Upsampling Layers for Music Source Separation

Jordi Pons, Joan Serrà, Santiago Pascual, Giulio Cengarle, Daniel Arteaga, Davide Scaini
Dolby Laboratories

***Abstract*—Upsampling artifacts are caused by problematic upsampling layers, and due to spectral replicas that emerge while upsampling. Also, depending on the used upsampling layer, such artifacts can either be tonal (additive high-frequency noise) or filtering artifacts (substractive, attenuating some bands). We investigate the practical implications of having upsampling artifacts in the resulting audio, by studying how different artifacts interact and assessing their impact on the models' performance. To that end, we benchmark a large set of upsampling layers for music source separation: different transposed and subpixel convolution setups, different interpolation upsamplers (including two novel layers based on stretch and sinc interpolation), and different wavelet-based upsamplers (including a novel learnable wavelet layer). Our results show that filtering artifacts, associated with interpolation upsamplers, are perceptually preferrable, even if they tend to achieve worse objective scores.**

***Index Terms*—synthesis, artifacts, music, source separation.**

I. Introduction

Upsampling layers are widely used in audio synthesis. They are known to introduce undesired upsampling artifacts [1]–[5], which can be categorized as filtering or tonal artifacts [3]. Filtering artifacts attenuate some bands and are known to "de-emphasize high-end frequencies" [3], while tonal artifacts introduce additive periodic noise percieved as a "high-frequency buzzing noise" [2] that "can be devastating to audio generation results" [1]. A full, in-depth description of why upsampling artifacts occur is given in [3]. Here, we go one step further and investigate which strategies can work to palliate such artifacts.

In our work, we consider transposed convolutions [1], [5], [6], interpolation upsamplers [2], subpixel convolutions [4], and wavelet-based upsamplers [7]. Transposed and subpixel convolutions can introduce tonal artifacts [1], [2], [5], whereas interpolation and wavelet-based upsamplers can produce filtering artifacts [3] (see section II). In addition, recent research shows that tonal and filtering artifacts interact with the spectral replicas introduced via the bandwidth extension performed by each upsampling layer [3]. In light of that, which upsampling layers are preferable? We discuss their characteristics and how they interact with spectral replicas. A gentle introduction to the role of spectral replicas in neural upsampling is in [3]. Here, we dive deeper into the topic to discuss its implications both theoretically (section II) and, for the first time, empirically with metrics and a subjective test (sections III and IV), by considering the challenging task of music source separation [2], [6], [15].

We extensively benchmark existing upsampling layers to understand their behavior, and investigate two additional strategies to mitigate upsampling artifacts: (i) employing post-processing networks as an "a posteriori" mechanism to palliate upsampling artifacts [1], [8], and (ii) using normalization layers as a way to reduce spectral replicas of signal offsets (a source of artifacts, discussed in section II.E). Finally, we also experiment with upsampling layers that, to the best of our knowledge, have never been used for audio synthesis: stretch and sinc interpolation layers, and learnable wavelet layers. Our results show that filtering artifacts are perceptually preferable than tonal artifacts, and that nearest-neighbor interpolation can provide better separation quality than other upsampling layers.

II. Overview of Upsampling Layers and Artifacts

A. Transposed convolution and tonal artifacts

Transpososed convolutions can introduce undesired tonal artifacts due to (i) their weights' initialization, (ii) the loss function, and (iii) overlap issues [3], [9]. The weights' initialization issue (i) is caused because randomly initialized transposed convolutional filters repeat accross time [3], [5]. This issue is commonly addressed via learning from data, since training may help mitigating the tonal artifacts caused by this problematic initialization. Loss function issues (ii) emerge when convolutional neural networks (CNNs) are used for loss calculatioin, since it involves a transposed convolution step during backpropagation [9]. These issues can be avoided by just not using adversarial [1], [5] or deep feature losses [10]. Finally, overlap issues (iii) can be mitigated by carefully choosing the filter-length and stride [5]. In particular, one can use no overlap (length=stride) and full overlap (length is multiple of the stride) setups [3]. Yet, even under these setups, the loss function and the weights initialization issues remain. One can observe how tonal artifacts emerge after (random) initialization in Fig. 2 (a, b, c).

B. Interpolation upsamplers and filtering artifacts

Interpolation upsamplers first interpolate a given feature map and then employ a learnable convolution [9]. They do not cause tonal artifacts, but can introduce filtering artifacts [3] caused by the (fixed, non-learnable) frecuency response of the interpolation, see Fig. 1 (e, f, g). Filtering artifacts vary depending on the frequency response of each interpolation:

– *Stretch interpolation* upsamples the signal with zeros. Its frequency response is $X_{\text{stretch by}M}(\mathrm{e}^{j\omega}) = X(\mathrm{e}^{j\omega M})$, where M is the upsampling factor. Hence, it scales the frequency axis by M, exposing the spectral replicas without further transforming the signal. In other words: its flat frequency response does not introduce filtering artifacts, see Fig. 1 (h).

– *Sinc interpolation* can be implemented as a stretch interpolation + convolution with a sinc filter [11]. It is known as bandlimited interpolation since the frequency response of a sinc is a low-pass filter that removes the introduced spectral replicas, see Fig. 1 (e).

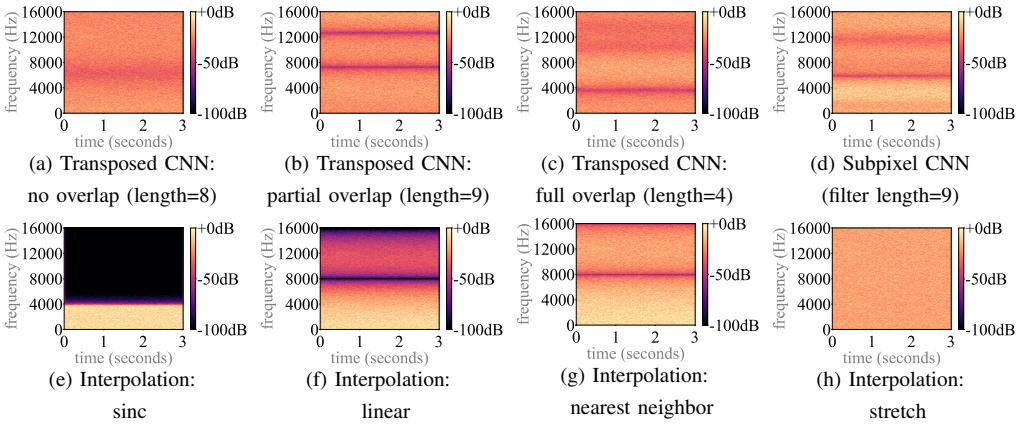


Fig. 1. Input: white noise at 8kHz. Upsampling (↑4) layers can introduce filtering artifacts that attenuate some bands. Only (e, f, g) "horizontal valleys" are considered filtering artifacts, because are caused by non-learnable interpolations. Hence, (e, f, g) layers would introduce filtering artifacts even after training—while the rest of "horizontal valleys" (a, b, c, d) can change during training. Transposed CNNs with stride=4.

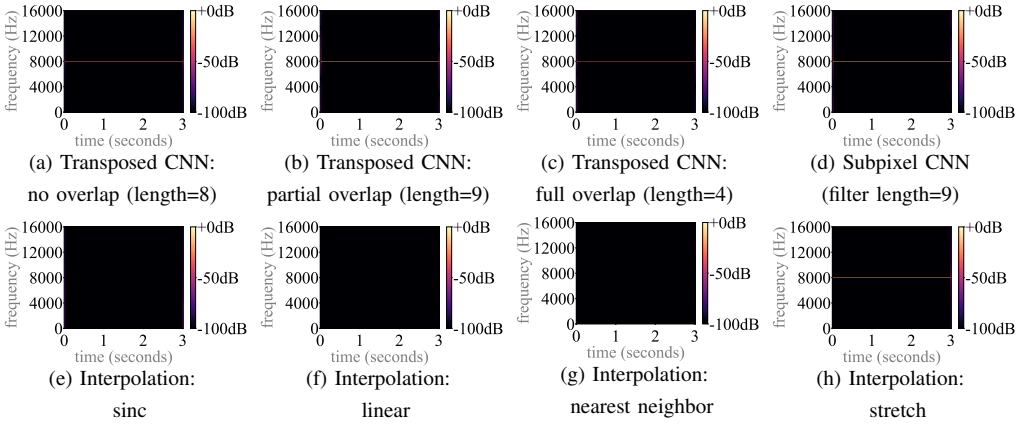


Fig. 2. Input: ones (constant) at 8kHz. Upsampling (↑4) layers can introduce tonal artifacts (horizontal lines). Transposed CNNs with stride=4.

– *Nearest neighbor interpolation*: can be implemented as stretch interpolation + convolution with a rectangular filter [3]. Since the frequency response of a rectangular filter is a sinc, its filtering artifacts attenuate (some) high-frequency bands, see Fig. 1 (g).

– *Linear interpolation* can be implemented as stretch interpolation + convolution with a triangular filter [3]. Since the frequency response of a triangular filter is a sinc^2, its filtering artifacts attenuate the high-frequency bands even more than the nearest neighbor interpolation, see Fig. 1 (f, g).

Note that the interpolations we have just discussed (e.g.: sinc or linear, explained as stretch interpolation + non-learnable convolution) are typically followed by a learnable convolution. To the best of our knowledge, we are the first to explore stretch and sinc interpolation layers for neural audio synthesis.

C. Subpixel convolution and tonal artifacts

Subpixel convolution is based on convolution and reshape [3], [4], [12]. The convolution upsamples the signal along the channel axis, and the reshape operation is based on periodic shuffling—that reorders the convolution's output to match the desired output shape, see Fig. 3. It can introduce tonal artifacts due to the periodic shuffle operator, since it

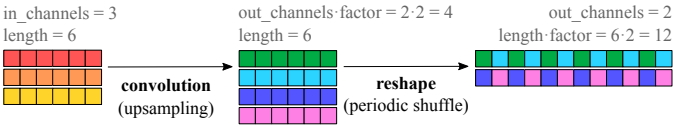


Fig. 3. Subpixel convolution: reshaping can introduce tonal artifacts.

interleaves consecutive samples out of convolutional filters having different weights (depicted in Fig. 3 as colored periodicities). Such periodicities are caused by the (different) dynamics each feature map can exhibit, since interleaving activations with different energies leads to periodic (tonal) artifacts [13]. This can be particularly noticeable after initialization, since feature maps can exhibit different dynamics due to the random weights of the filters as in Fig. 2 (d).

D. Wavelet-based upsamplers and filtering artifacts

Cascaded filter banks are often used to implement multiresolution discrete wavelet transforms [14] (Fig. 4). One interesting property of wavelets is that they allow for perfect reconstruction. Thus meaning that one can recover the input signal after following the analysis/synthesis steps. Hence, no additive (tonal) or substractive (filtering) artifacts emerge after downsampling/upsampling, but the perfect reconstruction

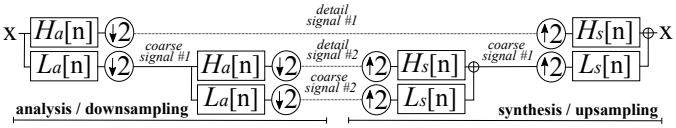


Fig. 4. Cascaded wavelets: with low-pass $L_a[n]$, $L_s[n]$ and high-pass $H_a[n]$, $H_s[n]$ filters. $\downarrow 2$ operator discards every other time step. $\uparrow 2$ operator refers to stretch interpolation, upsampling with zeros.

property only holds in the absence of any processing in the wavelet domain. We now describe wavelet-inspired neural downsampling/upsampling layers [7], [14]:

– *Lazy wavelet layers* downsample the signal into odd/even samples, and interleave odd/even samples for upsampling.

– *Haar wavelet layers* can be described following the filter bank scheme in Fig. 4, where analysis/synthesis filters are set as:

$$L_a[n] = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}], \quad H_a[n] = [\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}],$$
$$L_s[n] = L_a[-n], \quad H_s[n] = H_a[-n].$$

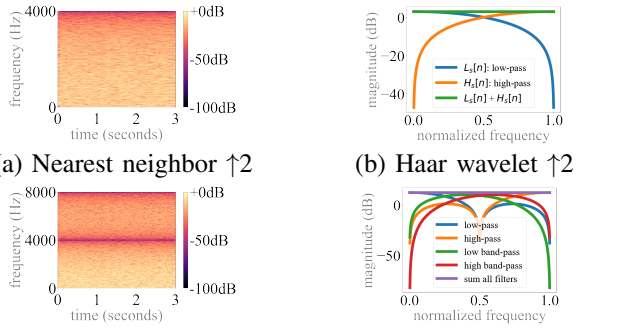
Importantly, the above wavelet upsampling paths have an overall frequency response that is flat. Note this in Fig. 5 (b, d), were we depict how Haar wavelet upsampling paths present a flat overall frequency response, allowing the above wavelets to compensate for filtering artifacts. To further undestand this, it is illustrative to see that the frequency response of nearest neigbor layers (Fig. 5: a, c) resemble the one of the low-pass filter of the Haar wavelets (Fig. 5: b, d). Hence, the alternative signal paths in wavelets allow compensating the filtering artifacts typically introduced by, e.g., nearest neighbor.

Further, wavelets provide a principled way to downsample. Inspired by that, Nakamura and Saruwatari [7] replaced WaveUnet's [2] downsampling (discarding every other time step) and upsampling (linear interpolation) layers by lazy and Haar wavelet layers. However, the above wavelet layers are designed to downsample/upsample by 2, as the original WaveUnet. In our work, we extend those to downsample/upsample by 4 via cascading wavelets as in Fig. 4.

We also experiment with *learnable wavelet layers*. To do so, we implement the above wavelets (both analysis and synthesis) using the lifting scheme [7], [14] that defines wavelet transforms with 3 parameters: a prediction operator P, an update operator U, and a normalization constant A [7], [14]. Haar and lazy wavelets can be implemented using the lifting scheme by setting P=1, U=0.5, A=$\sqrt{2}$ and P=0, U=0, A=1, respectively [7], [14]. By using the lifting scheme, wavelet layers can be implemented in a differentiable fashion where P, U and A are learnable, what defines our learnable wavelet downsampling/upsampling layers. In our experiments, we initialize P=0, U=0, and A=1 as the lazy wavelet. We are the first to study learnable wavelet layers for audio synthesis.

E. Discussion: pros and cons and spectral replicas

So far, we discussed filtering and tonal artifacts. Yet, spectral replicas can introduce additional artifacts [3]. In this section, we consider them to further study the above upsampling layers:



(a) Nearest neighbor $\uparrow 2$ (b) Haar wavelet $\uparrow 2$

(c) Nearest neighbor $\uparrow 4$ (d) Cascaded Haar wavelets $\uparrow 4$

Fig. 5. The frequency response of the nearest neighbor upsampler corresponds to the low-pass synthesis filter of the depicted wavelet layers. Left: white noise at 4kHz upsampled ($\uparrow$) by 2 or 4. Right: frequency response of the synthesis filters of Haar wavelet layers.

– *Spectral replicas of signal offsets*. Offsets are constants with zero frequency. Hence, its frequency transform contains an energy component at frequency zero. When upsampling, zero-frequency spectral replicas can appear in-band, introducing tonal artifacts. Specifically, they can appear at multiples of "sampling rate / upsampling factor" Hz (the sampling rate being the one of the upsampled signal). For example, in Fig. 2, such replicas appear at multiples of $32/4 = 8$ kHz. To further understand this concept, note that stretch interpolation does not introduce tonal artifacts. Yet, Fig. 2 (h) depicts tonal artifacts at multiples of 8kHz. This is because the spectral replicas of the constant signal appear in-band when upsampling [3].

– *Spectral replicas of offsets interact with filtering artifacts*. Note that interpolation-based upsamplers (which introduce filtering artifacts) can attenuate the exact bands where spectral replicas of signal offests appear. Namely, they can attenuate around "sampling rate / upsampling factor" Hz as depicted in Figs. 1 (e, f, g) and 5. Hence, filtering artifacts are a powerful tool to combat the spectral replicas of signal offsets. Note that no tonal artifacts due to signal offsets appear when uspampling a ones (constant) signal in Fig. 2 (e, f, g).

– *Spectral replicas of offsets interact with tonal artifacts*. The tonal artifacts introduced by transposed and subpixel convolutions occur at the same frequency as the first signal offset spectral replica (at "sampling rate / upsampling factor" Hz). Note in Fig. 2 (a, b, c, d) that tonal artifacts appear at 8kHz (due to transposed and subpixel convolutions, and signal offsets) and at 16kHz (due to signal offsets).

– *The relative energy of tonal artifacts*. Remarkably, we do not observe tonal artifacts in Fig. 1. Hence, the tonal artifacts introduced by transposed and subpixel convolutions (prone to introduce tonal artifacts) have little energy, compared to the energy of the signal being upsampled (zero-mean white noise). In contrast, Fig. 2 depicts tonal arifacts with similar energy for: (i) transposed and subpixel convolutions (prone to introduce tonal artifacts), and (ii) stretch interpolation (which does not introduce tonal artifacts). These observations denote that the spectral replicas of signal offsets can introduce stronger tonal artifacts than transposed and subpixel convolutions. For this reason, in section III, we include two modifications

meant to reduce signal offsets accross feature maps: removing learnable bias terms in our models, and studying to incorporate normalization layers.

– *Spectral replicas as a source of high-frequency content*. We can sort interpolation upsamplers by how strong their filtering artifacts are: sinc→linear→nearest neighbor→stretch, see Fig. 1. Note that sinc interpolation strongly filters the signal, and stretch introduces no filtering artifacts. While sinc interpolation is widely used in audio because it removes all spectral replicas, this might not be desirable for deep learning. We hypotesize that allowing spectral replicas accross feature maps is beneficial, as it allows the model to have access to coherent high-frequency feature maps for wide-band synthesis. We now experimentally validate this hypothesis.

III. Experimenting with upsampling layers

We use the MUSDB [15] music source separation benchmark to experiment with the above upsampling layers. It is composed of 150 stereo songs at 44.1 kHz: 86 train, 14 validation, 50 test. For each song, 4 stereo sources are extracted: vocals, bass, drums, and other. Table I reports our results on the test set, based on the average signal-to-distortion ratio (SDR) accross all sources [16].[2] We compare end-to-end Unet models, that are modified to accomodate the upsampling layers under discussion. Our base model is Demucs [6], an end-to-end Unet conformed by 6 encoding blocks (with strided 1D-CNN, ReLU, GLU) and 6 decoding blocks (with GLU, transposed 1D-CNN, ReLU), with skip connections and x2 LSTMs in the bottleneck (with 3200 units each). Strided and transposed convolution layers have 100, 200, 400, 800, 1600 and 3200 filters, respectively. We use no bias terms and the 1st encoder block has no ReLUs—since this helps reducing offsets accross feature maps, taming tonal artifacts after initialization (see section II.E or [3]). Our GLU non-linearities [17] rely on CNN filters of length 3. Like the original Demucs, we use: very large models, of ≈700M parameters; weight rescaling, so that input and output signals are of the same magnitude after initialization; and their data augmentation scheme, creating new mixes on-the-fly [6]. Further, we study two strategies to mitigate upsampling artifacts: (i) employing post-processing networks, as an "a posteriori" mechanism to palliate upsampling artifacts [1], [8]; and (ii) using normalization layers, as a way to reduce the spectral replicas of signal offsets. Post-networks (i) are conformed by 7 residual CNN layers (each with 8 filters of length 7, ≈6k parameters), and are trained keeping the "pre-network" pre-trained and frozen. We experimented with finetuning and with bigger/deeper post-networks with equivalent results. We use normalization layers (ii) both for the encoder and the decoder, to compensate for any signal offset (also coming from skip-connections) before upsampling. These are instance-norm [18] based, and implemented after each GLU (other normalizations performed worse). We train minimizing the L1 loss with Adam [19] for 600 epochs at a learning rate of 0.0002 (halved every 100 epochs) with 4 v100 GPUs, using batches of 32. Memory intensive runs, like WaveUnet and WaveletUnet ones, use batches of 16.

[2]Following previous works [2], [6]: for every source, we report the median over all tracks of the median SDR over each test track of MUSDB [15].

A. Demucs: models based on transposed convolution

Demucs are Unet models using strided convolutions for downsampling x4, transposed convolutions for upsampling x4, and are set to use the same filter length and stride for both downsampling and upsampling. We study 3 variants: partial overlap (length=9, stride=4), full overlap (length=8, stride=4), and no overlap (length=stride=4). Table I shows that partial overlap underperforms its counterparts, arguably because it is more prone to tonal artifacts due to the weights' initialization and overlap issues—note that no and full overlap only need to overcome the weights' initialization issue (see section II.A). Since the no overlap variant is faster to train (see epoch times, Table I) and obtains better results than full overlap, we select it for a listening test. When we extend no overlap with post-networks or with normalization, we observe no improvement.

B. WaveShuffle: models based on subpixel convolution

WaveShuffle are Unet models with strided convolutions (length=9, stride=4) for downsampling x4, and subpixel convolutions (length=9, stride=1) for upsampling x4. They achieve the best SDR results, specially when including normalization layers. Hence, we select the normalized variant for a listening test. Yet, informal listening[3] reveals that normalization layers did not remove the tonal artifacts as intended.

C. WaveUnet: interpolation-based models

WaveUnet models use strided convolutions (length=9, stride=4) for downsampling x4, and interpolations for upsampling x4. We study variants with sinc, linear, nearest neighbor, and stretch (see section II.B). In section II.E, we hypothesized that allowing spectral replicas accross feature maps can be beneficial. Here, we confirm that stretch and nearest neigbor (allowing high-frequency replicas) obtain better SDR scores than sinc and linear (attenuating high frequencies). Post-networks and normalization do not improve results drastically, for this reason we select the base stretch and nearest neigbor models for a listening test. Finally, the WaveUnets we consider obtain better results that the original (5dB vs. 3dB SDR), thanks to an increase in model size (from 10M to 700M learnable parameters) and a strong use of data augmentation.

D. WaveletUnet: wavelet-based models

WaveletUnet models rely on a cascade of two lazy, Haar or learnable wavelets to downsample/upsample x4 (see section II.D and Fig. 4). The "detail" signals in Fig. 4 contain the skip connections' signal, and the "coarse" signals are processed by the next layer. In Table I, we note that lazy wavelet outperforms Haar wavelet, and that learnable wavelet slighlty underperforms lazy wavelet. Further, post-networks and normalization do not improve SDR results. Hence, we select the base lazy WaveletUnet for a listening test. Finally, and similarly as for the WaveUnets above, the WaveletUnets we consider also obtatin much better results than the original.

TABLE I
BENCHMARKING UPSAMPLING LAYERS FOR SOURCE SEPARATION

Music Source Separation *MUSDB's test-set results*	input (sec)	approx. # parm	SDR (dB)	epoch (sec)
WaveUnet [2]: original publication	6.7	10M	3.23	-
WaveletUnet [7]: original publication	6.7	15M	3.39	-
Demucs [6]: original publication	10	648M	5.34	-
Demucs: partial overlap	12	716M	5.28	316
Demucs: full overlap	12	703M	5.37	311
Demucs: no overlap	12	648M	**5.39**	298
Demucs: no overlap + post-networks	12	648M	**5.39**	296
Demucs: no overlap + normalization	12	648M	5.30	299
WaveShuffle	12	729M	5.38	305
WaveShuffle + post-network	12	729M	5.38	302
WaveShuffle + normalization	12	729M	**5.44**	308
WaveUnet: sinc	12	716M	4.52	553
WaveUnet: linear	12	716M	4.62	430
WaveUnet: nearest neighbor (NN)	12	716M	**5.17**	420
WaveUnet: stretch	12	716M	**5.23**	423
WaveUnet NN + post-network	12	716M	**5.17**	422
WaveUnet NN + normalization	12	716M	5.08	421
WaveUnet stretch + post-network	12	716M	5.23	420
WaveUnet stretch + normalization	12	716M	**5.24**	429
WaveletUnet: lazy wavelet	12	716M	**5.31**	532
WaveletUnet: Haar wavelet	12	716M	4.55	534
WaveletUnet: learnable wavelet	12	716M	5.30	535
Lazy WaveletUnet + post-network	12	716M	**5.31**	530
Lazy WaveletUnet + normalization	12	716M	5.22	534

TABLE II
MEAN OPINION SCORES (MOS, FROM 1 TO 5) REPORTING OVERALL SEPARATION QUALITY OF THE MOST PROMISING UPSAMPLING LAYERS.

Demucs: no overlap	WaveShuffle + norm	WaveUnet: nearest neigh.	WaveUnet: stretch	WaveletUnet: lazy wavelet
2.70	2.48	**3.30**	2.83	2.45

IV. SUBJECTIVE EVALUATION

We further evaluate the most promising models with a listening test. 10 expert listeners evaluate 4 songs: 2 from the Free Music Archive [20], and 2 from the test-set[3]. Here, we evaluate "overall separation quality" (from 1 to 5, the higher the better) via averaging the ratings obtained from individually evaluating each estimated source (Table II). WaveUnets obtain the best MOS ratings, and nearest neighbour is preferred over stretch (t-test: $p<10^{-5}$). This is possibly because stretch can introduce tonal artifacts (due to spectral replicas of offsets), but nearest neighbor cannot (because its filtering artifacts attenuate spectral replicas of offsets)[3]. This denotes that filtering artifacts can be useful to combat the spectral replicas of signal offsets. We also find that stretch is preferred over Demucs no overlap (t-test: $p<10^{-8}$), that Demucs no overlap is preferred over WaveShuffle + norm (t-test: $p=0.03$), and that WaveShuffle + norm and lazy WaveletUnet are the worst rated (and perform equivalently, t-test: $p=0.79$). Further, WaveShuffle + norm (with the best SDR scores) obtains among the worse MOS ratings, and WaveUnet nearest neighbor (with the best MOS ratings) obtains lower SDR scores. This shows that small SDR differences don't translate directly to perceptual improvement. Finally, we want to note that post-neworks and normalization layers were unable to fully remove tonal artifacts.[3]

[3]Listen online: http://jordipons.me/apps/upsamplers/

V. CONCLUSIONS

Tonal artifacts are unpleasant artifacts difficult to tame because two causing factors are simultaneously interacting: the architecture (as for transposed and subpixel convolutions) and the spectral replicas. Further, filtering artifacts are perceptually managable and a convenient tool against tonal artifacts. We note this in our listening test, where nearest neighbor layers were preferred, possibly because these are free of tonal artifacts (since it attenuates bands with spectral replicas of offsets) and allow high-frequency replicas (yielding coherent high-frequency content for wide-band synthesis). Yet, our results also show that many upsamplers can perform comparably (MOS$\approx$3) and, depending on our goals (e.g., low memory footprint or no tonal artifacts), these might be interchangeable.

REFERENCES

[1] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *ICLR*, 2019.

[2] D. Stoller, S. Ewert, and S. Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," in *ISMIR*, 2018.

[3] J. Pons, S. Pascual, G. Cengarle, and J. Serrà, "Upsampling artifacts in neural audio synthesis," in *ICASSP*, 2020.

[4] A. Pandey and D. Wang, "Densely connected neural network with dilated convolutions for real-time speech enhancement in the time domain," in *ICASSP*, 2020.

[5] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, "Melgan: Generative adversarial networks for conditional waveform synthesis," in *NeurIPS*, 2019.

[6] A. Défossez, N. Usunier, L. Bottou, and F. Bach, "Music source separation in the waveform domain," in *arXiv*, 2019.

[7] T. Nakamura and H. Saruwatari, "Time-domain audio source separation based on wave-u-net combined with discrete wavelet transform," in *ICASSP*, 2020.

[8] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," in *arXiv*, 2020.

[9] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and checkerboard artifacts," *Distill*, vol. 1, no. 10, p. e3, 2016.

[10] F. G. Germain, Q. Chen, and V. Koltun, "Speech denoising with deep feature losses," in *Interspeech*, 2019.

[11] J. O. Smith, "Mathematics of the discrete fourier transform (dft): with audio applications," 2007.

[12] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *CVPR*, 2016.

[13] A. Aitken, C. Ledig, L. Theis, J. Caballero, Z. Wang, and W. Shi, "Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize," in *arXiv*, 2017.

[14] W. Sweldens, "The lifting scheme: A construction of second generation wavelets," *SIAM Journal on Mathematical Analysis*, vol. 29, no. 2, pp. 511–546, 1998.

[15] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[16] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE TASLP*, vol. 14, no. 4, pp. 1462–1469, 2006.

[17] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *ICML*, 2017.

[18] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," in *arXiv*, 2016.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *arXiv*, 2014.

[20] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A Dataset For Music Analysis," in *ISMIR*, 2017.