NEURAL WAVESHAPING SYNTHESIS

Ben Hayes, Charalampos Saitis, George Fazekas
Centre for Digital Music, Queen Mary University of London
{b.j.hayes, c.saitis, g.fazekas}@qmul.ac.uk

ABSTRACT

We present the Neural Waveshaping Unit (NEWT): a novel, lightweight, fully causal approach to neural audio synthesis which operates directly in the waveform domain, with an accompanying optimisation (FastNEWT) for efficient CPU inference. The NEWT uses time-distributed multilayer perceptrons with periodic activations to implicitly learn nonlinear transfer functions that encode the characteristics of a target timbre. Once trained, a NEWT can produce complex timbral evolutions by simple affine transformations of its input and output signals. We paired the NEWT with a differentiable noise synthesiser and reverb and found it capable of generating realistic musical instrument performances with only 260k total model parameters, conditioned on F0 and loudness features. We compared our method to state-of-the-art benchmarks with a multi-stimulus listening test and the Fréchet Audio Distance and found it performed competitively across the tested timbral domains. Our method significantly outperformed the benchmarks in terms of generation speed, and achieved real-time performance on a consumer CPU, both with and without FastNEWT, suggesting it is a viable basis for future creative sound design tools.

1. INTRODUCTION

Synthesisers are indispensable tools in modern music creation. Over the last six decades, their evolving sonic affordances have defined uncountable musical aesthetics and cultures, enabling composers, sound designers, and musicians to interact with human auditory perception in previously impossible ways.

The recent proliferation of deep neural networks as audio synthesisers is further expanding the capabilities of these tools: realistic instrument performances can be synthesised from simple, low dimensional control signals [1–3]; the timbre of one instrument can be convincingly transferred to another [1, 3–5]; instruments can be morphed and interpolated along nonlinear manifolds [6,7]; and sounds can be manipulated using high level descriptors of perceptual characteristics [7–9]. Yet despite their impressive abilities, these systems have not been widely adopted in music creation workflows.

We argue that this is largely a pragmatic issue. Modern music production centres around the digital audio workstation (DAW), with software instruments and signal processors represented as real-time plugins. These allow users to dynamically manipulate and audition sounds, responsively tweaking parameters as they listen or record. Neural audio synthesisers do not currently integrate elegantly with this environment, as they rely on deep neural networks with millions of parameters, and are often incapable of functioning in real-time on a CPU.

In this work we move towards integrating the benefits of neural audio synthesis into creative workflows with a novel, lightweight architecture built on the principles of digital waveshaping synthesis [10]. Our model implicity learns a bank of continuous differentiable waveshapers, which are applied to an exciter signal. A control module learns to generate time-varying timbres by dynamically shifting and scaling the learnt waveshaper's input and output. As the waveshapers encode information about the target timbre, our model can synthesise convincing audio using an order of magnitude fewer parameters than the current state-of-the-art methods.

This paper is laid out as follows. In section 2 we discuss related work on neural audio synthesis and waveshaping. Section 3 introduces our architecture, and we outline our training methodology in section 4. In section 5 we present and discuss evaluations of our model in comparison to the current state of the art methods [1,3]. Finally, we conclude with suggestions for future work in section 6. We provide full source code [1] and encourage readers to listen to the audio examples in the online supplement [2].

2. RELATED WORK

2.1 Neural Audio Synthesis

Audio synthesis with deep neural networks has received considerable attention in recent years. Autoregressive models such as WaveNet [11] and SampleRNN [12] defined a class of data-driven, general-purpose vocoder, which was subsequently expanded on with further probabilistic approaches, including flow-based models [13–15] and generative adversarial networks [16–19]. These models allow realistic synthesis of speech, and applications to musical audio [6, 20, 21] have yielded similarly impressive results. A parallel stream of research has focused on controllable musical audio synthesis [1–3, 7, 8, 22], in which

[1] https://github.com/ben-hayes/neural-waveshaping-synthesis
[2] https://ben-hayes.github.io/projects/nws/

models are designed to provide control affordances that may be of practical use. Such controls have included MIDI scores [2, 22], semantic or acoustical descriptors of timbre [7, 8], and F0/loudness signals [1, 3]. The representations of timbre learnt by these models have also been observed to show similarities to human timbre perception [23].

A recent category of model, [1, 3, 24] unified under the conceptual umbrella of differentiable digital signal processing (DDSP) [1], has enabled low-dimensional, interpretable control through strong inductive biases to audio synthesis. Whereas generalised neural vocoders must learn from scratch to produce the features that typify audio signals, such as periodicity and harmonicity, DDSP methods utilise signal processing components designed to produce signals exhibiting such features. These components are expressed as differentiable operations directly in the computation graph, effectively constraining a model's outputs to a subspace defined by the processor's capabilities.

DDSP methods fall into two groups: those where the network generates control signals for a processor, and those where the network is trained to be a signal processor itself. The DDSP autoencoder [1] falls into the first category as it generates control signals for a spectral modelling synthesiser [25]. The neural source-filter (NSF) approach [3, 24, 26] is in the second category. It learns a nonlinear filter that transforms a sinusoidal exciter to a target signal, guided by a control embedding generated by a separate encoder. In other words: the control module "plays" the filter network.

The NSF filter network transforms its input through amplitude distortion, as each activation function acts as a nonlinear waveshaper. A given layer's ability to generate a target spectrum is thus bounded by the distortion characteristics of its activation function. For this reason, neural source-filter models are typically very deep: Wang et al.'s simplified architecture [24] requires 50 dilated convolutional layers, and Michelashvili & Wolf's musical instrument model [3] consists of 120 dilated convolutional layers – 30 for each of its four serial generators.

Our method avoids the need for such depth by learning continuous representations of detailed waveshaping functions as small multilayer perceptrons. These functions are optimised such that their amplitude distortion characteristics allow them to produce spectral profiles appropriate to the target timbre. This allows our model to accurately transform an exciter signal considerably more efficiently, whilst still exploiting the benefits of the network-as-synthesiser approach.

2.2 Digital Waveshaping Synthesis

In *waveshaping synthesis* [10], timbres are generated using the amplitude distortion properties of a nonlinear shaping function $f : \mathbb{R} \mapsto \mathbb{R}$, which is memoryless and shift invariant. Due to its nonlinearity, f is able to introduce new frequency components to a signal [27]. When a pure sinusoid $\cos \omega n$ is used as the input to f, only pure harmonics are introduced to the signal. An exciter signal with multiple frequency components, conversely, would result in intermodulation distortion, generating components at frequencies $a\omega_1 \pm b\omega_2,\ \forall a, b \in \mathbb{Z}^+$, for input frequencies ω_1 and ω_2. This would result in inharmonic components if ω_1 and ω_2 are not harmonically related.

The shaping function f is designed to produce a specific spectral profile when excited with $\cos \omega n$. This is achieved as a weighted sum of Chebyshev polynomials of the first kind, which possess the property that the kth polynomial T_k directly transforms a sinusoid to its kth harmonic: $T_k(\cos \omega n) = \cos \omega k n$. With a function specified in this way, we can define a simple discrete time waveshaping synthesiser

$$x[n] = N[n] f(a[n] \cos \omega n), \tag{1}$$

where $a[n]$ is the distortion index and $N[n]$ is a normalising coefficient. As the frequency components generated by a nonlinear function vary with input amplitude, varying the distortion index over time allows us to generate evolving timbres, whilst the normalising coefficient allows us to decouple the frequency content and overall amplitude envelope of the signal.

3. NEURAL WAVESHAPING SYNTHESIS

Our model acts as a harmonic-plus-noise synthesiser [25]. This architecture separately generates periodic and aperiodic components and exploits an inductive bias towards harmonic signals. Fig. 1 illustrates the overall architecture of our model.

3.1 Control Encoder

We condition our model on framewise control signals extracted from the target audio with a hop size of 128. We project these to a 128-dimensional control embedding z using a causal gated recurrent unit (GRU) of hidden size 128 followed by a time distributed dense layer of the same size. We leave the exploration of the performance of alternative sequence models to future work.

3.2 NEWT: Neural Waveshaping Unit

The shaping function f of a waveshaping synthesiser can be fit to only a single instantaneous harmonic spectrum. The spectral evolution afforded by the distortion index $a[n]$ is thus usually unrelated to the target timbre. This is a limitation of the Chebyshev polynomial method of shaping function design. Here, we propose to instead learn a shaping function f_θ parameterised by a multilayer perceptron (MLP). As demonstrated in recent work on implicit neural representations [28, 29], MLPs with sinusoidal activations dramatically outperform ReLU MLPs in learning continuous representations of detailed functions with arbitrary support. We therefore use sinusoidal activations in f_θ, which enables useful shaping functions to be learnt by very compact networks. Here, we use 64 parallel shaper MLPs, each with 4 layers, with a hidden size of 8 neurons.

To enable our model to fully exploit the distortion characteristics of f_θ, we replace the distortion index $a[n]$ and normalising coefficient $N[n]$ with affine transforms before

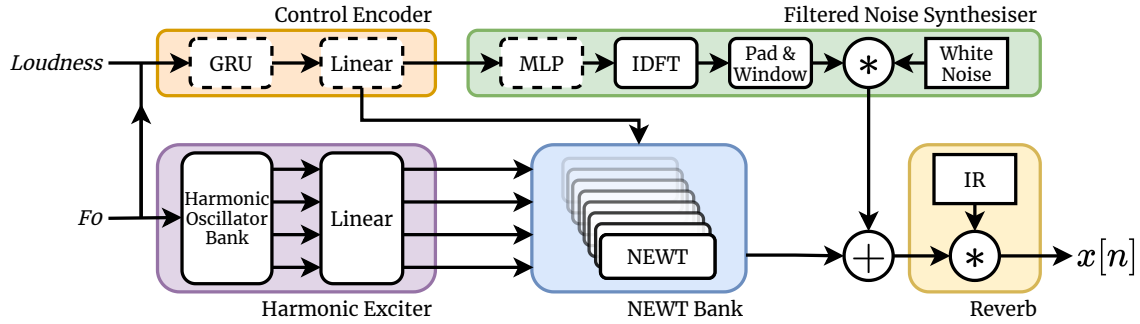

Figure 1. The full architecture of our neural audio synthesiser. All linear layers and MLPs are time distributed. Convolution is denoted $*$ and applied by multiplication in the frequency domain. Blocks with dashed outlines operate at the same coarse time steps as the control signal, whilst those with solid outlines operate at audio rate.

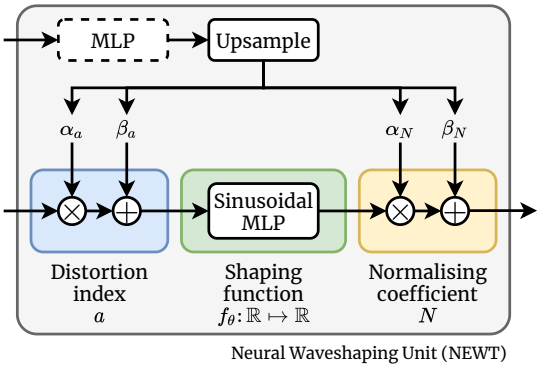

Figure 2. A block diagram depicting the structure of the neural waveshaping unit (NEWT). Blocks with dashed outlines operate at control signal time steps, whilst solid blocks operate at audio rate.

and after the shaping function. The parameters of these transforms, denoted α_a and β_a for the distortion index and α_N and β_N for the normalising coefficient, are generated by a separate MLP (depth 4, width 128, ReLU activations with layer normalisation [30]) which takes z as input, and then upsampled to audio rate. The output of a single NEWT in response to exciter signal $y[n]$ is thus given by:

$$x[n] = \alpha_N f_\theta(\alpha_a y[n] + \beta_a) + \beta_N. \tag{2}$$

In this way, the NEWT disentangles two tasks: it learns a synthesiser parameterised by $(\alpha_a, \alpha_N, \beta_a, \beta_N)$, and it learns to "play" that synthesiser in response to a control signal z. Fig. 2 illustrates the structure of the NEWT. In practice, we use multiple such units in parallel. We can implement this efficiently using grouped 1-dimensional convolutions with a kernel size of 1 — essentially a bank of parallel time-distributed dense layers.

3.3 FastNEWT

The NEWT is an efficient approach to generating time-varying timbres, but its reliance on grouped 1-dimensional convolutions best suits it to GPU inference. Many use-cases for our model do not guarantee the availability of a GPU, and so efficient CPU inference is of crucial importance. For this reason, we propose an optimisation called the *FastNEWT*: as each learnable shaping function simply maps $\mathbb{R} \mapsto \mathbb{R}$, it can be replaced by a lookup table of arbitrary resolution. Forward passes through f_θ are then simply replaced with the $\mathcal{O}(1)$ operation of reading values from an array and calculating an interpolation.

To produce a *FastNEWT*, we sample f_θ across a closed interval. The sampling resolution and interval are tunable parameters of this operation, and represent a trade-off between memory cost and reconstruction quality. Here, we opt for a lookup table of 4096 samples over the interval $[-3, 3]$, using a naïve implementation with linear interpolation. Like the rest of our model, this is implemented using PyTorch operations, and so we treat this as an upper bound on the computational cost of the *FastNEWT*. In practice, an implementation in a language with low level memory access would confer performance improvements.

3.4 Harmonic Exciter

To reduce the resolution required of the shaping functions, we produce our exciter with a harmonic oscillator bank generating up to 101 harmonics, truncated at the Nyquist frequency. The outputs of this oscillator bank are passed through a time distributed linear layer, acting as a mixer which provides each NEWT channel with a weighted mixture of harmonics. Thus, the ith output channel of the exciter module is given by:

$$y_i[n] = \sum_{k=1}^{K} A(k\omega) w_{ik} \cos k\omega n + b_i, \tag{3}$$

where the antialiasing mask $A(k\omega)$ is 1 if $-\pi < k\omega < \pi$ and 0 otherwise.

3.5 Noise Synthesiser

In spectral modelling synthesis [25], audio signals are decomposed into a harmonic portion and a residual portion. The residual portion is typically modelled by filtered noise, with filter coefficients varying over time according to the spectrum of the residual. Here, we use an MLP (depth 4, hidden size 128, ReLU activations with layer normalisation) to generate 256-tap FIR filter magnitude responses conditioned on z. We apply a differentiable window-design method like that used in the DDSP model [1] to

apply the filters to a white noise signal. First, we take the inverse DFT of these magnitude responses, then shift them to causal form, and apply a Hann window to the impulse response. We then apply the filters to a white noise signal by multiplication in the frequency domain.

3.6 Learnable Reverb

To model room acoustics, we apply a differentiable convolutional reverb to the signal. We use an impulse response $c[n]$ of length 2 seconds, initialised as follows:

$$c[n] \begin{cases} \sim \mathcal{N}(0; \text{1e-6}), & \text{if } n > 1, \\ = 0, & \text{if } n = 0. \end{cases} \tag{4}$$

$c[n]$ is trainable for $n \geq 1$, whilst the 0th value is fixed at 0. The reverberated signal $(c * x)[n]$ is computed by multiplication in the frequency domain, and the output of the reverb is summed with the dry signal.

4. EXPERIMENTS

Our model can be trained directly through maximum likelihood estimation with minibatch gradient descent. Here we detail the training procedure used in our experiments.

4.1 Loss

We trained our model using the multi-resolution STFT loss from [18]. A single scale of the loss is defined as the expectation of the sum of two terms. The first is the spectral convergence L_{sc} (Eqn. 5) and the second is log magnitude distance L_{m} (Eqn. 6), defined as:

$$L_{\text{sc}}(x, \hat{x}) = \frac{\left\| |STFT_m(x)| - |STFT_m(\hat{x})| \right\|_F}{\left\| |STFT_m(x)| \right\|_F} \tag{5}$$

and

$$L_{\text{m}}(x, \hat{x}) = \frac{1}{m} \left\| \log |STFT_m(x)| - \log |STFT_m(\hat{x})| \right\|_1 \tag{6}$$

respectively, where $\|\cdot\|_F$ is the Frobenius norm, $\|\cdot\|_1$ is the L1 norm, and $STFT_m$ gives the short-time Fourier transform with analysis window of length m for $m \in \{512, 1024, 2048\}$. We used the implementation of this loss provided in the *auraloss* library [31].

4.2 Data

We collated monophonic audio files from three instruments (violin, trumpet, & flute) from across the University of Rochester Music Performance (URMP) dataset [32], and for each instrument applied the following preprocessing. We normalised amplitude across each instrument subset, made all audio monophonic by retaining the left channel, and resampled to 16kHz. We extracted F0 and confidence signals using the full CREPE model [33] with a hop size of 128 samples. We extracted A-weighted loudness using the procedure laid out in [21] using a window of 1024 samples and a hop size of 128 samples. We divided audio and control signals into 4 second segments, and discarded any segment with a mean pitch confidence < 0.85. Finally, control signals were standardised to zero mean and unit variance. Each instrument subset was then split into 80% training, 10% validation, and 10% test subsets.

Model	Parameters
HTP	5.6M
DDSP-full	6M
DDSP-tiny	280k*
NWS	*266k*

* The paper reports 240k [1], but the official implementation contains a model with 280k parameters.

Table 1. Trainable parameter counts of models under comparison.

4.3 Training

We trained our models with the Adam optimiser using an initial learning rate of 1e-3. The learning rate was exponentially decayed every 10k steps by a factor of 0.9. We clipped gradients to a maximum norm of 2.0. All models were trained for 120k iterations with a batch size of 8.

5. EVALUATION & DISCUSSON

To evaluate the performance of our model across different timbres, we trained a neural waveshaping model for each instrument subset. We denote these models *NWS*, specifying the instrument where relevant. After training, we created optimised models with *FastNEWT*, denoted *NWS-FN*, and included these in our experiments also.

5.1 Benchmarks

We evaluated our models in comparison to two state of the art methods: DDSP [1] and Hierarchical Timbre Painting (referred to from here as *HTP*) [3]. We trained these on the same data splits as our model, preprocessed in accordance with each benchmark's requirements.

Two DDSP architectures were used as benchmarks: the "full" model used to train a solo violin synthesiser in the original paper, and the "tiny" model described in the paper's appendices. Each model was trained for 30k iterations as recommended in the supplementary materials. We denote these *DDSP-full* and *DDSP-tiny*, respectively. HTP consists of four distinct Parallel WaveGAN [18] generators operating at increasing timescales. We trained each for 120k iterations, as recommended in the original paper. Table 1 lists the total trainable parameter counts of all models under comparison.

5.2 Fréchet Audio Distance

The Fréchet Audio Distance (FAD) is a metric originally designed for evaluating music enhancement algorithms [34], which correlates well with perceptual ratings of audio quality. It is computed by fitting multivariate Gaussians to embeddings generated by a pretrained VGGish model [35]. This process is performed for both the set under evaluation,

Model	Fréchet Audio Distance		
	Flute	Trumpet	Violin
Test Data	0.463	0.327	0.096
HTP	6.970	14.848	2.529
DDSP-full	3.091	**1.391**	**1.062**
DDSP-tiny	3.673	5.301	*2.454*
NWS	**2.704**	*2.158*	5.101
NWS-FN	*2.717*	2.163	5.091

Table 2. Fréchet Audio Distance scores for all models using background embeddings computed across each instrument's full dataset. Bold type indicates the best performance in a column and italics the second best.

yielding $\mathcal{N}_e(\mu_e, \Sigma_e)$, and a set of "background" audio samples which represent desirable audio characteristics, yielding $\mathcal{N}_b(\mu_b, \Sigma_b)$. The FAD is then given by the Fréchet distance between these distributions:

$$F(\mathcal{N}_b, \mathcal{N}_e) = \|\mu_b - \mu_e\|^2 + tr(\Sigma_b + \Sigma_e - 2\sqrt{\Sigma_b \Sigma_e}). \quad (7)$$

Thus, a lower FAD score indicates greater similarity to the background samples in terms of the features captured by the VGGish embedding. Here, we used the FAD to evaluate the overall similarity of our model's output to the target instrument. We computed our background embedding distribution $\mathcal{N}_b$ from each instrument's full dataset, whilst the evaluation embedding distributions $\mathcal{N}_e$ were computed using audio resynthesised from the corresponding test set. FAD scores for our model, all benchmarks, and the test datasets themselves are presented in Table 2.

In general, the closely matched scores of the NWS and NWS-FN models indicate that, across instruments, the *FastNEWT* optimisation has a minimal effect on this metric of audio quality. On trumpet and flute, our models consistently outperform HTP and DDSP-tiny, and also outperform DDSP-full on flute. On violin, conversely, both DDSP models are the best performers, with HTP achieving a similar score to DDSP-tiny.

5.3 Listening Test

Our model and benchmarks can be considered as highly specified audio codecs. We therefore applied a listening test inspired by the MUSHRA (MUltiple Stimuli with Hidden Reference and Anchor) standard [36], which is used to assess the perceptual quality of audio codecs. We used the webMUSHRA framework [37], adapted to incorporate a headphone screening test [38]. For each instrument, we selected two stimuli from the test set representing distinct register and articulation, giving six total trials. In each trial, we used the original recording as the reference and produced the anchor by applying a 1kHz low pass filter. We recruited 19 participants from a pool of audio researchers, musicians, and audio engineers. We excluded the responses of one participant, who rated the anchor above the reference in greater than 15% of trials. Responses for each trial are plotted in Fig. 3. In general, NWS and NWS-FN performed similarly across trials,

Model	Real-time Factor			
	GPU		CPU	
	Mean	*90th Pctl.*	*Mean*	*90th Pctl.*
HTP	0.105	0.106	2.203	2.252
DDSP-full	0.038	0.047	0.363	0.395
DDSP-tiny	0.032	0.039	0.215	0.223
NWS	*0.004*	*0.004*	*0.194*	*0.208*
NWS-FN	**0.003**	**0.003**	**0.074**	**0.076**

Table 3. Real-time time factor computed by synthesising four seconds of audio in a single forward pass across all benchmarks. Statistics computed over 100 runs. Bold type indicates the best performance in a column and italics the second best.

suggesting that FastNEWT has little, if any, impact on the perceptual quality of the synthesised audio. Across flute and trumpet trials our models were rated similarly to the benchmarks. In the first violin trial, our models' ratings were similar to those of DDSP-tiny, whilst in the second they were lowest overall. These ratings are concordant with FAD scores: our model performs competitively on trumpet and flute whilst struggling somewhat with violin.

To examine the influence of melodic stimuli on participants' ratings, we performed Wilcoxon's signed-rank test between scores given for each instrument's two stimuli, for each synthesis model. For example, scores given to DDSP-full for stimulus Flute 1 were compared to scores given to DDSP-full for Flute 2. Out of fifteen tests, significant differences ($p < .001$) were observed in two: between trumpet stimuli for both DDSP-full and HTP. No other significant effects were observed ($\alpha = 0.05$).

To examine the effect of synthesis model, we performed Friedman's rank sum test on ratings from each trial. For flute stimuli, no significant effects were found. Significant effects were observed for both trumpet stimuli, although Kendall's W suggested only weak agreement between raters (Trumpet 1: $Q = 27.45, p < 0.001, W = 0.38$; Trumpet 2: $Q = 14.18, p < 0.01, W = 0.20$) . Both violin stimuli also resulted in significant effects with moderate agreement between raters (Violin 1: $Q = 42.28, p < 0.001, W = 0.59$; Violin 2: $Q = 37.95, p < 0.001, W = 0.53$). Post-hoc analysis was performed within each trial using Wilcoxon's signed-rank test with Bonferroni p-value correction. Significant differences (corrected threshold $p < .005$) were observed for Trumpet 1, Violin 1, and Violin 2. These are illustrated as brackets in Fig. 3.

5.4 Real-time Performance

We evaluated the real-time performance of our model in two scenarios. In both cases we took measurements on a GPU (Tesla P100-PCIe 16GB) and a CPU (Intel i5 1038NG7 2.0GHz) and used the real-time factor (RTF) as a metric. The RTF is defined as

$$RTF \coloneqq \frac{t_p}{t_i}, \quad (8)$$

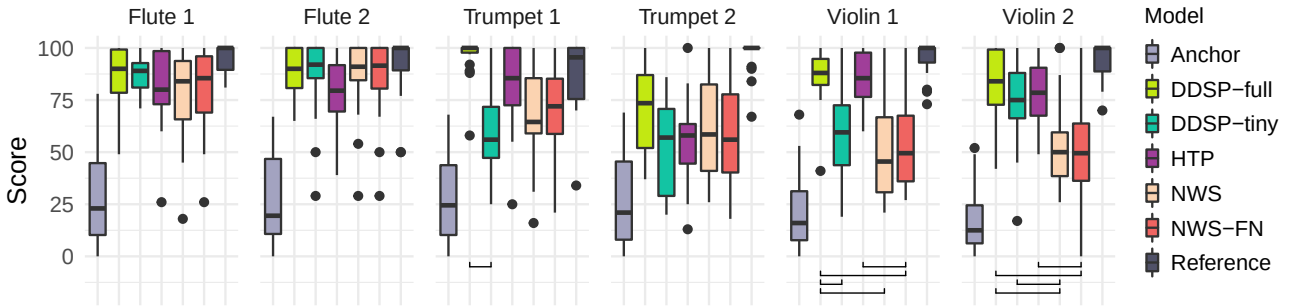

Figure 3. Boxplots of ratings given to each synthesis model during each trial in our listening test. Brackets indicate significant (corrected $p < .005$) differences in pairwise Wilcoxon signed-rank tests with Bonferroni correction.

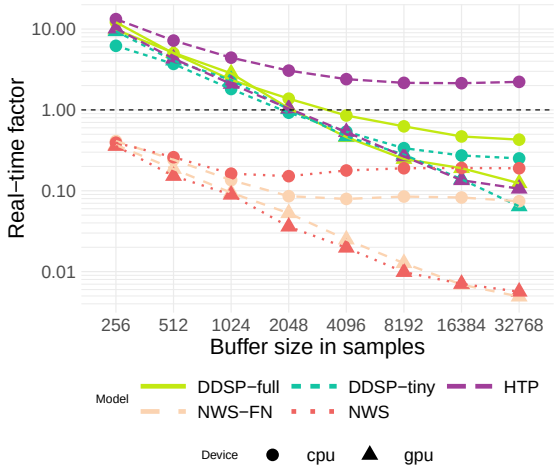

Figure 4. A plot of the mean real-time factor against buffer size across all benchmarks. Mean computed over 100 runs per model per device per buffer size.

where t_i is the temporal duration of the input and t_p is the time taken to process that input and return an output. Real-time performance thus requires $RTF < 1$. In all tests we computed RTF statistics over 100 measurements.

The first scenario models applications where an output is expected immediately after streaming an input. To test this, we computed the RTF on four second inputs. We report the mean and 90th percentile in Table 3. On the GPU, NWS and NWS-FN outperformed all benchmarks, including DDSP-tiny. On the CPU, NWS still outperformed all other models, albeit by a narrower margin. The benefit of the *FastNEWT* optimisation was clearer on CPU: NWS-FN had a mean RTF 2.9× lower than the best performing benchmark. On both platforms, HTP was significantly slower, likely due to its considerable depth.

The second scenario assumes applications where immediate response to input is expected, such as in a software instrument. Here, samples are processed in blocks to ensure that sufficient audio is delivered to the DAC in time for playback. We computed the RTF for each buffer size in $B := \{2^n \mid n \in \mathbb{Z}, 8 \leq n < 16\}$. The means of these runs are plotted in Fig. 4. Again, NWS and NWS-FN outperformed all benchmarks on both CPU and GPU, sitting comfortably below the real-time threshold of 1.0 at all tested buffer sizes. HTP did not achieve real-time performance at any buffer size on the CPU, and only did so for buffer sizes over 2048 on the GPU. DDSP-full, similarly, was unable to achieve realtime performance for buffer sizes of 2048 or lower on GPU or CPU, while DDSP-tiny sat on the threshold at this buffer size. It should be noted that a third-party, stripped down implementation of the DDSP model was recently released, which is capable of real-time inference when the convolutional reverb module is removed [3] .

6. CONCLUSION

In this paper, we presented the NEWT: a neural network structure for audio synthesis based on the principles of waveshaping [10]. We also present full source code, pretrained checkpoints, and an online supplement containing audio examples. Our architecture is lightweight, causal, and comfortably achieves real-time performance on both GPU and CPU, with efficiency further improved by the *FastNEWT* optimisation. It produces convincing audio directly in the waveform domain without the need for hierarchical or adversarial training. Our model is also capable of many-to-one timbre transfer by extracting F0 and loudness control signals from the source audio. Examples of this technique are provided in the online supplement, where we also offer insight into the specific shaping functions learned by the NEWT.

In evaluation with a multi-stimulus listening test and the Fréchet audio distance our model performed competitively with state-of-the-art methods with over 20× more parameters on trumpet and flute timbres, whilst performing similarly to a comparably sized DDSP benchmark on violin timbres. We suspect the lower scores on violin timbres were due to the greater proportion of harmonic energy at higher frequencies in these sounds. The NEWT may thus have failed to learn shaping functions capable of producing these high harmonics without introducing aliasing artefacts. Using deeper or wider MLPs inside the NEWT may allow more accurate shaping functions to be learnt, whilst retaining efficient inference with FastNEWT. Future work will investigate this and other differentiable antialiasing strategies, including adaptive oversampling [39]. We will also explore extending our model to multi-timbre synthesis.

[3] https://github.com/acids-ircam/ddsp_pytorch

7. ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers at *ISMIR* for their thoughtful comments. We would also like to thank our colleague Cyrus Vahidi for many engaging and insightful discussions on neural audio synthesis. This work was supported by UK Research and Innovation [grant number EP/S022694/1].

8. REFERENCES

[1] J. Engel, L. H. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable Digital Signal Processing," in *8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.

[2] J. W. Kim, R. Bittner, A. Kumar, and J. P. Bello, "Neural Music Synthesis for Flexible Timbre Control," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, Brighton, United Kingdom, 2019, pp. 176–180.

[3] M. M. Michelashvili and L. Wolf, "Hierarchical Timbre-painting and Articulation Generation," in *Proceedings of the 21th International Society for Music Information Retrieval Conference*, Oct. 2020.

[4] S. Huang, Q. Li, C. Anil, S. Oore, and R. B. Grosse, "TimbreTron A WaveNet(CycleGAN(CQT(Audio))) Pipeline for Musical Timbre Transfer," in *7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019, p. 17.

[5] D. K. Jain, A. Kumar, L. Cai, S. Singhal, and V. Kumar, "ATT: Attention-based Timbre Transfer," in *2020 International Joint Conference on Neural Networks (IJCNN)*. Glasgow, United Kingdom: IEEE, Jul. 2020, pp. 1–6.

[6] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with WaveNet autoencoders," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, Sydney, Australia, Aug. 2017, pp. 1068–1077.

[7] P. Esling, A. Chemla, and A. Bitton, "Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, Paris, France, 2018, pp. 175–181.

[8] P. Esling, N. Masuda, A. Bardet, R. Despres, and A. Chemla-Romeu-Santos, "Flow Synthesizer: Universal Audio Synthesizer Control with Normalizing Flows," *Applied Sciences*, vol. 10, no. 1, p. 302, 2020.

[9] J. Nistal, S. Lattner, and G. Richard, "DrumGAN: Synthesis of Drum Sounds With Timbral Feature Conditioning Using Generative Adversarial Networks," in *Proceedings of the 21th International Society for Music Information Retrieval Conference*, Montréal, Aug. 2020.

[10] M. Le Brun, "Digital Waveshaping Synthesis," *Journal of the Audio Engineering Society*, vol. 27, no. 4, pp. 250–266, Apr. 1979.

[11] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," *arXiv:1609.03499 [cs]*, Sep. 2016, arXiv: 1609.03499.

[12] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, "SampleRNN: An Unconditional End-to-End Neural Audio Generation Model," in *5th International Conference on Learning Representations*, Toulon, France, 2017.

[13] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A Flow-based Generative Network for Speech Synthesis," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, United Kingdom: IEEE, May 2019, pp. 3617–3621.

[14] W. Song, G. Xu, Z. Zhang, C. Zhang, X. He, and B. Zhou, "Efficient WaveGlow: An Improved WaveGlow Vocoder with Enhanced Speed," in *Interspeech 2020*. ISCA, Oct. 2020, pp. 225–229.

[15] A. v. d. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. v. d. Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis, "Parallel WaveNet: Fast High-Fidelity Speech Synthesis," *arXiv:1711.10433 [cs]*, Nov. 2017, arXiv: 1711.10433.

[16] 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 *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.

[17] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 17 022–17 033.

[18] R. Yamamoto, E. Song, and J.-M. Kim, "Parallel Wavegan: A Fast Waveform Generation Model Based on Generative Adversarial Networks with Multi-Resolution Spectrogram," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech*

and Signal Processing (ICASSP). Barcelona, Spain: IEEE, May 2020, pp. 6199–6203.

[19] C. Donahue, J. McAuley, and M. Puckette, "Adversarial Audio Synthesis," in *7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019, p. 16.

[20] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial Neural Audio Synthesis," in *7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019, p. 17.

[21] 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*, Delft, The Netherlands, 2019, pp. 524–530.

[22] N. Jonason, B. L. T. Sturm, and C. Thome, "The control-synthesis approach for making expressive and controllable neural music synthesizers," in *Proceedings of the 2020 AI Music Creativity Conference*, 2020, p. 9.

[23] B. Hayes, L. Brosnahan, C. Saitis, and G. Fazekas, "Perceptual Similarities in Neural Timbre Embeddings," in *DMRN+15: Digital Music Research Network One-day Workshop 2020*, London, UK, 2020.

[24] X. Wang, S. Takaki, and J. Yamagishi, "Neural Source-Filter Waveform Models for Statistical Parametric Speech Synthesis," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 402–415, 2020.

[25] X. Serra and J. Smith, "Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.

[26] Y. Zhao, X. Wang, L. Juvela, and J. Yamagishi, "Transferring Neural Speech Waveform Synthesizers to Musical Instrument Sounds Generation," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Barcelona, Spain: IEEE, May 2020, pp. 6269–6273.

[27] J. D. Reiss and A. P. McPherson, "Overdrive, Distortion, and Fuzz," in *Audio effects: theory, implementation and application*. Boca Raton London New York: CRC Press, Taylor & Francis Group, 2015, pp. 167–188, oCLC: 931666647.

[28] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit Neural Representations with Periodic Activation Functions," *arXiv:2006.09661 [cs, eess]*, Jun. 2020, arXiv: 2006.09661.

[29] D. W. Romero, A. Kuzina, E. J. Bekkers, J. M. Tomczak, and M. Hoogendoorn, "CKConv: Continuous Kernel Convolution For Sequential Data," *arXiv:2102.02611 [cs]*, Feb. 2021, arXiv: 2102.02611.

[30] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," *arXiv:1607.06450 [cs, stat]*, Jul. 2016, arXiv: 1607.06450.

[31] C. J. Steinmetz and J. D. Reiss, "auraloss: Audio focused loss functions in PyTorch," in *Digital music research network one-day workshop (DMRN+15)*, 2020.

[32] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, "Creating a Multitrack Classical Music Performance Dataset for Multimodal Music Analysis: Challenges, Insights, and Applications," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 522–535, Feb. 2019.

[33] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A Convolutional Representation for Pitch Estimation," in *2018 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2018 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., Sep. 2018, pp. 161–165.

[34] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet Audio Distance: A Reference-Free Metric for Evaluating Music Enhancement Algorithms," in *Interspeech 2019*. ISCA, Sep. 2019, pp. 2350–2354.

[35] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. New Orleans, LA: IEEE, Mar. 2017, pp. 131–135.

[36] I.-R. BS.1534-3, "Method for the subjective assessment of intermediate quality level of audio systems," ITU-R, Tech. Rep., 2015.

[37] M. Schoeffler, S. Bartoschek, F.-R. Stöter, M. Roess, S. Westphal, B. Edler, and J. Herre, "webMUSHRA — A Comprehensive Framework for Web-based Listening Tests," *Journal of Open Research Software*, vol. 6, p. 8, Feb. 2018.

[38] A. E. Milne, R. Bianco, K. C. Poole, S. Zhao, A. J. Oxenham, A. J. Billig, and M. Chait, "An online headphone screening test based on dichotic pitch," *Behavior Research Methods*, Dec. 2020.

[39] B. De Man and J. D. Reiss, "Adaptive control of amplitude distortion effects," in *Audio engineering society conference: 53rd international conference: Semantic audio*, Jan. 2014.