SynthNet: Learning to Synthesize Music End-to-End

Florin Schimbinschi¹, Christian Walder², Sarah M. Erfani¹ and James Bailey¹

¹The University of Melbourne
²Data61 CSIRO, Australian National University
florinsch@student.unimelb.edu.au, christian.walder@data61.csiro.au, sarah.erfani@unimelb.edu.au, baileyj@unimelb.edu.au

Abstract

We consider the problem of learning a mapping directly from annotated music to waveforms, bypassing traditional single note synthesis. We propose a specific architecture based on WaveNet, a convolutional autoregressive generative model designed for text to speech. We investigate the representations learned by these models on music and conclude that mappings between musical notes and the instrument timbre can be learned directly from the raw audio coupled with the musical score, in binary piano roll format. Our model requires minimal training data (9 minutes), is substantially better in quality and converges 6 times faster in comparison to strong baselines in the form of powerful text to speech models. The quality of the generated waveforms (generation accuracy) is sufficiently high, that they are almost identical to the ground truth. Our evaluations are based on both the RMSE of the Constant-Q transform, and mean opinion scores from human subjects. We validate our work using 7 distinct synthetic instrument timbres, real cello music and also provide visualizations and links to all generated audio.

1 Introduction

WaveNets [Van Den Oord et al., 2016] have revolutionized text to speech by producing realistic human voices. Even though the generated speech sounds natural, upon a closer inspection the waveforms are different to genuine recordings. As a natural progression, we propose a WaveNet derivative called SynthNet which can learn and render (in a controlled way) the complex harmonics in the audio training data, to a high level of fidelity. While computer vision is well established, there is little understanding over what audio generative models are learning. Towards enabling similar progress, we give insights into the learned representations of WaveNets, upon which we build our model.

WaveNets were trained using raw audio waveforms aligned with linguistic features. We take a similar approach to learning music synthesizers and train our model based on the raw audio waveforms of entire songs and their symbolic representation of the melody. This is more challenging than speech due to the following differences: 1) in musical compositions multiple notes can be played at the same time, while words are spoken one at a time; 2) the timbre of a musical instrument is arguably more complex than speech; 3) semantically, utterances in music can span over a longer time.

[Van Den Oord et al., 2016] showed that WaveNets can generate new piano compositions based on raw audio. Recently, this work was extended by [Dieleman et al., 2018], delivering a higher consistency in compositional styling. Closer to our work, [Engel et al., 2017] describe a method for learning synthesizers based on individually labelled note-waveforms. This is a laborious task and is impractical for creating synthesizers from real instruments. Our method bypasses this problem since it can directly use audio recordings of an artist playing a given song, on the target instrument.

SynthNet can learn representations of the timbre of a musical instrument more accurately and efficiently via the dilated blocks through depthwise separable convolutions. We show that it is enough to condition only the first input layer, where a joint embedding between notes and the corresponding fundamental frequencies is learned. We remove the skip connections and instead add an additional loss for the conditioning signal. We also use an embedding layer for the audio input and use SeLU [Klambauer et al., 2017] activations in the final block.

The benchmarks against the WaveNet [Van Den Oord et al., 2016] and DeepVoice [Arik et al., 2017] architectures show that our method trains faster and produces high quality audio. After training, SynthNet can generate new audio waveforms in the target timbre, based on a given song which was not seen at training time. While we focus on music, SynthNet can be applied to any time-series domain.

Our contributions are as follows: 1) We show that musical instrument synthesizers can be learned end-to-end based on real and synthetic raw audio and a binary note representation, with minimal training data. Multiple instruments can be learned by a single model, including real instruments (subsection 4.5). 2) We give insights into the representations learned by dilated causal convolutional blocks and consequently propose SynthNet, which provides substantial improvements in quality and training time. Indeed, we demonstrate (Figure 3) that the generated audio is practically identical to the ground truth. 3) The benchmarks against existing architectures contains an extensive set of experiments spanning over three sets of hyperparameters, where we control for the receptive field size. We show that the RMSE of the Constant-Q Transform...
(RMSE-CQT) is highly correlated with the mean opinion score (MOS). 4 We find that reducing quantization error via dithering is a critical preprocessing step towards generating the correct melody and learning the correct pitch to fundamental frequency mapping.

2 Related Work

In music, style can be defined as the holistic combination of the melodic, rhythmic and harmonic components of a particular piece. The delay and sustain variation between notes determines the rhythmic style. The latter can vary over genres (e.g. Jazz vs Classical) or composers. Timbre or harmonic style can be defined as the short term and steady state (sustained) characteristics of these architectures by benchmarking them in the domain of learning music synthesizers. There are considerable differences between the human voice and musical instruments. Firstly, the harmonic complexity of musical instruments is higher than the human voice. Second, even for single instrument music, multiple notes can be played at the same time. This is not true for speech, where only one sound utterance is produced at a time. Lastly, the melodic and rhythmic components in a musical piece span a larger temporal context than a series of phonemes as part of speech. All of these factors make the music domain arguably more challenging than speech.

3.1 Baseline Architectures

Our starting point is the model proposed by [Van Den Oord et al., 2016] with the subsequent refinements in [Arik et al., 2017]. We refer the reader to the these articles for further details. Our data consists of triplets \( \{(x_1, y_1, z_1), \ldots, (x_N, y_N, z_S)\} \) over \( N \) songs and \( S \) styles, where \( x_i \) is the 256-valued encoded waveform, \( y_i \) is the 128-valued binary encoded MIDI and \( z_i \in \{1, 2, \ldots, S\} \) is the one-hot encoded style label. Each audio sample \( x_i \) is conditioned on the audio samples at all previous timesteps \( x_{<t} = \{x_{t-1}, x_{t-2}, \ldots, x_1\} \), all previous binary MIDI samples and the global conditioning vector. The joint probability of a waveform \( x = \{x_1, \ldots, x_T\} \) is factorized as follows:

\[
p(x|y, z) = \prod_{t=1}^{T} p(x|\{x_{<t}, y_{<t}, z\}).
\]  

The hidden state before the residual connection in dilation block \( \ell \) is

\[
h^\ell = \sigma(W_f^\ell \ast x^{\ell-1} + V_f^\ell \ast y^{\ell-1} + U_f^\ell \ast z)\circ\tau(W_g^\ell \ast x^{\ell-1} + V_g^\ell \ast y^{\ell-1} + U_g^\ell \ast z),
\]

while the output of every dilation block, after the residual connection is

\[
x^\ell = x^{\ell-1} + W_c^\ell \cdot h^\ell,
\]

where \( \tau \) and \( \sigma \) are respectively the tanh and sigmoid activation functions, \( \ell \) is the layer index, \( f \) indicates the filter weights, \( g \) the gate weights, \( r \) the residual weights and \( W, V \) and \( U \) are the learned parameters for the main, local conditioning and global conditioning signals respectively. The \( f \) and \( g \) convolutions are computed in parallel as a single operation [Arik et al., 2017]. All convolutions have a filter width of \( F \). The convolutions with \( W_c^\ell \) and \( V_c^\ell \) are dilated.

3 End-to-end Synthesizer Learning

[Van Den Oord et al., 2016] and [Arik et al., 2017] have shown that generative convolutional networks effectively learn the human voice from raw audio. This has advanced the state of the art in text to speech (TTS). We further explore the possibilities of these architectures by benchmarking them in the
We hypothesize that the skip connections are superfluous and perform a set of initial experiments to gain more insight into the learned representations. We use Gram matrices to extract activation statistics since these have been previously used for artistic style transfer [Gatys et al., 2015]. After training, the validation data is fed through five locally conditioned networks, each trained with a distinct timbre.

The data has identical melodic content but has different harmonic content (i.e. same song, different instruments). The Gram matrices are extracted from the outputs of each dilated block (Equation 3) for each network - timbre. These are flattened and projected onto 2D, simultaneously over all layers and styles via T-SNE [Maaten and Hinton, 2008]. The results presented in Figure 1 show that the activations separate further as the layer index increases. A broad interpretation is that the initial layers extract low-level generic audio features, these being common to all waveforms.

However, since this is a controlled experiment, we can be more specific. The timbre of a musical instrument is characterized by a specific set of resonating frequencies on top of the fundamental frequency (pure sine wave). Typically one identifies individual notes based on their fundamental, or lowest prominent frequency. These depend on the physics of the musical instrument and effects generated, for example, by the environment. Since the sequence of notes is identical and the timbres differ, we conjectured that Figure 1 could imply a frequency-layer correspondence. While the latter statement might be loose, Gram matrices extracted from the lower layers are nevertheless much closer due the increased similarity with the fundamental frequency.

### 3.3 SynthNet Architecture

Figure 1 provides indicative results from many experiments. We hypothesize that the skip connections are superfluous and the conditioning of the first input layer should suffice to drive the melodic component. We also hypothesize that the first audio input layer learns an embedding corresponding to the fundamental pitches. Then, we aim to learn mapping from the symbolic representation (binary midi code) to the pitch embeddings (Equation 8). Therefore, in SynthNet there are no parameters learned in each dilation block for local conditioning and the hidden activation with global conditioning (omitted in Figure 2) becomes

\[ h^\ell = \tau \left( W^\ell_f \cdot x^{\ell-1} + U^\ell_f \cdot z \right) \odot \sigma \left( W^\ell_g \cdot x^{\ell-1} + U^\ell_g \cdot z \right). \]  

\[ \hat{y} = V^{\text{out}} \cdot \tau(V^h \cdot y^h). \]

As it can be seen in Figure 2 there are no skip connections and Equation 4 no longer applies. We also found that using SeLU activations [Klambauer et al., 2017] in the last layers improves generation stability and quality. Other normalization strategies could have been used, we found SeLU to work well. In addition, we further increase sparsity by changing the dilated convolution in Equation 6 with a dilated depthwise separable convolution. Separable convolutions perform a channel-wise spatial convolution that is followed by a 1 × 1 convolution.

In our case each input channel is convolved with its own set of filters. Depthwise separable convolutions have been successfully used in mobile and embedded applications [Howard et al., 2017] and in the Xception architecture [Chollet, 2017]. As we show in Table 4 and Table 5, the parsimonious approach works very well since it reduces the complexity of the architecture and speeds up training.

SynthNet is trained with an additional auxiliary autoencoder latent code:

\[ h^\ell = \tau(V^0 \cdot y) \]  
\[ x^0 = \tau(W^0 \cdot x) + \tau(V^h \cdot y^h) \]  
\[ \hat{y} = V^{\text{out}} \cdot \tau(V^h \cdot y^h). \]
We compare exact replicas of the architectures described in [Van Den Oord et al., 2016; Arik et al., 2017] with our proposed architecture SynthNet. All are implemented in PyTorch, available at https://github.com/florinsch/synthnet. We train the networks to learn the instrument timbre using raw audio waveforms. The networks are conditioned locally with a 128 valued vector which designates note on-off, extracted from the midi files. The latter describes the melodic content.

For the purpose of validating our hypotheses, we chose to eliminate extra sources of error by manually upsampling the midi files. That is, we simply treat the midi note on/offs as points in continuous time and look up whether the note is on at the given grid point, where the grid may be arbitrarily fine grained. For the results in Table 4, the network is also conditioned globally with a one-hot vector which designates the style (instrument) identity. Hence, multiple instrument synthesizers are learned in a single model. For the hyperparameter search experiments (Table 3) and the final MOS results (Table 5) we train one network for each style, since it is faster. We use the Adam [Kingma and Ba, 2014] optimizer with a batch size of 1, a learning rate of $10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$ with a weight decay of $10^{-5}$. We find that for most instruments 100-150 epochs is enough for generating high quality audio, however we keep training up to 200 epochs to observe any unexpected behaviour or overfitting. All training is done on Tesla P100 GPUs with 16GB of memory.

4 Experiments

4.1 Synthetic Registered Audio

We generate the dataset using the freely available Timidity++1 software synthesizer. For training we selected parts 2 to 6 from Bach’s Cello Suite No. 1 in G major (BWV 1007). We found that this was enough to learn the mapping from midi to audio and to capture the harmonic properties of the musical instruments. From this suite, the Prelude (since it is most commonly known) is not seen during training and is instead used for measuring the validation loss and for conditioning the generated audio.

After synthesizing the audio, we have approximately 12 minutes of audio for each timbre, of which 9 minutes (75%) is used for training and 3 minutes (25%) for validation. We experiment with $S = 7$ timbres, which were selected to be as different as possible. Each style corresponds to a specific preset from the ‘Fluid-R3-GM’ sound font. These are (preset number - instrument): S01 - Bright Yamaha Grand, S09 - Glockenspiel, S24 - Nylon String Guitar, S42 - Cello, S56 - Trumpet, S75 - Pan Flute and S80 - Square Lead.

For training, the single channel waveforms are sampled at 16kHz and the bit-depth is reduced to 8 bit via mu-law encoding. Before reducing the audio bit depth, the waveforms are dithered using a triangular noise distribution with limits $(-0.009, 0.009)$ and mode 0. This reduces perceptual noise but more importantly keeps the quantization noise out of the signal frequencies, which is critical for the learning process.

Without dithering there are melodic discontinuities and clipping errors in the generated waveforms. The latter errors are most likely due to notes getting mapped to the wrong set of frequencies (artifacts appear due to the quantization error). From all timbres, the added white noise due to dithering is most noticeable for Glockenspiel, Cello and Pan Flute. The midi is upsampld to 16kHz to match the audio sampling rate and each frame contains a 128 valued vector which designates note on-off times for each note (piano roll).

4.2 Measuring Audio Generation Quality

Quantifying the performance of generative models is a non-trivial task. Similarly to [Van Den Oord et al., 2016; Arik et al., 2017] we have found that once the training and validation losses go beyond a certain lower threshold, the quality improves. However, the losses are only informative towards convergence and overfitting (Figure 4) - they are not sensitive enough to accurately quantify the quality of the generated audio. This is critical for ablation studies where precision is important.

1http://timidity.sourceforge.net/
[Theis et al., 2015] argue that generative models should be evaluated directly. Then, the first option is the mean opinion score (MOS) via direct listening tests. This can be impractical, slowing down the hyperparameter selection procedure. However, we provide MOS ratings for the best found models in Table 5. Instead, we propose to measure the root mean squared error (RMSE) of the Constant-Q Transform (RMSE-CQT) between the generated audio and the ground truth waveform (Figure 4, lower plots). Similarly to the Fourier transform, the CQT [Brown, 1991] is built on a bank of filters, however unlike the former, it has geometrically spaced center frequencies that correspond to musical notes.

Although we evaluated other metrics, only the RMSE-CQT was correlated with the quality of the generated audio. This subjective observation was initially made by listening to the audio samples and by comparing the plots of the audio waveforms (Figure 3). Roughly speaking, as we also show in Figure 3 (top captions) and Figure 4 (lower plots), we find that a RMSE-CQT value below 10 corresponds to a generated sample of reasonable quality. The RMSE-CQT also penalizes temporal delays (Figure 3 - right) and is also correlated with the MOS (Table 3 and Table 5).

We generate every 20 epochs during training and compute the RMSE-CQT to check generation quality. Indeed, Figure 4 shows that the generated signals match the target audio better as the training progresses, while the losses flatten. However, occasionally the generated signals are shifted or the melody is slightly inaccurate – that is, the wrong note is played (Figure 3 - right). This is not necessarily only a function of the network weight state since the generation process is stochastic. We set a fixed random seed at generation time, thus we only observe changes in the generated signal due to weight changes. For one model, the RMSE-CQT is averaged over all epochs.

### 4.3 Hyperparameter Selection

There are many possible configurations for the filter width $F$, the number of blocks $B$, and the maximum dilation rate $R$. The dilation rates per each block are: $\{2^0, 2^1, \ldots, 2^{R-1}\}$. In addition, there is the choice of the number of residual and skip channels. For speech [Arik et al., 2017] use 64 residual channels and 256 skip channels, [Engel et al., 2017] use 512 residual channels and 256 skip channels, while [Mor et al., 2018] use 512 for both. These methods have a receptive field $\Delta < 1$. Since the latter two works are also focused on music, we use 512 channels for both the residual and skip convolutions and set the final two convolutions to 512 and 384 channels respectively. We hypothesize that it is better to maximize the receptive field $\Delta$ while minimizing the number of layers. Thus, in Table 3 the receptive field is set to 1 second and the other parameters are varied according to Table 2. We observed that the networks train faster and the quality is better when the length of the audio slice is maximized within GPU memory constraints. It can be seen in Table 3 that SynthNet outperforms both baselines. Some instruments are more difficult to learn than others (also see Figure 4). This is also observable from listening to and visualizing the generated data, available here: http://bit.ly/synthnet_table3.

The lowest errors for the first four instruments are observed for SynthNet L48 while the last three are lowest for SynthNet L24 (Table 3 italic text). This could be due to either an increased granularity over the frequency spectrum, provided by the extra layers of the L48 model or a better overlap. The best overall configuration is SynthNet L24. For DeepVoice and WaveNet, both L24 and L48 have more parameters (Table 3, second last row) and are slower to train, even though all setups have the same number of hidden channels (512) over both baseline architectures. This is because of the skip connections and associated convolutions.

**Global Conditioning.** We benchmark only DeepVoice L26 against SynthNet L24, with the difference that now one model is trained to learn all 7 timbres simultaneously (Table 4). This slows down training considerably. The errors are higher as opposed to learning one model per instrument, however SynthNet has the lowest error. We believe that increasing the number of residual channels would have resulted in lower errors.

### Table 2: Three setups for filter, dilation and number of blocks resulting in a similar receptive field.

<table>
<thead>
<tr>
<th>Filter width $F$</th>
<th>Num blocks $B$</th>
<th>Max dilation $R$</th>
<th>Receptive field $\Delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>L24</td>
<td>3</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>L26</td>
<td>2</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>L48</td>
<td>2</td>
<td>4</td>
<td>12</td>
</tr>
</tbody>
</table>

Figure 3: Left: 1 second of ground truth audio of Bach’s BWV1007 Prelude, played with FluidSynth preset 56 Trumpet. Center: SynthNet high quality generated. Right: DeepVoice low quality generated showing delay. We encourage the readers to listen to the samples here: http://bit.ly/synthnet_appendix_a
Table 3: Mean RMSE-CQT and 95% confidence intervals (CIs). Two baselines are benchmarked for three sets of model hyperparameter settings (Table 2), all other parameters identical. One second of audio is generated every 20 epochs (over 200 epochs) and the error versus the target audio is measured and averaged over the epochs, per instrument. Total number of parameters and training time are also given. Audio and visuals available here: http://bit.ly/synthnet_table3

<table>
<thead>
<tr>
<th></th>
<th>WaveNet</th>
<th>DeepVoice</th>
<th>SynthNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>S01</td>
<td>16.56±4.67</td>
<td>14.83±5.39</td>
<td>18.15±2.88</td>
</tr>
<tr>
<td>S09</td>
<td>24.01±5.38</td>
<td>22.20±4.56</td>
<td>25.47±3.96</td>
</tr>
<tr>
<td>S24</td>
<td>17.68±3.05</td>
<td>18.95±4.13</td>
<td>19.30±1.26</td>
</tr>
<tr>
<td>S42</td>
<td>15.83±3.91</td>
<td>17.20±3.53</td>
<td>16.29±3.38</td>
</tr>
<tr>
<td>S56</td>
<td>18.50±2.23</td>
<td>17.25±2.98</td>
<td>22.89±1.73</td>
</tr>
<tr>
<td>S75</td>
<td>20.89±6.90</td>
<td>20.03±6.73</td>
<td>19.78±5.15</td>
</tr>
<tr>
<td>S80</td>
<td>27.73±2.29</td>
<td>26.74±3.92</td>
<td>26.96±4.71</td>
</tr>
<tr>
<td>All</td>
<td>20.02±1.79</td>
<td>19.60±1.73</td>
<td>21.35±1.48</td>
</tr>
</tbody>
</table>

Params 8.23e+7 6.18e+7 1.14e+8 8.90e+7 6.89e+7 1.27e+8 7.35e+6 7.80e+6 1.36e+7
Time 4d2h 4d2h 8+ days 4d10h 3d9h 8+ days 16 hours 16 hours 1d5h

Figure 4: Seven networks are trained, each with a different timbre. Top, losses: training (left) validation (right). Bottom, RMSE-CQT: DeepVoice L26 (left [Table 3, column 6]) and SynthNet L24 (right [Table 3, column 8]). DeepVoice overfits for Glockenspiel (top right, dotted line). Convergence rate is measured via the RMSE-CQT, not the losses. The capacity of DeepVoice is larger, so the losses are steeper.

4.4 MOS Listening Tests

Given the results in Table 3, we benchmark the best three obtained setups: SynthNet L24, DeepVoice L26 and WaveNet L26. For these experiments, we generate samples based on three musical pieces using the converged models, from all instruments. We generate 5 seconds of audio from Bach’s Cello suites not seen during training, namely Part 1 of Suite
We tested the hypothesis that the first causal layer learns fundamental frequencies. We validated this empirically, arriving at the SynthNet architecture which converges faster and produces higher quality audio. To the best of our knowledge, this is the first time an autoregressive generative model is used to learn synthesizers. Our method is able to simultaneously learn the characteristic harmonics of a musical instrument (timbre) and a joint embedding between notes and the corresponding fundamental frequencies. While we focus on music, we believe that SynthNet can also be successfully used for other time series problems, and we plan to investigate this in future work.

4.5 Synthesizing Real Instruments

We used labeled audio recordings of real audio performances from the MusicNet dataset [Thickstun et al., 2016]. We selected Bach’s 3rd and 4th cello suites for training, leaving out the first part of Bach’s 3rd cello suite for testing (i.e. generation) - it was also used during the synthetic audio MOS tests. We used 44kHz audio for training the L24 SynthNet setup. We encourage the reader to listen to the audio generated based on real cello recordings here: http://bit.ly/synthnet_real_cello. The latter demonstrates that timbre can be reproduced even from real recordings and, importantly, that the melody remains accurate. While improvements can be made by pre- and post-processing the audio (e.g. with compression) and further tuning the hyperparameters (e.g. increasing the network capacity to account for increased complexity); our objective is to demonstrate the effectiveness of the model itself.

5 Discussion

In the current work, we gave some insights into the learned representations of autoregressive generative convolutional models. We tested the hypothesis that the first causal layer learns fundamental frequencies. We validated this empirically, arriving at the SynthNet architecture which converges faster and produces higher quality audio. To the best of our knowledge, this is the first time an autoregressive generative model is used to learn synthesizers.

Table 4: RMSE-CQT Mean and 95% CIs. All networks learn 7 timbres simultaneously - global conditioning, in addition to local conditioning.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Piano</th>
<th>Glockenspiel</th>
<th>Guitar</th>
<th>Cello</th>
<th>Trumpet</th>
<th>Flute</th>
<th>Square</th>
<th>All</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SynthNet L24</td>
<td>9.37 ± 0.71</td>
<td>15.12 ± 3.39</td>
<td>11.88 ± 0.95</td>
<td>11.66 ± 2.35</td>
<td>13.98 ± 1.70</td>
<td>12.01 ± 1.36</td>
<td>10.90 ± 1.17</td>
<td>12.13 ± 0.74</td>
<td>5d23h</td>
</tr>
<tr>
<td>WaveNet L26</td>
<td>2.22 ± 0.25</td>
<td>2.48 ± 0.23</td>
<td>2.18 ± 0.25</td>
<td>2.37 ± 0.28</td>
<td>2.18 ± 0.29</td>
<td>2.37 ± 0.22</td>
<td>2.30 ± 0.09</td>
<td>2.30 ± 0.10</td>
<td></td>
</tr>
<tr>
<td>DeepVoice L26</td>
<td>2.55 ± 0.32</td>
<td>1.85 ± 0.23</td>
<td>2.30 ± 0.39</td>
<td>2.62 ± 0.27</td>
<td>2.28 ± 0.32</td>
<td>2.20 ± 0.25</td>
<td>1.87 ± 0.03</td>
<td>2.24 ± 0.11</td>
<td></td>
</tr>
<tr>
<td>DeepVoice L26</td>
<td>4.75 ± 0.14</td>
<td>4.45 ± 0.17</td>
<td>4.30 ± 0.19</td>
<td>4.50 ± 0.15</td>
<td>4.25 ± 0.18</td>
<td>4.15 ± 0.21</td>
<td>4.10 ± 0.16</td>
<td>4.36 ± 0.07</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Listening MOS and 95% CIs. 5 seconds of audio are generated from 3 musical pieces (Bach’s BWV 1007, 1008 and 1009), over 7 instruments for the best found models. Subjects are asked to listen to the ground truth reference, then rate samples from all 3 algorithms - it was also used during the synthetic audio MOS tests. The latter demonstrates that timbre can be reproduced even from real recordings and, importantly, that the melody remains accurate. While improvements can be made by pre- and post-processing the audio (e.g. with compression) and further tuning the hyperparameters (e.g. increasing the network capacity to account for increased complexity); our objective is to demonstrate the effectiveness of the model itself.

Acknowledgments

This research was supported by ARC Discovery Grant DP170102472, Data61 CSIRO and sponsored by NVIDIA.

References


