


EMPIRICAL RESEARCH

Open Access



Efficient bandwidth extension of musical signals using a differentiable harmonic plus noise model

Pierre-Amaury Grumiaux^{1†} and Mathieu Lagrange^{1*†} 

Abstract

The task of bandwidth extension addresses the generation of missing high frequencies of audio signals based on knowledge of the low-frequency part of the sound. This task applies to various problems, such as audio coding or audio restoration. In this article, we focus on efficient bandwidth extension of monophonic and polyphonic musical signals using a differentiable digital signal processing (DDSP) model. Such a model is composed of a neural network part with relatively few parameters trained to infer the parameters of a differentiable digital signal processing model, which efficiently generates the output full-band audio signal.

We first address bandwidth extension of monophonic signals, and then propose two methods to explicitly handle polyphonic signals. The benefits of the proposed models are first demonstrated on monophonic and polyphonic synthetic data against a baseline and a deep-learning-based ResNet model. The models are next evaluated on recorded monophonic and polyphonic data, for a wide variety of instruments and musical genres. We show that all proposed models surpass a higher complexity deep learning model for an objective metric computed in the frequency domain. A MUSHRA listening test confirms the superiority of the proposed approach in terms of perceptual quality.

Keywords Audio restoration, Deep learning, Differentiable sound model

1 Introduction

Audio bandwidth extension (BWE) is a subtask of audio enhancement [1] whose goal is to extrapolate the audio spectrum to higher frequencies, in contrast with *audio inpainting* whose goal is to interpolate missing parts [2]. BWE has been considered early in telecommunication systems to overcome bandwidth limitations, especially in telephony for which the typical sampling rate is 16 kHz, i.e., leading to the highest frequency in the processed signal be 8 kHz. In the case of human conversations, the

quality of speech can be greatly improved if the sampling rate is increased to 44.1 or 48 kHz [3]. In the same vein, another application of BWE is to improve the quality of old music recordings, possibly in addition to the removal of clicks and noise [4] or declipping [5]. In both applications, the signal enhancement is handled without access to the original signal with better quality. Informed BWE algorithms can also be useful in audio coding [6] where signals of smaller sampling rates are more effectively compressed, requiring the use of a BWE to restore the full sampling rate of the decoded signal. In most cases, low bitrate side information is transmitted along the compressed low-frequency signal to improve the performance of the BWE module.

Finally, BWE is also meaningful for interoperability of audio processing tools as many audio signal processing methods, such as source separation [1], speech synthesis [7], or voice conversion [8], focus on 16-kHz signals,

[†]Pierre-Amaury Grumiaux and Mathieu Lagrange contributed equally to this work.

*Correspondence:

Mathieu Lagrange
mathieu.lagrange@ls2n.fr

¹ Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, 1, rue de la Noë, F-44000 Nantes, France

hence the need for a BWE system beforehand if the acquired signal is not at the desired sampling rate.

Even though many deep learning-based systems have been proposed to tackle BWE, most of them do not consider runtime efficiency as critical, leading to high-quality systems that can be very costly at inference. High-quality generators based on autoregressive signal models such as Wavenet [9] or on diffusion [10] have intrinsic high complexity and sequentiality which limit their use for time or delay critical applications.

In this paper, we propose to consider differentiable digital signal processing (DDSP) models derived from the seminal work of [11] in order to tackle BWE in an efficient manner. Controlling a harmonic plus noise sound model with a deep learning architecture allows us to considerably reduce inference time. Experiments described in this paper demonstrate a large speed increase compared to a reference ResNet implementation [12], this with better resulting perceptual quality. This is due to several factors, including the reduction of learnable parameters. Using the DDSP approach, the sound is generated using deterministic synthesizers that are controlled by several deep-learning modules of relatively small sizes. For comparison, the ResNet architecture has more than 55k learnable parameters, while the tested DDSP approach has around 4k parameters.

The remaining of this article is organized as follows. In Section 2, we present a general overview of works related to BWE. In Section 3, we explain the proposed models designed to address BWE. The experimental protocol whose code is available online¹, and which rely on publicly available datasets, is detailed in Section 4. In Section 5, we show how the proposed models are well designed when considering synthetic data, and in Section 6 they are evaluated on real data. Finally, we conclude this article in Section 7.

2 Related work

Most approaches considered speech signals with application to telephony. The literature that consider music or general audio is more scarce. When available, we here put the focus on literature related to musical audio.

2.1 Signal processing approaches

Early works employed pure signal processing methods for BWE. In the area of audio coders, some non-blind systems rely on spectral band replication (SBR) [6] using side information extracted during compression. The SBR algorithm is based on the replication of the low-band spectrum to the high-band region, possibly with the

benefits of side information about the high frequencies to improve the overall performance. It has been extended in several works [13, 14], e.g., by replacing the replication by a stretching of the low-band content towards the high-band part, thus preserving the intrinsic harmonic relationships. Source-filter models have been also employed to extend the bandwidth using line spectral frequencies in [15]. Systems based on dictionary learning to map low-frequency patterns to high-frequency components have been proposed in [16, 17]. Classic machine learning methods have also been explored for BWE, such as Gaussian mixture models (GMMs) [18], hidden Markov models (HMM) [19, 20], or non-negative matrix factorization (NMF) [21, 22].

2.2 Convolutional deep learning approaches

Recently, deep learning (DL) methods have shown great performance to synthesize the upper band spectrum. The first works that apply DL technique in BWE literature used deep neural networks (DNNs) with dense layers to infer the high frequencies up to 8 kHz [23–25]. In [23], the log short-time Fourier transform is fed into several dense layers with the last one inferring the high-band spectrum magnitude. The waveform is reconstructed by using the flipped phase from the low-band to estimate the high-band phase information. While this flipped method avoids having phase discontinuities at the low/high frontier, [24] proposes to cope with this potential issue by extended the mean-squared error (MSE) loss function with a regularization term. Gaussian-Bernoulli restricted Boltzmann machines (GBRBM) has been employed alongside dense layers in [25] in order to estimate the higher spectral envelope. Other systems make use of convolutional neural networks (CNNs) to infer the high frequencies from the low-band input features, using 1D convolutions in the time domain [12, 26–28] or 2D convolutions in the spectro/temporal domain [29, 30]. In [26], the authors show that using a network architecture of 1D dilated convolutions and residual connections outperforms a state-of-the-art based on a long short-term memory (LSTM) system on speech signals. The authors of [27] make use of 1D convolutional layers in an encoder-decoder scheme to extend the bandwidth of speech and musical signals in three upscaling ratios: 2, 4, and 6. They show the effectiveness of their system compared to [24] for objective and perceptive metrics. In the same vein, an encoder-decoder architecture in the time domain is also used in [28], but the authors propose to opt for subpixel layers instead of classical transposed convolutional layers because it was shown that fewer artifacts are created by considering those layers. Another encoder-decoder system can be found in [12] where the architecture also contains residual connections in a

¹ <https://github.com/mathieulagrange/ddspMusicBandwidthExtension>.

U-Net scheme. The authors show (1) that using a ResNet architecture outperforms the U-Net, probably because of the loss of information in the bottleneck layer of the former and (2) that the employed DNNs overfit on the filter shapes present in the training data. This latter problem can fortunately be alleviated with a data augmentation strategy which utilizes a wide variety of low-pass filters during training.

As the paper describes thoroughly the architecture proposed as well as its learning procedure, we choose to use this latter system as a reference “high complexity” system.

2.3 Generative adversarial networks

Generative adversarial networks (GAN) have been explored in several works for BWE. In [31], the authors show that relying on GANs can improve the generated speech quality by using a simple DNN. In [32], the generator is based on a U-Net-like architecture and the discriminator is trained to distinguish between generated and true wide-band signals, with the addition of a perceptual loss expressing the distance between features learned by a pre-trained automatic speech recognition (ASR) network. A combination of two discriminators, one based on spectral features and the other based on temporal features, have been proposed in [33] to extend the bandwidth from 8 kHz to 48 kHz. In [34], the generator is also based on a U-Net architecture yet it is proposed to employ CNNs for the three discriminators, each one being applied on a downsampled version of generated or true waveform (downsampling factors = 1, 2, 4). The generator is then trained to generate piano signals.

While GAN do not impose strong constraints in terms of inference complexity, GANs are known to be notoriously difficult to train, as they require very specific choices in optimization and architectures in order to stabilize training and could fail to cover modes of the data distribution [35].

At the time of the design of this study, we found no pre-trained general audio BWE model learnt with an adversarial procedure. We thus do not consider a GAN trained generator as another reference method.

2.4 Diffusion models

In terms of quality of generation, diffusion models now provides very convincing performance for a wide variety of data, including audio [10, 36]. As for autoregressive architecture like Wavenet [9], this important increase of quality comes at a strong computational cost at inference. The network has to be called sequentially a large number of times (usually from 100 to 1000 times) in order to perform the inference. One can reduce the size of the network or reduce the number of steps in order to accelerate

sampling [35], but those approaches are detrimental to the quality of the generated audio and the inference time remains high.

In this paper, we find that the inference of a standard ResNet architecture is already about 1000 times real time on a standard central processing unit (CPU) and our study focuses on efficient BWE, we choose not to consider diffusion models as a reference.

3 Differentiable sound models

In this article, we address BWE using DDSF models derived from the seminal work of [11] that focuses on the generation of audio signals with a combination of neural networks and digital signal processing models. This approach allows one to train the neural network parameters in an end-to-end fashion with backpropagation, if the rest of the model is differentiable. Besides several sound synthesis models [37, 38], DDSF has also been successfully applied to other tasks, such as neural audio effect [39], style transfer [40], sound matching [41], or virtual analog [42].

In this section, we describe the DDSF models we propose for monophonic and polyphonic BWE.

3.1 Monophonic BWE system

To address BWE for monophonic musical signal, we adapt the model proposed in [11], which is monophonic by design. The main difference with the original DDSF model is that, in order to reconstruct the higher frequencies, the model takes as input the low-band (LB) audio signal of bandwidth $\frac{f_N}{\alpha}$, with f_N the Nyquist frequency, and is trained to output the wide-band (WB) signal of bandwidth f_N . The overall architecture, illustrated on Fig. 1 is the same as in [11], and consists in two parts: a trainable encoder-decoder neural network, and a harmonic-plus-noise synthesizer. The neural network is illustrated in blue, the extracted features are shown in yellow, and the differentiable synthesizer is colored in red. This monophonic model is labeled *DDSF-mono-dec*, referring to the design of the decoder to generate monophonic parameters.

3.1.1 Extracted features

The input LB signal is first analyzed to extract the fundamental frequency $f_0(n)$ and loudness $l(n)$ over time. In the monophonic setting, we use CREPE [43], a state-of-the-art monophonic pitch estimator based on a convolutional neural network, to estimate f_0 . The loudness l is obtained with a A-weighting of the power spectrum [44].

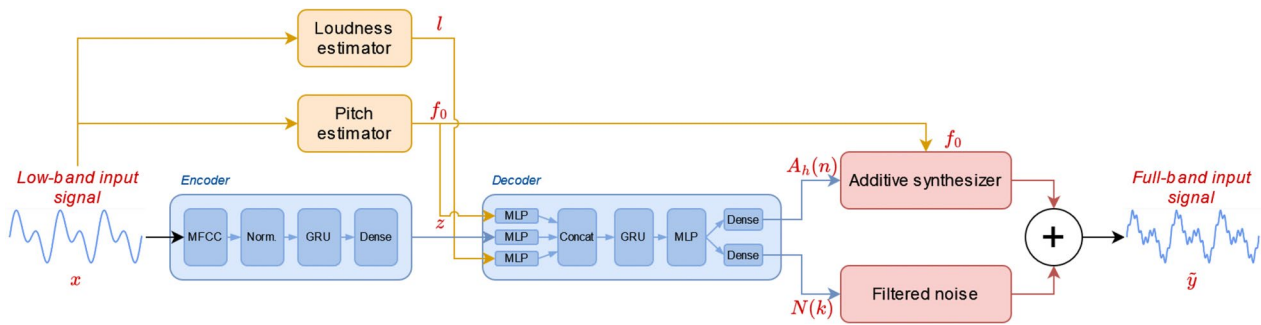


Fig. 1 DDSP architecture for monophonic bandwidth extension (model *DDSP-mono-dec*)

3.1.2 Neural network

The input LB signal waveform is processed by an encoder which creates a latent vector z . In the encoder, the first 30 mel frequency cepstrum coefficients (MFCC) are extracted from the audio input (fast Fourier transform (FFT) size of 1024, overlap of 75 % and 128 mels between 20 and 8000 Hz) and then passed into a trainable normalization layer. After that, the MFCCs go into a gated recurrent unit (GRU) with 512 units, and finally, a 512-neuron linear layer outputs the latent vector $z(n)$.

The three vectors $z(n)$, $f_0(n)$ and $l(n)$ are then fed into the decoder. Each of them first goes into a separate multi-layer perceptron (MLP) with three layers, and the outputs are concatenated. The obtained vector is processed by a 512-unit GRU and then another 3-layer MLP. Finally, two separate dense layers are used: the first one outputs the harmonic amplitudes $A_h(n)$ (see Section 3.1.3) using a softmax activation, and the second one gives the noise filter coefficients $N(k)$. Note that, as in [11], we use a modified sigmoid function $\sigma(x) = 2 \cdot \text{sigmoid}(x)^{\log(10)} + 10^{-7}$. This architecture has around 3k learnable parameters.

3.1.3 Harmonic-plus-noise synthesizer

Both outputs from the neural network are used separately in the additive synthesizer and noise modules. The additive synthesizer takes the estimated $f_0(n)$ and the inferred harmonic amplitudes $A_h(n)$ to generate the audio signal $y(n)$:

$$y(n) = \sum_{h=1}^H A_h(n) \sin(\phi_h(n)), \quad (1)$$

where ϕ_h is the instantaneous phase of the h th sinusoidal component. It is computed by integrating the instantaneous frequency $f_h(n) = hf_0(n)$:

$$\phi_h(n) = 2\pi \sum_{m=0}^n f_h(m) + \phi_{0,h}, \quad (2)$$

where $\phi_{0,h}$ is a random initial phase. In the filtered noise module, we obtain a time-domain finite impulse response (FIR) filter as the inverse discrete Fourier transform of the noise filter coefficients $N(k)$ from the neural network output. The filtered noise signal is synthesized by convolving a white noise with the FIR filter. The harmonic signal and filtered noise are finally summed to obtain the wide-band output signal.

Even if the full-band output signal is generated, only the missing high frequency content is kept, and added to the input low-band signal.

3.1.4 Noise-only synthesizer

We also consider a noise only synthesizer in which the output of the autoencoder only contains the noise filter coefficients $N(k)$. We label this model *DDSP-noise*. This will allow us in the experimental part to evaluate the respective value of the harmonic and noise parts of the synthesizer.

3.1.5 Loss function

We use the multi-scale spectral (MSS) loss function to train our models computed on the missing high-frequency region. It is defined as $\mathcal{L}(y, \tilde{y}) = \sum_s L_s(Y_s, \tilde{Y}_s)$, where Y_s and \tilde{Y}_s are the high-frequency magnitude spectrograms of the ground-truth signal y and the reconstructed signal \tilde{y} , respectively, computed using a FFT size s , and :

$$L_s(Y_s, \tilde{Y}_s) = \|Y_s - \tilde{Y}_s\|_1 + \|\log Y_s - \log \tilde{Y}_s\|_1, \quad (3)$$

$\|\cdot\|$ being the common L_1 norm. Indeed, experiments demonstrated that it is preferable to compute each loss only on the high-frequency region for solving the BWE task. We use the same set of FFT sizes as in [11], that is [2048, 1024, 512, 256, 128, 64] samples.

3.2 Polyphonic BWE methods

By the use of a single harmonic synthesizer, the DDSP architecture can only generate high frequency content

harmonically from a single f_0 . To address BWE for polyphonic musical signals, we propose two systems: a cyclic use of a monophonic BWE system detailed above, and a BWE system based on a polyphonic DDSP architecture.

3.2.1 Cyclic monophonic decoder

In the monophonic BWE system, the DDSP model generates a harmonic signal based on a single f_0 estimated from a monophonic pitch estimator [43]. Now that we are in a polyphonic context, we use a state-of-the-art multi-pitch estimator [45] which outputs a maximum of I different fundamental frequencies f_0^i . Considering that this multi-pitch estimator has a rather good performance, we propose to iteratively use the monophonic DDSP model *DDSP-mono-dec* in a cyclic manner, as illustrated in Fig. 2. We label this model *DDSP-mono-dec-cyclic*. Pseudocode of the overall algorithm is detailed in Algorithm 1.

Algorithm 1 Pseudocode algorithm of cyclic use of the monophonic DDSP model.

```

 $f_0^1, \dots, f_0^I = \text{multi\_pitch\_estimator}(x)$ 
 $x_{LB}^1 \leftarrow x$ 
 $\tilde{y} = 0$ 
for  $i := 1$  to  $I$  step 1 do
   $l^i = \text{loudness\_estimator}(x_{LB}^i)$ 
   $\tilde{y}^i = \text{DDSP\_mono\_dec}(x_{LB}^i, f_0^i, l^i)$ 
   $\tilde{y}_{LB}^i = \text{low\_pass}(\tilde{y}^i)$ 
   $x_{LB}^{i+1} = x_{LB}^i - \tilde{y}_{LB}^i \approx \text{STFT}(\tilde{y}_{LB}^i)$ 
   $\tilde{X}_{LB}^{i+1} = \text{STFT}(\tilde{y}_{LB}^i)$ 
   $|X_{LB}^{i+1}| = |X_{LB}^i| - |\tilde{X}_{LB}^{i+1}|$   $\triangleright$  removing low-band content
   $x_{LB}^{i+1} = \text{ISTFT}(|X_{LB}^{i+1}|, \angle X_{LB}^i)$   $\triangleright$  back to time domain
   $\tilde{y} = \tilde{y} + \tilde{y}^i$   $\triangleright$  constructing final signal step by step
end for
return  $\tilde{y}$ 

```

Algorithm 1 Pseudocode algorithm of cyclic use of the monophonic DDSP model

The monophonic DDSP model is applied for I iterations on a low-band signal x_{LB}^i which correspond to

the original low-band signal minus the $i - 1$ estimated sources. At each iteration i , a loudness contour l^i is extracted from what we label a *residual* low-band input signal x_{LB}^i and passed, along with the i^{th} estimated pitch f_0^i (obtained on x at the beginning of the algorithm) and x_{LB}^i into the DDSP model.

The output full-band monophonic signal \tilde{y}^i , which contains a harmonic content from the current f_0^i , is then low-pass filtered to keep only the low-frequency part \tilde{y}_{LB}^i . Finally, the magnitude spectrogram of \tilde{y}_{LB}^i is subtracted to the magnitude spectrogram of the residual low-band input signal:

$$|X_{LB}^i| = \begin{cases} |X| & \text{if } i = 1 \\ |X_{LB}^{i-1}| - |\tilde{Y}_{LB}^{i-1}| & \text{if } i \in \{2, \dots, I\} \end{cases} \quad (4)$$

The low-band input signal is then obtained in the time-domain using an inverse short-term Fourier transform (STFT) on X_{LB}^i (phase is kept in place).

In that way, at each iteration i , the harmonic content generated at the previous step is removed in the spectral domain from the residual low-band input signal, so that a different f_0^i should be extracted. The residual low-band signal should contain less and less harmonics during this process.

At the beginning of the iteration, the loudness contour is then estimated on the full polyphonic signal, which will lead to estimations errors, that hopefully will decrease at each iteration.

The output of the noise synthesizer, which is part of the monophonic DDSP model at each iteration in order to have a more precise estimate of the amplitude of the harmonic of the sinusoidal part. While the noise part is thus estimated at each iteration we only considered the noise part of the last iteration I in order not to overestimate the noise part.

Finally, the full-band monophonic output signals \tilde{y}^i are summed and mixed with the noise part to obtain the estimated full-band polyphonic signal \tilde{y} . As in the

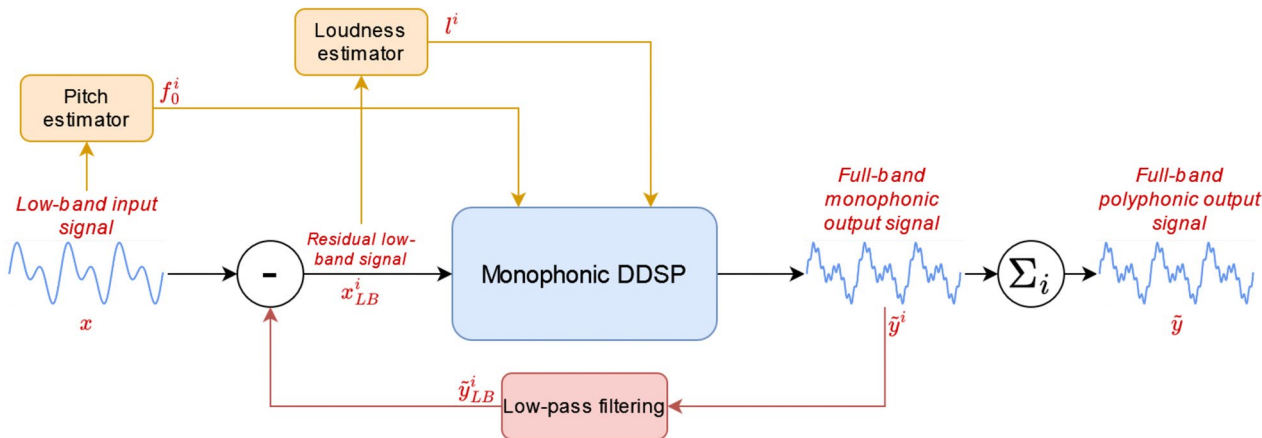


Fig. 2 Cyclic use of the *DDSP-mono-dec-cyclic* for bandwidth extension of polyphonic signals

monophonic BWE setting, the high frequency content from this full-band signal is mixed with the low-band input signal.

3.2.2 Polyphonic decoder

To address BWE for polyphonic signals, we propose another model adapted from the original DDSP models, illustrated in Fig. 3, which we label *DDSP-poly-dec* because the decoder outputs the parameters intended to control a polyphonic synthesizer. As before, the model is trained on polyphonic data.

In this model, I additive harmonic synthesizers are used, where I is the estimated number of fundamental frequencies $f_0^i, i \in 1, \dots, I$ present in the input low-band signal. To estimate the parameters for each separate additive synthesizer, we extend the decoder detailed in section 3.1.2 by using I separate MLPs for each f_0^i (instead of a single MLP for vector f_0 in the monophonic DDSP model). The outputs of those I MLPs are then concatenated into one vector, which is itself concatenated to the outputs of the two other MLPs applied on z and l . Then, as in the monophonic model, the obtained vector goes through a GRU and another MLP. After that, $I + 1$ dense layers are used: one for estimating the noise filter coefficients $N(k)$, and I other layers to output the H harmonic amplitudes of the I additive synthesizers.

In this model, we employ the same multi-pitch estimator [45] as in the cyclic model to estimate a maximum of I f_0^i . If only $I' < I$ fundamental frequencies are given by the estimator, we set $f_0^i = 0, i > I'$, and all f_0^i are fed in the decoder. To prevent any adverse impact on sound quality of those missing values, only the I' first sets of H harmonic amplitudes are extracted from the decoder output and used with the first I' additive synthesizers.

4 Experimental protocol

In this section, we detail the datasets, metrics and baselines used to assess the performance of the proposed BWE models.

The task that we consider is bandwidth extension task where the input signal is sampled at 4kHz, thus with

frequencies up to 2kHz and the output signal is sampled at 16kHz, thus with frequencies up to 8kHz.

As our approach is quite flexible in terms of extension scenario, we also performed experiments for the task going from a sampling frequency of 8kHz to 16kHz. We found that the ranking between models was the same as the one for upsampling from 2 to 8kHz. We thus display and discuss results only for the latter, as the task is more challenging and lead to more salient perceptual differences, a required aspect for a successful perceptual evaluation.

4.1 Datasets

To train and evaluate our models, we used both monophonic and polyphonic datasets. Synthetic data has been also considered in order to check expected behaviors of proposed systems. Those systems are then evaluated on uncontrolled real-world data.

4.1.1 Synthetic datasets

In order to analyze the inference capabilities of the trained models, we generated two synthetic datasets, respectively containing monophonic and polyphonic signals. These signals are generated using a harmonic-plus-noise synthesizer, as for the DDSP models, allowing for precise analysis of the models generating capabilities.

Each monophonic signal is generated given a f_0 corresponding to a certain MIDI pitch between C3 (e.g., 130.82 Hz) and G#6 (e.g., 1661.22 Hz). A harmonic signal is generated from this f_0 with H harmonics ($H \in \{10, 15, 20\}$), where the amplitude of the h th harmonic is $\frac{1}{h^2}$. A pink noise is added to this harmonic signal with a signal-to-noise ratio of 10 dB. Then, an *attack, sustain, decay* (ASD) envelope is generated and multiplied to the harmonic-plus-noise signal. The durations of attack and decay and the sustain level are randomly picked in the interval $[0, 0.3]$, $[0.5, 1]$, and $[0, 2]$ (in seconds), respectively. Finally, a random gain in interval $[0.75, 1]$ scales the final monophonic harmonic-plus-noise signal. The final monophonic synthetic dataset is obtained by generating all combinations of f_0 with the three H values, giving three signals.

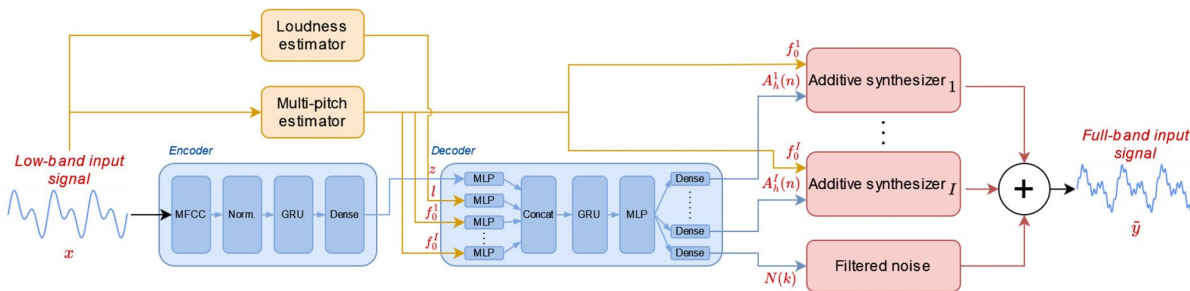


Fig. 3 Polyphonic DDSP model (*DDSP-poly-dec*) for bandwidth extension

The polyphonic synthetic dataset is generated by composing chords on the diatonic scale simply by considering multiple notes from the monophonic synthetic dataset, as follows. To generate a I -note polyphonic chord signal, we randomly pick I monophonic signals by taking care that a particular pitch (regardless of the octave) does not appear more than once among these I signals. For each note, a gain is randomly picked in $[0.5, 1]$, and all notes are mixed with corresponding gains. To build the full database, we generated polyphonic signals for all combinations of f_0 and $I \in \{2, 3, 4, 5\}$.

From the generated monophonic and polyphonic synthetic datasets, 90% of the signals form the train set, and the remaining signals form the test set.

4.1.2 Real-world monophonic datasets

Two real-world datasets consisting of monophonic musical signals are used to evaluate our models. The OrchideaSOL dataset [46] includes signals of single notes from many different instruments (accordion, bassoon, tuba, horn, trombone, trumpet, guitar, harp, contrabass, viola, violin, violoncello, clarinet, flute, oboe, and saxophone). In the original dataset, many different playing styles are available for each instrument; however, we only keep the *ordinario* one, corresponding to a natural playing. The training set for our experiments contains 90% of the original dataset, i.e., about 5.5 h of audio, while the test set contains 10%, i.e., about 42 min of audio.

Medley-solos-db [47] is another largely monophonic dataset which contains melodies of one of eight different instruments (clarinet, distorted electric guitar, female singer, flute, piano, saxophone, trumpet, and violin), i.e. the f_0 changes over time in those signals. In our experiments, we considered the original provided test and train splits, which corresponds to about 2.4 and 5 h of audio, respectively. As some of the instruments are polyphonic i.e. distorted electric guitar, piano, and violin, a small part of the dataset cannot strictly be considered as monophonic. In order to preserve the integrity of train/test splits of the dataset, we chose not to discard those instruments.

4.1.3 Real-world polyphonic datasets

To assess the proposed model for polyphonic BWE, we employed two real-world datasets containing multiple multi-track mixes. Gtzan dataset [48] has been widely exploited in many audio signal processing tasks. It contains 1000 30-s music tracks equally split into 10 genres (blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock). The train and test splits contain 7.5 h and 50 min of audio, respectively.

We also used the mixed version of each track of the MedleyDB dataset [49], since most of the corresponding

stems are already part of the training split of the previously mentioned Medley-solos-db dataset. The whole MedleyDB dataset is split into train and test sets in a 90% /10% way, corresponding to approximately 6 h and 50 min of audio data, respectively.

4.2 Evaluation and metrics

To evaluate the performance of the proposed models, we first employ an objective metric computed in the frequency domain named log-spectral distance (LSD), defined as:

$$LSD = \frac{1}{T} \sum_{t=1}^T \sqrt{\frac{1}{K} \sum_{k=1}^K \left(\log|Y(n,k)|^2 - \log|\tilde{Y}(n,k)|^2 \right)^2}, \quad (5)$$

where $Y(n, k)$ and $\tilde{Y}(n, k)$ are the STFT representation of the target full-band signal and the estimated full-band signal, respectively (see Section 4.4 for STFT parameter values).

Secondly, we ran a listening test based on a MUSHRA methodology [50] to assess the perceptive accuracy of the proposed models. We followed the classical MUSHRA specifications to build a listening test which was completed by 44 participants. More details are provided in Section 6.

4.3 Reference methods

4.3.1 Null baseline

The null baseline is simply the absence of addition of any content in the missing high frequency range. It provides a “ground floor” baseline to assess if the contribution of a given method is not actually worse than doing nothing.

4.3.2 Spectral band replication

To compare the performance of our models with existing approaches in the literature, a simplified version of the SBR algorithm [13] has been evaluated on the considered datasets. This algorithm has a long history in audio codec technologies and comes in various designs that often consider the use of side information, transmitted in the bitstream for the decoder to perform BWE. In this work, we implemented a simplified version that is blind, i.e. does not require any information for performing BWE.

In this algorithm, the input signal x is treated in the frequency domain, frame by frame. The upper half frequencies are inferred by replicating the lower half frequencies, with the idea of transposing the lower harmonics upwards. As we aim to extend the bandwidth of musical signal from 2 to 8 kHz, we extend this algorithm by replicating the lower band three times to reconstruct the full spectrum. In order to obtain a typical frequency amplitude decay, for each replication, the

amplitudes of the transposed frequencies are adjusted so that there is an energy continuity at the replication frontier, i.e., for the j th replication ($j \in \{1, 2, 3\}$ as it is a fourfold bandwidth extension), the energies of the same portion of frequencies on both sizes are equal:

$$\sum_{k=(j-\alpha)\frac{K}{4}}^{j\frac{K}{4}} |X(k)|^2 = \sum_{k=j\frac{K}{4}}^{(j+\alpha)\frac{K}{4}} |X(k)|^2, \quad (6)$$

where K is the number of frequency bins, and $\alpha \in [0, 1]$ the fraction of frequency bins considered for matching the energies of adjacent replicated bands. Experimentally, we found that $\alpha = 0.5$ led to the best performance for the overall algorithm.

In our experiment, to consider the SBR at its best performance, the ground-truth phase information is used to obtain the full-band signal in the temporal domain. We acknowledge that the phase is not known in practice and would have to be estimated in a realistic production setting.

4.3.3 ResNet architecture

We also compare our models to a higher complexity system based on deep learning [12]. We chose this system because the ResNet architecture shows better results than the other proposed model based on a U-Net architecture. The ResNet architecture takes an input signal in the temporal domain and outputs a signal of the same size, with high-frequency components. It is composed of 15 residual blocks made of two 1D convolutional layers each, with 512 convolutional filters of size 7, with a rectified linear unit (ReLU) activation after the first layer. For each layer, the input is added back to the output after being multiplied by a factor of 0.1 (for stabilizing the training) in a residual fashion. The input signal is added back to the output. Batch normalization and dropout with a factor of 0.5 are used after each convolutional layer. This model has around 55M learnable parameters.

To train this model, we use the same strategy as in [12], i.e., using a mean square error loss with a learning rate reducing schedule.

4.4 Experimental parameters

In our work, all audio signals are sampled at $f_s = 16$ kHz. To compute the STFT of these signals, we used an analysis window of 1024 samples with a hop length of 256 samples. The input signals are of length 64,000 samples (= 4 s) for the DDSP models and the SBR baseline, and 8192 samples (≈ 0.5 s) for the ResNet model. In the DDSP models, we considered $H = 100$ a number of

harmonics and a size of $K = 65$ for the noise transfer function $N(k)$. In the cyclic DDSP system, we use a total number of $I = 5$ iterations.

DDSP models are trained for 25,000 steps with batches of size 32. We used the Adam optimizer with an initial learning rate of 0.001 for DDSP models, and the latter is halved if the loss has not been decreased during four plateaus of 2500 steps. We used A100 GPUs for the training, which permit us to train DDSP models for around 1 h for *DDSP-mono-dec*, 2 h for *DDSP-poly-dec*, while ResNet training took around 19 h.

5 Validation on synthetic data

In this section, we first study the performance of the proposed models against the baselines on the monophonic and polyphonic synthetic data. It allows for more detailed insights on the models' ability to accurately generate the missing high frequency content.

5.1 Monophonic dataset

We first trained and evaluated our monophonic DDSP model on the monophonic synthetic dataset against the SBR method [13] and the ResNet model [12]. Table 1 summarizes the results.

The results show the benefit of the DDSP model over the two reference models. On Fig. 4, the generated upper band from the proposed monophonic DDSP model, SBR baseline and ResNet models are illustrated for one frame of a particular synthetic signal with $f_0 \approx 830$ Hz. The DDSP model is robust enough to synthesize the wanted harmonics with matching amplitudes, showing that it is capable to learn the chosen harmonic amplitude decay. The SBR baseline duplicates the low-band harmonic content with an offset because of the mismatch between the cutoff frequency and f_0 , and the ResNet model is apparently not capable of generating relevant high frequency harmonics, thus minimizing its loss by very few addition of energy.

5.2 Polyphonic dataset

Table 2 shows the LSD metric for all models and baselines on the polyphonic synthetic dataset. We can see that all proposed models surpass the SBR baseline and the ResNet model, and that the BWE performance has

Table 1 Evaluation results for monophonic BWE model and baselines on the monophonic synthetic dataset

| Model | LSD |
|---------------|-------------|
| Null | 6.15 |
| SBR [13] | 4.19 |
| ResNet [12] | 4.34 |
| DDSP-noise | 5.04 |
| DDSP-mono-dec | 2.93 |

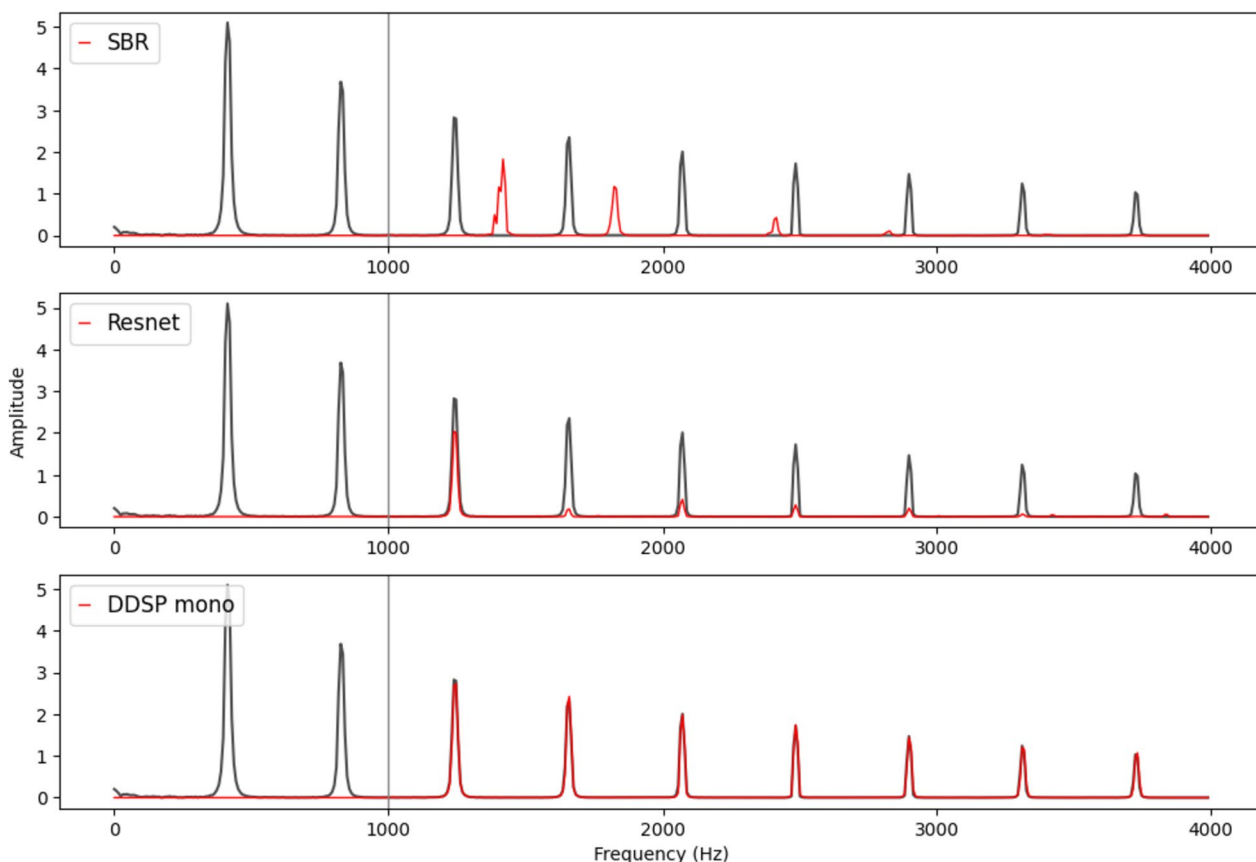


Fig. 4 Generated upper-frequency band using the model *DDSP-mono-dec*, the baseline SBR, and the reference ResNet model, for a synthetic signal containing harmonics based on the MIDI note G#5 (≈ 830 Hz). The vertical line shows the limit between the low and high bands

Table 2 Evaluation results for the proposed BWE models, the baseline and the reference ResNet model on the polyphonic synthetic datasets

| Model | LSD |
|----------------------|-------|
| Null | 11.03 |
| SBR [13] | 8.77 |
| ResNet [12] | 8.37 |
| DDSP-mono-dec | 7.86 |
| DDSP-noise | 9.18 |
| DDSP-mono-dec-cyclic | 5.59 |
| DDSP-poly-dec | 4.72 |

been improved by the design of both the cyclic system and the polyphonic model. The polyphonic DDSP model is almost twice as good as SBR and Resnet, which is an important improvement. When looking at Fig. 5, which illustrated the upper band generation for one polyphonic example from all the considered models and baselines, we notice that both cyclic and polyphonic methods are capable of generating precise harmonics, with a relatively

good amplitude match compare to the ground-truth. For the monophonic setting, the SBR baseline generates shifted harmonics. The ResNet model seems to be able to focus only on some harmonics, with relatively precise amplitudes, while also generating some noise in the lowest generated frequencies.

The three DDSP-based models seems quite capable of estimating the low harmonic amplitudes, while the high harmonic content suffers from too high amplitudes, which may lead to non-natural artifacts. Possible reasons for this defect are given at the end of the next section.

6 Evaluation on real-world datasets

In this section, we present the performance results for each monophonic and polyphonic recorded datasets of the proposed models against the reference methods: namely SBR and ResNet model.

6.1 Objective evaluation

The proposed models are first evaluated objectively using the LSD metrics on the real-world datasets. Monophonic models *DDSP-mono-dec* and *DDSP-noise* are evaluated

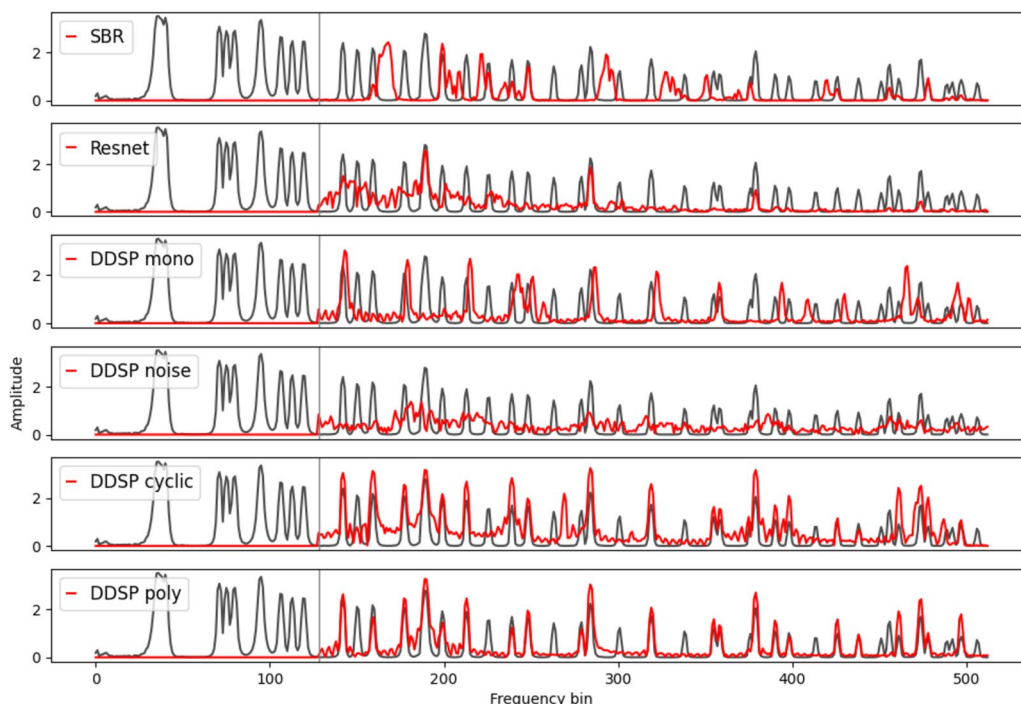


Fig. 5 Generated upper-frequency band using the proposed models and the baselines, for one frame of a synthetic signal containing four notes: C#5 (≈ 554 Hz), D#5 (≈ 587 Hz), D5 (≈ 622 Hz), and F#6 (≈ 1479 Hz). The vertical line shows the limit between the low and high bands

on both monophonic and polyphonic datasets, while polyphonic models *DDSP-mono-dec-cyclic* and *DDSP-poly-dec* are evaluated only on polyphonic datasets (*Gtzan* and *MedleyDB*). Table 3 shows the results.

First, we can see that all proposed models surpass both SBR and ResNet model in terms of LSD, except for the cyclic model which is worse than SBR. On the *OrchideaSOL*, *Gtzan*, and *MedleyDB* datasets, the gain in performance is substantial for the best model compared to the reference ones. For example, *DDSP-mono-dec* leads to a

LSD of 5.68 where SBR and ResNet achieve 9.27 and 14.04, respectively. On polyphonic signals, the ResNet model seems to be quite bad at predicting high frequencies (LSD = 26.84 and 16.17 on *Gtzan* and *MedleyDB*, respectively), whereas our DDSP-based models give quite lower LSDs (less than 12 for all these models on both datasets).

When looking at the performance of the proposed models, we first observe that the polyphonic models *DDSP-mono-dec-cyclic* and *DDSP-poly-dec* do not achieve a better performance than the monophonic one

Table 3 LSD performance of the evaluated models for monophonic and polyphonic real-world datasets. Best models are shown in bold for each dataset. The last two columns show CPU inference time expressed as real-time percentage and the number of parameters of the models

| Model | Log-spectral distance | | | | Inference time (% real-time) | # of parameters |
|----------------------|-----------------------|-----------------|---------------------|-------------|---------------------------------|-----------------|
| | Monophonic datasets | | Polyphonic datasets | | | |
| | OrchideaSOL | Medley-solos-db | MedleyDB | Gtzan | | |
| Null | 15.9 | 18.53 | 24.37 | 33.84 | 0 | 0 |
| SBR [13] | 9.27 | 8.78 | 11.15 | 12.96 | 2 | 0 |
| ResNet [12] | 14.04 | 15.65 | 16.17 | 26.84 | 48 | 55M |
| DDSP-noise | 7.20 | 8.28 | 8.96 | 10.06 | 3 | 3.5M |
| DDSP-mono-dec | 5.68 | 8.09 | 8.98 | 9.95 | 9 | 4.4M |
| DDSP-mono-dec-cyclic | / | / | 11.57 | 11.60 | 44 | 4.4M |
| DDSP-poly-dec | / | / | 9.53 | 10.31 | 9 | 7.5M |

DDSP-mono-dec, whereas the noise-only model *DDSP-noise* is on par with its results. This observation is quite contrasted from what we obtained on the synthetic datasets. When having a look at Fig. 6, we can see that *DDSP-poly-dec* seems to generate the highest frequency with too low amplitudes, whereas *DDSP-mono-dec* is a bit more precise in the high frequencies.

By informal listening of some reconstructed signals, we managed to distinguish two types of unwanted artifacts. The first kind happens when the amplitudes of the reconstructed harmonics are too high, which leads to a very synthetic high frequency reconstruction. One of the reasons for these wrongly inferred harmonic amplitudes is that, in both *DDSP-mono-dec-cyclic* and *DDSP-mono-dec*, the loudness contour is estimated for a mixture made of several f_0 , making it less trivial for the autoencoder to estimate each f_0 harmonic amplitudes. The second type of artifacts can be heard when the synthesized noise handles much of the high frequency content, while the harmonic amplitudes are too low, or even non-existent. This happens when the multi-pitch estimator fails to correctly predict the set of f_0 s, then the overall system do not generate high amplitude harmonics, and compensates with noise. Because of that, we conjecture that the proposed models should be more effective with a more robust multi-pitch estimation system.

6.2 Perceptual evaluation

In order to assess the perceptive value of our models, we conducted a listening test based on the MUSHRA methodology [50]. During the listening test, 42 subjects were asked to rate the quality of audio signals between 0 (poor quality) and 100 (perfect quality) against the reference (ground-truth full-band signal), which is expected to be rated 100. This behavior is expected by normal hearing and focused subjects, as the reference sound is provided for each trial. 10 stimuli are given in a random order. For each of them, 6 signals are to be rated :

- 1 Anchor 1: low-band input signal (model *Null*)
- 2 Anchor 2: hidden reference (ground-truth full-band signal)
- 3 SBR reconstruction
- 4 ResNet output
- 5 *DDSP-mono-dec* output
- 6 *DDSP-noise* output

The signals are taken from the Gtzan dataset [48], one of each genre, and only 5 s are extracted in the middle of the original signal. Information about participants is asked at the end of the survey, including gender, age, and the number of years of musical practice. Given the poorer LSD performance of our proposed polyphonic schemes

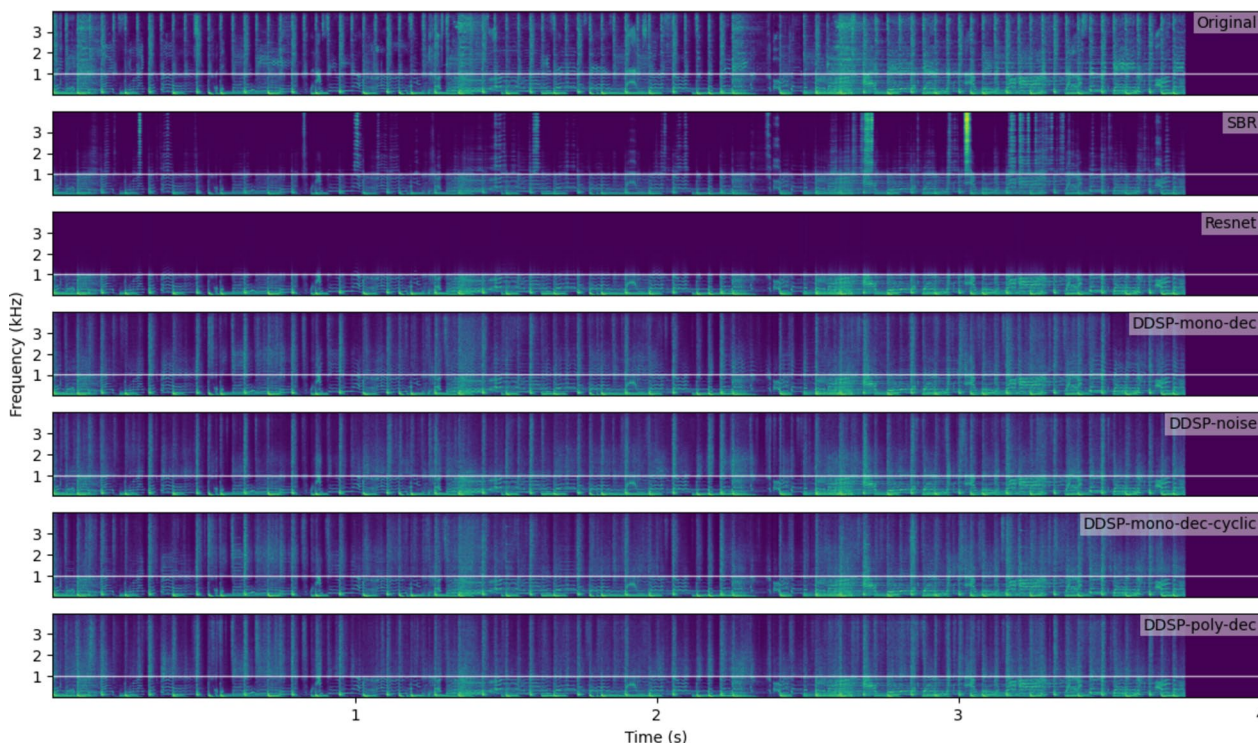


Fig. 6 Spectrograms showing the generated upper-frequency band using the proposed models and the baselines for a real-world signal from a pop music track. The horizontal white line shows the limit between the low and high bands at 1000 Hz

compared to the monophonic one, also confirmed by informal listening by the authors, it has been decided not to consider them for subjective evaluation. This had the benefit of maintaining the duration of the listening test into a reasonable range of about 20 min.

First, we conduct an analysis of variance (ANOVA) to check whether the factor of musical training is a significant source of variation in the rating data. We consider an individual as being a musician if it has an experience of at least 1 year. Considering that, the ANOVA run on the rating distributions for musician and non-musician subjects gives a p -value of $1.61 \cdot 10^{-8}$, which tells us that being a musician or not has a significant effect on the test ratings.

Close inspection of the ratings showed that the rankings of the different methods are the same for both populations. The only difference is a different bias, where musicians were on average more severe than non-musicians, as can be seen on Fig. 7, that show the distributions of ratings for musicians and non-musicians the over all models.

Next, conducting another ANOVA in which the analyzed factor is the model gives us a p -value of $2.18 \cdot 10^{-88}$, which is very small and shows that the choice of model is a significant source of variation in our collected data, thus the possibility of comparing the rating distributions of all models. With another ANOVAs on the models but for the data splits in musician or non-musician subsets, we obtain similar very low-valued p -values, which tells us that in both case the choice of models has an significant impact on the ratings among the participants.

The distributions of the participants' ratings for all models and all stimuli are plotted as boxplots in Fig. 8. We can see that the outputs of model *DDSP-mono-dec* are in average rated to be of *fair* quality (almost *good*), whereas the outputs from *ResNet*, *DDSP-noise*, and no processing are typically rated as *poor*, and *SBR* outputs are quite often rated as being of *bad* quality. An important outcome is that *DDSP-mono-dec* provides a large margin improvement compared to the *Null* baseline,

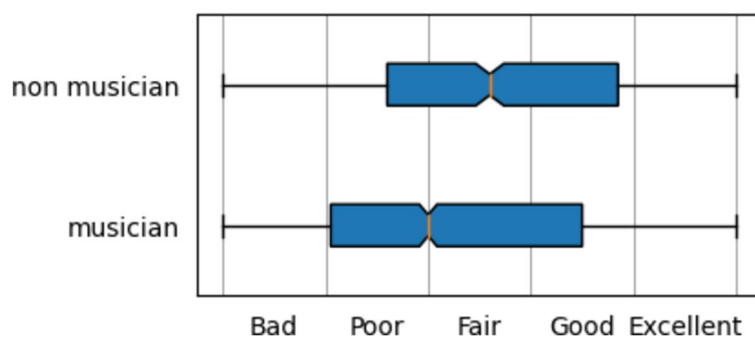


Fig. 7 Stimuli ratings from subjects with (bottom) and without (top) musical training. Boxes correspond to the interquartile range (IQR) over all participants, with the mean indicated by an orange vertical line. Lower and upper whiskers are set to $1.5 \times IQR$ below and above Q_1 and Q_3 , respectively

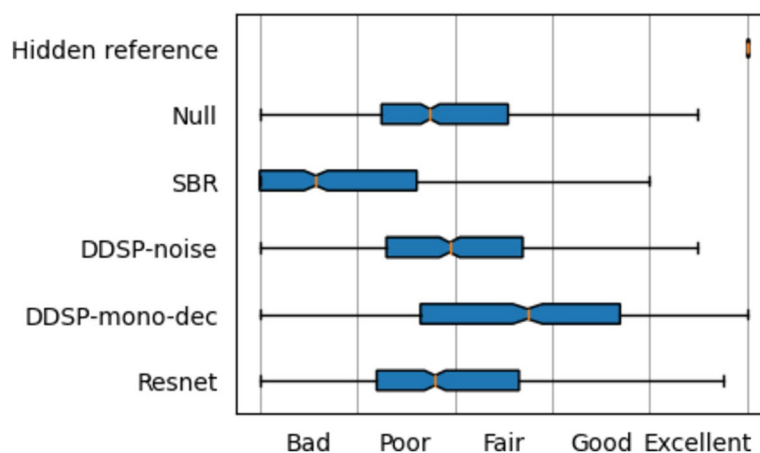


Fig. 8 Stimuli ratings for, from top to bottom, the anchor *Null* corresponding to the input signal without any process, the *SBR* baseline, the proposed models *DDSP-mono-dec* and *DDSP-noise*, and the reference *ResNet* model. Boxes correspond to the interquartile range (IQR) over all participants, with the mean indicated by an orange vertical line. Lower and upper whiskers are set to $1.5 \times IQR$ below and above Q_1 and Q_3 , respectively

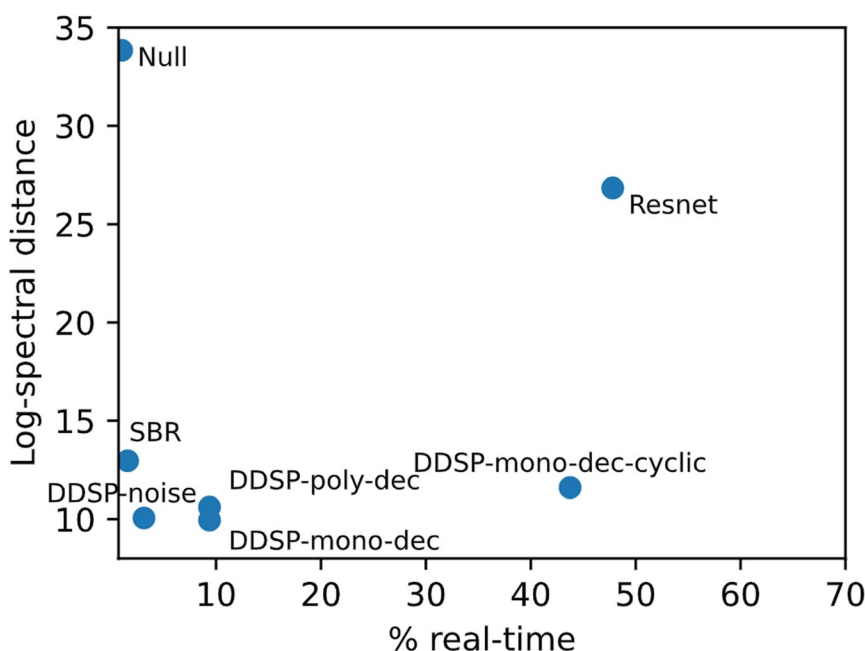


Fig. 9 Real-time CPU inference percentage vs. LSDs for the proposed models and the baselines over the dataset Gtzan

meaning that this method is able to improve audio quality at a low computational cost.

By computing a t -test on rating distribution of *DDSP-mono-dec* against the other models, we can verify that it is significantly better than the other ones. The p -values obtained for the t -test are well below the typical threshold of 0.05, so the distributions are significantly different from each other. We can thus conclude that, from the data of the listening test, the *DDSP-mono-dec* gives perceptively better high-frequency contents than the other evaluated models.

Monophonic and polyphonic examples are available online². The latter have been considered as stimuli in the listening test.

6.3 Inference time

One great advantage of the proposed DDSP approach is the important reduction of inference time compared to neural networks with a lot of training parameters such as Resnet. Figure 9 shows a scatter plot of the performance against the inference time of the different proposed models and the baselines SBR and Resnet, on the Gtzan dataset. The inference was made on a laptop equipped with an Intel Core i7 CPU at a frequency of 2.8 GHz. We can clearly see that a neural network architecture such as

ResNet takes a lot of computing time to process an input signal, well above a potential real-time behavior. While SBR is very fast, DDSP-based models such as *DDSP-mono-dec*, *DDSP-poly-dec*, and *DDSP-noise* are quite efficient in terms of computation time. *DDSP-mono-dec* and *DDSP-poly-dec* take the same amount of computing time because their architecture is very similar, and *DDSP-noise* is a bit faster because of the smaller matrices in the decoder. On the other hand, *DDSP-mono-dec-cyclic* is less computationally efficient because of its iterative nature, as an inference from *DDSP-mono-dec* is computed at each iteration. These insights on the computational power of the DDSP-based model show the advantage of such hybrid models compared to neural networks with a huge number of parameters such as the ResNet architecture.

7 Conclusion

In this article, we explored differentiable digital signal processing models for bandwidth extension of monophonic and polyphonic musical signals. We showed the benefit of using a monophonic DDSP model to generate high frequencies of monophonic signals against the two baselines, including a high-complexity deep-learning-based ResNet model. Then, we designed two systems to address polyphonic BWE: a cyclic use of a monophonic DDSP model, and an adapted DDSP model with polyphonic

² <https://mathieulagrange.github.io/ddspMusicBandwidthExtension>

synthesis capacities. On polyphonic signals, the proposed polyphonic systems showed to be more effective on polyphonic synthetic signals, but failed to surpass the monophonic DDSP model on real data. In addition, we conducted a listening test with the MUSHRA methodology, which showed that the *DDSP-mono-dec* model was more pleasant to the ear for most participants, when compared to the baselines. For future work, we think that considering a more advanced multi-pitch estimator could enable the polyphonic models to generate less artifacts and that other artifacts could be avoided by researching further the loudness estimation procedure.

Abbreviations

| | |
|--------|---|
| ANOVA | Analysis of variance |
| ASD | Attack, sustain, release |
| ASR | Automatic speech recognition |
| BWE | Bandwidth extension |
| CNN | Convolutional neural network |
| CPU | Central processing unit |
| DDSP | Differentiable digital signal processing |
| DL | Deep learning |
| DNN | Deep neural network |
| FFT | Fast Fourier transform |
| FIR | Finite impulse response |
| GAN | Generative adversarial network |
| GBRBM | Gaussian-Bernoulli restricted Boltzmann machine |
| GMM | Gaussian mixture model |
| GPU | Graphics processing unit |
| GRU | Gated recurrent unit |
| HMM | Hidden Markov model |
| LB | Low-band |
| LSD | Log-spectral distance |
| LSTM | Long short-term memory |
| MFCC | Mel frequency cepstral coefficients |
| MLP | Multi-layer perceptron |
| MSE | Mean-squared error |
| MSS | Multi-scale spectral |
| MUSHRA | Multiple stimuli with hidden reference and anchor |
| NMF | Non-negative matrix factorization |
| ReLU | Rectified linear unit |
| SBR | Spectral band replication |
| STFT | Short-term Fourier transform |
| WB | Wide-band |

Acknowledgements

We would like to thank Vincent Lostanlen for fruitful discussions and suggestions.

Authors' contributions

PAG conducted the numerical experiments and wrote the manuscript. ML provided guidance and wrote the manuscript.

Funding

This research have been partially funded by an RFI OIC grant.

Availability of data and materials

Experiments reported in this paper rely on publicly available code and on the following publicly available datasets:

- 1 OrchideaSOL: <https://forum.ircam.fr/projects/detail/orchideasol>
- 2 Medley-solos-db: <https://zenodo.org/record/1344103>
- 3 MedleyDB: <https://medleydb.weebly.com>
- 4 GTZAN: <https://www.kaggle.com/datasets/andradaolteanu/gtzan-datas-et-music-genre-classification>

The synthetic datasets can be reproduced using the experimental code available at: <https://github.com/mathieulagrange/ddspMusicBandwidthExtension>.

Declarations

Competing interests

The authors declare that they have no competing interests.

Received: 16 June 2023 Accepted: 2 November 2023

Published online: 05 December 2023

References

1. E. Vincent, T. Virtanen, S. Gannot (eds.), *Audio Source Separation and Speech Enhancement* (Wiley, 2018)
2. A. Adler, V. Emiya, M.G. Jafari, M. Elad, R. Gribonval, M.D. Plumbley, Audio inpainting. *IEEE Trans. Audio Speech Lang. Process.* **20**, 922–932 (2012)
3. N.R. French, J.C. Steinberg, Factors governing the intelligibility of speech sounds. *J. Acoust. Soc. Am.* **19**, 90–119 (1947)
4. S.V. Vaseghi, R. Frayling-Cork, Restoration of old gramophone recordings. *J. Audio Eng. Soc.* **40**, 791–801 (1992)
5. C. Gaultier, S. Kitić, R. Gribonval, N. Bertin, Sparsity-based audio declipping methods: selected overview, new algorithms, and large-scale evaluation. *IEEE/ACM Trans. Audio Speech Lang. Process.* **29**, 1174–1187 (2021)
6. M. Dietz, L. Liljeryd, K. Kjørling, O. Kunz, in *Audio Engineering Society Convention*, Spectral band replication, a novel approach in audio coding (2002)
7. Y. Ning, S. He, Z. Wu, C. Xing, L.-J. Zhang, A review of deep learning based speech synthesis. *Appl. Sci.* **9**, 4050–4066 (2019)
8. S.H. Mohammadi, A. Kain, in *Speech Communication*, An overview of voice conversion systems (2017)
9. A.v.d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, Wavenet: a generative model for raw audio. *Proc. ISCA* (2016)
10. E. Moliner, J. Lehtinen, V. Välimäki, in *International Conference on Acoustics, Speech and Signal Processing*, Solving audio inverse problems with a diffusion model (2023)
11. J. Engel, L. Hantrakul, C. Gu, A. Roberts, DDSP: Differentiable Digital Signal Processing. (International Conference on Learning Representations, 2020)
12. S. Sulun, M.E.P. Davies, On filter generalization for music bandwidth extension using deep neural networks. *J. Sel. Top. Signal Process.* **15**, 132–142 (2021)
13. S. Meltzer, R. Bohm, F. Henn, in *Audio Engineering Society Convention*, SBR enhanced audio codecs for digital broadcasting such as "Digital Radio Mondiale" (DRM) (2002)
14. F. Nagel, S. Disch, in *International Conference on Acoustics, Speech and Signal Processing*, A harmonic bandwidth extension method for audio codecs (2009)
15. S. Chennouk, A. Gerrits, G. Miet, R. Sluijter, in *International Conference on Acoustics, Speech, and Signal Processing. Proceedings*, Speech enhancement via frequency bandwidth extension using line spectral frequencies (2001)
16. J. Sadasivan, S. Mukherjee, C.S. Seelamantula, in *International Conference on Acoustics, Speech and Signal Processing*, Joint dictionary training for bandwidth extension of speech signals (2016)
17. Y. Yoshida, M. Abe, in *International Conference on Spoken Language Processing*, An algorithm to reconstruct wideband speech from narrowband speech based on codebook mapping (1994)
18. K.-Y. Park, H.S. Kim, in *International Conference on Acoustics, Speech, and Signal Processing. Proceedings*, Narrowband to wideband conversion of speech using GMM based transformation (2000)
19. P. Bauer, T. Fingscheidt, in *International Conference on Acoustics, Speech and Signal Processing*, An HMM-based artificial bandwidth extension evaluated by cross-language training and test (2008)
20. G.-B. Song, P. Martynovich, A study of HMM-based bandwidth extension of speech signals. *Signal Process.* **89**, 2036–2044 (2009)
21. D. Bansal, B. Raj, P. Smaragdis, in *Interspeech*, Bandwidth expansion of narrowband speech using non-negative matrix factorization (2005)
22. D.L. Sun, R. Mazumder, in *International Workshop on Machine Learning for Signal Processing*, Non-negative matrix completion for bandwidth extension: A convex optimization approach (2013)
23. K. Li, C.-H. Lee, in *International Conference on Acoustics, Speech and Signal Processing*, A deep neural network approach to speech bandwidth expansion (2015)
24. K. Li, Z. Huang, Y. Xu, C.-H. Lee, in *Interspeech*, DNN-based speech bandwidth expansion and its application to adding high-frequency missing features for automatic speech recognition of narrowband speech (2015)

25. Y. Wang, S. Zhao, W. Liu, M. Li, J. Kuang, in *Interspeech*, Speech bandwidth expansion based on deep neural networks (2015)
26. Y. Gu, Z.-H. Ling, in *Interspeech*, Waveform modeling using stacked dilated convolutional neural networks for speech bandwidth extension (2017)
27. V. Kuleshov, S.Z. Enam, S. Ermon, Audio super resolution using neural networks. (International Conference on Learning Representations, 2017)
28. H. Wang, D. Wang, in *International Conference on Acoustics, Speech and Signal Processing*, Time-frequency loss for CNN based speech super-resolution (2020)
29. G. Campos, N. Fonseca, A. Ferreira, M. Davies, in *Conference on Digital Audio Effects*, High frequency magnitude spectrogram reconstruction for music mixtures using convolutional autoencoders (2018)
30. M. Lagrange, F. Gontier, in *International Conference on Acoustics, Speech and Signal Processing*, Bandwidth extension of musical audio signals with no side information using dilated convolutional neural networks (2020)
31. S. Li, S. Villette, P. Ramadas, D.J. Sinder, in *International Conference on Acoustics, Speech and Signal Processing*, Speech bandwidth extension using generative adversarial networks (2018)
32. X. Li, V. Chebiyyam, K. Kirchhoff, in *Interspeech*, Speech audio super-resolution for speech recognition (2019)
33. J. Su, Y. Wang, A. Finkelstein, Z. Jin, in *International Conference on Acoustics, Speech and Signal Processing*, Bandwidth extension is all you need (2021)
34. E. Moliner, V. Välimäki, BEHM-GAN: Bandwidth Extension of Historical Music using Generative Adversarial Networks. *IEEE/ACM Trans. Audio Speech Lang. Process* **31**, 943–956 (2022)
35. J. Song, C. Meng, S. Ermon, Denoising diffusion implicit models. *International Conference on Learning Representations* (2020)
36. E. Moliner, F. Elvander, V. Välimäki, Zero-shot blind audio bandwidth extension. *arXiv preprint arXiv:2306.01433* (2023)
37. B. Hayes, C. Saitis, G. Fazekas, Neural waveshaping synthesis (International Conference on Music Information Retrieval, 2021)
38. S. Shan, L. Hantrakul, J. Chen, M. Avent, D. in *International Conference on Acoustics, Speech and Signal Processing*, Trevelyan, Differentiable wavetable synthesis (2021)
39. S. Lee, H.-S. Choi, K. Lee, Differentiable artificial reverberation. *IEEE/ACM Trans. Audio Speech Lang. Process* **30**, 2541–2556 (2022)
40. C.J. Steinmetz, N.J. Bryan, J.D. Reiss, Style transfer of audio effects with differentiable signal processing. *J. Audio Eng. Soc* **70**, 708–721 (2022)
41. N. Masuda, D. Saito, in *International Society for Music Information Retrieval Conference*, Synthesizer sound matching with differentiable DSP (2021)
42. F. Esqueda, B. Kuznetsov, J.D. Parker, in *International Conference on Digital Audio Effects*, Differentiable white-box virtual analog modeling (2021)
43. J.W. Kim, J. Salamon, P. Li, J.P. Bello, in *International Conference on Acoustics, Speech and Signal Processing*, Crepe: a Convolutional Representation for Pitch Estimation (2018)
44. L. Hantrakul, J. Engel, A. Roberts, C. Gu, in *International Society for Music Information Retrieval Conference*, Fast and flexible neural audio synthesis (2019)
45. R.M. Bittner, J.J. Bosch, D. Rubinstein, G. Meseguer-Brocal, S. Ewert, in *International Conference on Acoustics, Speech and Signal Processing*, A lightweight instrument-agnostic Model for polyphonic note transcription and multipitch estimation (2022)
46. C.E. Cella, D. Ghisi, V. Lostanlen, F. Lévy, J. Fineberg, Y. Maresz, Orchidea-SOL: a dataset of extended instrumental techniques for computer-aided orchestration (International Computer Music Conference, 2020)
47. V. Lostanlen, C.-E. Cella, Deep convolutional networks on the pitch spiral for musical instrument recognition (International Conference on Music Information Retrieval, 2017)
48. B.L. Sturm, An analysis of the GTZAN music genre dataset. In *Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies 7–12* (2012)
49. R.M. Bittner, J. Wilkins, H. Yip, J.P. Bello, in *International Conference on Music Information Retrieval*, MedleyDB 2.0: New data and a system for sustainable data collection (2016)
50. M. Schoeffler, S. Bartoschek, F.-R. Stöter, M. Roess, S. Westphal, B. Edler, J. Herre, webMUSHRA - A Comprehensive Framework for Web-based Listening Tests. *J. Open Res. Softw* **6**, 1-8 (2018)

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► [springeropen.com](https://www.springeropen.com)
