the 16th International Conference on Digital Audio Effects (DAFx)*, paper 12 (Maynooth, Ireland) (2013 Sep.).

[62] R. Diaz, C. Saitis, and M. Sandler, "Interactive Neural Resonators," *arXiv preprint arXiv:2305.14867* (2023 May).

[63] F. Caspe, A. McPherson, and M. Sandler, "FM Tone Transfer With Envelope Learning," in *Proceedings of the 18th International Audio Mostly Conference (AM)*, pp. 116–123 (Edinburgh, UK) (2023 Aug.). https://doi.org/10.1145/3616195.3616196.

[64] J. Chowdhury, "RTNeural: Fast Neural Inferencing for Real-Time Systems," *arXiv preprint arXiv:2106.03037* (2021 Jun.).

[65] V. Ackva and F. Schulz, "ANIRA: An Architecture for Neural Network Inference in Real-Time Audio Applications," in *Proceedings of the IEEE 5th International Symposium on the Internet of Sounds (IS2)*, pp. 1–10 (Erlangen, Germany) (2024 Sep.). https://doi.org/10.1109/IS262782.2024.10704099.

[66] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Frèchet Audio Distance: A Metric for Evaluating Music Enhancement Algorithms," arXiv preprint arXiv:1812.08466 (2019 Jan.).

[67] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, "Creating A Multi-Track Classical Musical 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.

[68] D. A. Black, M. Li, and M. Tian, "Automatic Identification of Emotional Cues in Chinese Opera Singing," in *Proceedings of 13th International Conference on Music Perception and Cognition and the 5th Conference for the Asian-Pacific Society for Cognitive Sciences of Music (ICMPC-APSCOM)*, pp. 250–255 (Seoul, South Korea) (2014 Aug.).

[69] L. Nunes, M. Rocamora, L. Jure, and L. W. P. Biscainho, "Beat and Downbeat Tracking Based on Rhythmic Patterns Applied to the Uruguayan Candombe Drumming," in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pp. 264–270 (Málaga, Spain) (2015 Oct.).

[70] A. Delgado, "Amateur Vocal Percussion Dataset," *Zenodo* (2019 Jun.). https://doi.org/10.5281/zenodo.3245959.

[71] E. Moliner, S. Braun, and H. Gamper, "Gaussian Flow Bridges for Audio Domain Transfer With Unpaired Data," *arXiv preprint arXiv:2405.19497* (2024 May).

[72] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A Kernel Two-Sample Test," *J. Mach. Learn. Res.*, vol. 13, no. 25, pp. 723–773 (2012 Mar.).

[73] L. Hantrakul, J. Engel, A. Roberts, and C. Gu, "Fast and Flexible Neural Audio Synthesis," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, pp. 524–530 (Delft, Netherlands) (2019 Nov.).

[74] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "CREPE: A Convolutional Representation for Pitch Estimation," *arXiv preprint arXiv:1802.06182* (2018 Feb.).

[75] J. Salamon, E. Gómez, D. P. Ellis, and G. Richard, "Melody Extraction From Polyphonic Music Signals: Approaches, Applications, and Challenges," *IEEE Signal Process. Mag.*, vol. 31, no. 2, pp. 118–134 (2014 Mar.).

[76] J. J. Webber, C. Valentini-Botinhao, E. Williams, G. E. Henter, and S. King, "Autovocoder: Fast Waveform Generation from a Learned Speech Representation Using Differentiable Digital Signal Processing," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5 (Rhodes Island, Greece) (2023 Jun.). https://doi.org/10.1109/ICASSP49357.2023.10095729.

[77] C.-Y. Yu, C. Mitcheltree, A. Carson, et al., "Differentiable All-Pole Filters for Time-Varying Audio Systems," in *Proceedings of the 27th International Conference on Digital Audio Effects (DAFx)*, pp. 345–352 (Guildford, UK) (2024 Sep.).

THE AUTHORS


Franco Caspe


Jordie Shier


Mark Sandler


Charalampos Saitis


Andrew McPherson

Franco Caspe is a Ph.D. candidate 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, using deep learning as an analysis tool to capture performance features from instrument audio and as a generation tool for synthetic sound rendering. He holds an M.Sc. in computer vision and a degree in electronic engineering. Previously, he worked on real-time systems for image processing and communications. His expertise spans signal processing, machine learning, and audio, focusing on real-time applications and performance-driven synthesis.

•

Jordie Shier is a third year Ph.D. student in the Artificial Intelligence and Music (AIM) program based at Queen Mary University of London, studying under the supervision of Prof. Andrew McPherson and Dr. Charalampos Saitis. His research is focused on the development of novel methods for synthesizing audio and the creation of new interfaces for interacting with music synthesizers. Prior to his doctoral studies, Jordie studied computer science and music at the University of Victoria in Canada, cofounded a drum education start-up, and performed in the electronic music duo Napoleon Skywalker.

•

Mark Sandler, Fellow of the Royal Academy of Engineering, is Director of the Centre for Digital Music at Queen Mary University of London, where he is a professor of signal processing. He has been active in digital audio and music research since the late 1970s when he started his Ph.D. in digital audio power amplifiers. Since then, he has published well over 500 papers across many topics in digital audio and computer music. Sound Synthesis has been a particular interest of his since the 1980s, when he worked with the pioneering British drum synthesizer company Simmons Electronics. He has been active in artificial intelligence for music and audio since about 2014. He has won more than £20 million funding as principal investigator, published more than 500 papers, and supervised more than 50 Ph.D. and 20 postdoctoral research assistants.

•

Charalampos Saitis is a lecturer (assistant professor) at the Centre for Digital Music of Queen Mary University of London. He studied mathematics and computational music acoustics in Athens and Belfast and obtained a Ph.D. in music technology from McGill University. He is an investigator of UK Research and Innovation's £6.5M Centre for Doctoral Training in Artificial Intelligence and Music and founding codirector of the International Conference on Timbre. He has authored several recent publications in the intersecting fields of cognitive science, musical acoustics, music informatics, and machine listening.

•

Andrew McPherson is a computing researcher, composer, electronic engineer, and musical instrument designer. He is a professor of design engineering and music in the Dyson School of Design Engineering, Imperial College London, where he leads the Augmented Instruments Laboratory. Andrew holds undergraduate degrees in both engineering and music from MIT, an M.Eng. in electrical engineering from MIT, and a Ph.D. in music composition from the University of Pennsylvania. Prior to joining Imperial in 2023, he was a professor in the Centre for Digital Music at Queen Mary University of London.