Transposition of Simple Waveforms from Raw Audio with Deep Learning

Patrick J. Donnelly(✉) and Parker Carlson

Oregon State University, Corvallis, OR 97331, USA
{donnellp,carlspar}@oregonstate.edu

Abstract. A system that is able to automatically transpose an audio recording would have many potential applications, from music production to hearing aid design. We present a deep learning approach to transpose an audio recording directly from the raw time domain signal. We train recurrent neural networks with raw audio samples of simple waveforms (sine, square, triangle, sawtooth) covering the linear range of possible frequencies. We examine our generated transpositions for each musical semitone step size up to the octave and compare our results against two popular pitch shifting algorithms. Although our approach is able to accurately transpose the frequencies in a signal, these signals suffer from a significant amount of added noise. This work represents exploratory steps towards the development of a general deep transposition model able to quickly transpose to any desired spectral mapping.

Keywords: Deep learning · FFT · Music · Timbre · Transposition

1 Introduction

Transposing a recorded audio file to a different set of frequencies is a difficult research task. Automatic transposition requires scaling frequency by a fixed multiplicative constant while preserving the time scale. A system that is able to transcribe audio in near real-time would have a variety potential uses ranging from automatic pitch correction during live vocal performances to intelligent filtering and transformations to improve hearing aid-mediated perception.

Changing the pitch is often the desired goal of audio manipulation, such as when making adjustments to the intonation of vocal lines or for the transposition of an audio recording to another musical key. In other circumstances, a change in pitch is an undesired side effect, often resulting from resampling or time-stretching manipulations. Whichever the goal, such audio manipulations must balance a trade-off between temporal and spectral fidelity.

In this paper, we investigate the automatic transposition of audio directly from raw audio signals. We present a novel approach to train deep recurrent neural networks to learn new frequency mappings in order to perform transposition directly on the time-domain signal itself. Currently, other deep learning approaches rely on Fourier analysis, feature extraction, spectral manipulation,

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
C. Johnson et al. (Eds.): EvoMUSART 2023, LNCS 13988, pp. 341–356, 2023.
https://doi.org/10.1007/978-3-031-29956-8_22

and resampling or resynthesis, which all require time to compute. To compensate, near real time systems must use small windows for spectral analysis, which reduces the fidelity of the frequency analysis by limiting the number of samples available. Our proposed approach tasks the neural network to approximate the spectral transformation, without explicit use of the Fast Fourier Transform (FFT) algorithm. To our knowledge, this is the first attempt to train deep neural networks for music transposition directly from the raw audio signal.

2 Related Work

Following the success of deep learning approaches for speech synthesis (see review [10]), these techniques were soon adapted for other audio learning tasks. Researchers have applied deep learning in a number of musical tasks including signal processing [20], content description [19], information retrieval [3], recommendation systems [23], and composition [8].

Although many recent approaches and models for audio deep learning have focused primarily upon speech synthesis, some models also support music synthesis. One such example is Wavenet [18], a deep generative model for generating speech and music that uses a fully connected convolutional network with dilation factors to generate output audio one sample at a time. Another recent example is Differential Digital Signal Processing [6], which provides a differentiable framework for traditional signal-processing techniques to enable audio generation with smaller models. Despite these recent advances in deep learning of music, many challenges and unsolved research tasks persist [2].

In the remainder of this section, we briefly review techniques, algorithms, and studies related to automatic music transposition. First, we review signal processing approaches for time-scale modification of pitch. Next, we discuss the fast Fourier transform and the phase vocoder. Lastly, we review the recent approaches for automatic pitch correction and intonation adjustment.

2.1 Time-Scale Approaches

Before the availability of deep-learning approaches, researchers employed signal processing approaches to attempt to manipulate audio frequency. When an audio signal is resampled and played back at it original sampling rate, both the pitch and duration of the signal will change. To decouple this dependency, researchers developed numerous resampling and time-scale modification approaches. These algorithms facilitate applying changes of pitch with retaining the original duration, or conversely, they allow changes of duration without changing the pitch.

Noteworthy examples of such time-domain algorithms include the Synchronised Overlap-Add (SOLA) [22], Time-Domain Pitch Synchronous Overlap-Add (TD-PSOLA) [16], Waveform Similarity Overlap-Add (WSOLA) [24], and Adaptive Overlap-add (AOLA) [11] algorithms. More recent approaches, such as Canonical Time Warping (CTW) [26] have also been employed to aid in aligning vocal corrections with the accompaniment mixture [13].

2.2 Phase Vocoder

The fast Fourier transform (FFT) is an efficient algorithm to compute the discrete Fourier transform (DFT) in $\mathcal{O}(n \log n)$ multiplications. In the context of deep-learning, Bengio and LeCun describe the FFT as a deep representation of the Fourier transform [1]. And researchers have found that performing convolutions as products in the Fourier domain can accelerate training and inference of convolutional neural networks [15]. FFT-based deep representations have also been shown to help reduce complexity while performing inference on embedded platforms [12]. However, one recent study failed to learn the FFT representation within $\mathcal{O}(n \log n)$ scaling using a deep linear neural network [17].

A phase vocoder (see tutorial [5]) is technique for analyzing and synthesising an audio signal. Originally developed for synthesizing the human voice, the phase vocoder has been a popular technique in electroacoustic music since the 1970's. A phase vocoder uses the FFT to decompose a signal into narrow frequency bands, which are used to tune a series of bandpass filters. A carrier signal, such as a noise source, is then sent through these filters to synthesis the original sound.

In addition to its other creative uses, a phase vocoder can be used to time stretch a signal without changing the pitch and to manipulate pitch without changing the signal's duration. One study proposed a modulation vocoder for selective frequency pitch transposition [4]. The authors compared their approach against a commercial algorithm in a small subject study. The participants judged the vocoded transposition satisfactory with respect to pitch and harmony but preferred the propriety approach for preservation of timbre. Another study explored adaptive pitch shifting with a non-linear phase vocoder to perform global and local intonation adjustments in recordings of a cappella singing [21].

2.3 Pitch Correction

Pitch correction is an important tool in audio engineering and music production. This is the task of altering the pitches in an audio stream, usually with the intent to tune the notes to a specific track or reference frequency. Autotune[1], VariAudio[2], and Melodyne[3] are commercial tools that enable manipulation of pitch, primarily intended for corrections of intonation in vocal tracks.

Current approaches to pitch correction typically require a pipeline that includes a reference track as input, a DFT, spectral feature or filter manipulation, and post-processing to reconstruct the adjusted audio as the output. These approaches are usually limited by how much transposition they are able to perform. Many auto-tuner systems work best for transposition of small musical intervals but may sound distorted when transposing larger intervals, such as those greater than an octave. Furthermore, these approaches are usually not well suited to audio streams with accompaniment or other background vocals.

[1] http://www.antarestech.com/products/.
[2] https://www.steinberg.net/cubase/new-features/.
[3] https://www.celemony.com/en/melodyne/what-is-melodyne.

Other approaches have attempted data-driven approaches for automatic pitch correction. In one such example, the authors trained deep neural networks using Constant Q-Transform features on a dataset of short excerpts of amateur karaoke performances [25]. In subjective listening tests, study participants preferred the automatically auto-tuned examples compared to the original 49% of time.

3 Dataset

We begin our exploration of audio transposition directly from raw audio by investigating the four basic waveform shapes used in music synthesis. We consider the pure sine tone, as well as the non-sinusoidal square, triangle, and sawtooth waveforms. In Fig. 1, we illustrate the waveform and spectra for each waveform. We synthetically-generate the waveforms for the dataset of our experiment, rather than using an existing dataset of sounds or music so that we can precisely control the frequencies of the sounds used in training and testing.

In this section we provide an overview of our process to synthetically generate our dataset, our mapping between our input and our desired output based on transposition step size, and our partition between training and test examples.

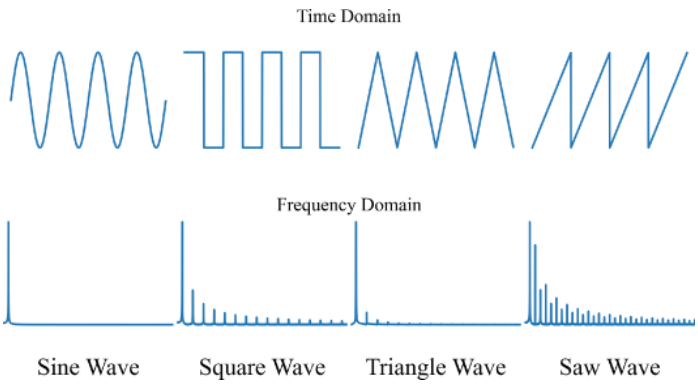


Fig. 1. Time domain (top) and frequency domain (bottom) for each waveform.

3.1 Waveform Generation

For each waveform shape, we generate one second examples with fundamental frequencies linearly spaced along the frequency range using a spacing of 0.5 Hz. We generate all signals with amplitudes of 0.1 to avoid potential clipping in our output. These waveforms are saved as a single channel WAV file with a sampling rate of 16 kHz. Given our sampling rate and the Nyquist limit, we explore frequencies up to 8 kHz.

3.2 Input and Output

In order to evaluate different transposition step sizes, ranging from one semitone (minor second) to 12 semitones (octave), we generate 12 separate datasets each with a different mapping from input frequencies to corresponding output frequencies. Because we evaluate the octave as our largest possible transposition, we consider frequencies in the range [20, 4000] for inclusion as input examples for training our neural network.

For each waveform, we consider examples in our input range and identify the corresponding output resulting in the transposition from semitone s in the range [1, 12]. Using intervals from equal temperament, we calculate $d = 2^{\frac{s}{12}}$ as a multiplier to find the frequency of a tone raised by s semitones. For each semitone up to an octave, we map each input frequency in range [20, 4000] Hz to the corresponding transposed frequency in range $[20 \times d, 4000 \times d]$ Hz.

3.3 Train and Test Split

In this work we seek to train a deep neural network to transpose audio directly from raw audio. In other words, we desire the trained network to approximate the Fourier transform over the range of possible input frequencies. For this reason, our approaches does not lend itself to a traditional partition of examples into train examples and held-out test examples. Instead we seek to learn to approximate a general transposition function for any input frequency, using a model trained only on a discrete and finite sampling of frequencies in the input range.

Specifically, we train the neural network with a set of fundamental frequencies covering the entire range linearly spaced by 1.0 Hz. We then test our models using frequencies spaced between the training examples. This ensures that test examples are evenly spread throughout the training domain. For example, examples with fundamental frequency F_0 of 440, 441, 442, etc. are added to the training partition, while examples with F_0 of 440.5, 441.5, and 442.5, etc. are held out for testing. This spacing of 0.5 was chosen to ensure an equal distance in frequency between any test example and its neighboring train examples.

In total, we have 12 datasets, one for each semitone [1, 12], each including all four wavesforms: sine, square, triangle, and sawtooth. Each dataset contains 15,920 examples for training and 15,920 examples for testing.

4 Experimental Design

In this section we describe our choices of deep learning architecture, model parameters, and training paradigm. We also discuss the two pitch shifting algorithms against which we compare our results as well as the algorithm we use to measure the noise of our generated transpositions.

4.1 Model

Based on preliminary experiments, we select a recurrent network topology consisting of three stacked layers of Bidirectional LSTMs (BiLSTMs) each with 1024 hidden units, with a recurrent dropout of 0.4, and batch normalization performed between the BiLSTM layers. We train using `RMSProp` [27] with a learning rate of 0.001 and a batch size of 64. We initialize our model weights and parameters based on a random seed, which we hold constant across our experiments.

When training the model, we split each input audio stream into a sequence of chunks of 1024 samples. This ensures that each input chunk contains a full cycle 20 Hz, the lowest frequency in our input set. The last input chunk of the audio stream is zero-padded to a multiple of 1024. For each experiment, we train our models for 250 epochs. We use a fixed stopping criteria to better support comparison between the results of our different experiments.

4.2 Evaluation

We compare our results against two open-source pitch shifting algorithms. First, we consider the pitch-shifting algorithm in Librosa v.0.8.1[4], a Python package for music and audio analysis. This naïve implementation stretches the signal in time, then resamples to ensure the output's duration is the same of the original. Second, we consider the pitch shifting algorithm implemented by Rubber Band[5]. This algorithm is known for its robust qualities, such as handling audio transients and its ability to pitch shift up to three octaves.

We also calculate the signal-to-noise ratio (SNR) to evaluate the noisiness of our generated output. We use the implementation in Matlab[6] to measure the noise in decibels relative to the carrier (dBc). This implementation expects a sinusoidal input and uses a modified periodogram with a Kaiser window of $\beta = 38$ and explicitly excludes the first six harmonics as not noise.

5 Results

For each of the 12 semitone steps, we train a separate recurrent neural network to transpose audio. The models are trained on the raw audio signal with examples from all four waveforms, sampled in the range [20, 4000] Hz. These models generate a transposed signal that approximates the desired transposition step size. The output of the model does not require any post-processing and can be saved directly as a WAV file.

We compare our transpositions against those generated by the Librosa and Rubberband baseline algorithms, described in Sect. 4.2. We examine the quality of our generated output in three ways. First, we examine how well our generated signal matches the expected signal in the time domain. Second, we examine the accuracy of generated signal in the frequency domain. Lastly, we measure the amount of noise generated in our signal, compared to the expected signal.

[4] https://librosa.org/doc/main/generated/librosa.effects.pitch_shift.html.
[5] https://breakfastquay.com/rubberband/.
[6] https://www.mathworks.com/help/signal/ref/snr.html.

5.1 Error in the Time Domain

We begin by comparing our generated signal against the expected transposed audio. We calculate the mean squared error (MSE) between the two signals in the time-domain. This approach, used by [9], measures how well our system is able to precisely match the expected output signal, sample by sample. We compare our result against the Librosa and Rubberband algorithms in Table 1.

Across the different transposition sizes, our approach achieves a slightly lower MSE error than the other two algorithms. However, we do not find this difference statistically significant. Moreover, from the manual examination of individual examples, we observe that the lower performance of Librosa and Rubberband relative to our approach is likely due to small differences in phase. In other words, our approach was better able to match the exact phase of the expected transposed signal. The other approaches, which rely on time-stretching and resampling, were slightly out of phase with the expected output. This contributed to a higher overall error using this comparison, and does not necessarily reflect the quality or accuracy of the transposed frequencies.

Table 1. Mean Squared Error (MSE) in the time domain for each model across the different transposition sizes.

Model	Sine	Square	Triangle	Saw	Overall
Our Approach	0.0089	0.0113	0.0071	0.0070	**0.0086**
Librosa	0.0094	0.0129	0.0078	0.0077	**0.0094**
Rubberband	0.0100	0.0138	0.0081	0.0080	**0.0100**

In left column of Fig. 2 we breakdown this performance by transposition step-size for each waveform. Overall we found similar trends between our algorithm and the two baseline algorithms for many of the transpositions. Unexpectedly, we found that all approaches had more difficulty with some transpositions for certain waveforms. For example, all three algorithms demonstrated increased error for step size $s = 3$ for sine waves, $s = 6$ for triangle waves, and $s = 7$ and $s = 9$ for sawtooth waves. Although we expected a more even performance across the different transposition sizes, we instead found a variance between step sizes and that this variance was fairly consistent regardless of the algorithm.

To facilitate direct comparison between our different experiments, we used a consistent model architecture and a fixed number of training epochs, as described in Sect. 4.1. To further reduce error, we might benefit from exploring different network topologies or varying model training time for each transposition size and waveform. However, the similarities of error between the different algorithms

imply that certain transposition sizes are more difficult than others. More work is needed to understand if this is a side-effect of our choices in experiment design. As future work, we will investigate the effects of the example length, sample rate, and the frequency resolution of training set.

5.2 Error in the Frequency Domain

We also compare the frequencies in our generated signal against those in the expected transposed audio. We calculated the MSE between the two signals in the frequency-domain, after performing a FFT over each one second excerpt. This measures how well our system was able to precisely match the frequency content of the expected transposition. As before, we compare our result against the Librosa and Rubberband algorithms in Table 2.

Table 2. Mean Squared Error (MSE) in the frequency domain of each model across the different transposition sizes.

Model	Sine	Square	Triangle	Saw	Overall
Our Approach	0.00000209	0.00000265	0.00000168	0.00000163	**0.00000201**
Librosa	0.00000163	0.00000188	0.00000209	0.00000150	**0.00000177**
Rubberband	0.00000247	0.00000338	0.00000199	0.00000195	**0.00000245**

In the right column of Fig. 2 we show the error in frequency domain by transposition step-size and waveform type. Overall we found similar performance between our approach and the two baseline algorithms. Just as we observed with MSE of the time-domain signal, we find that our approach often had increased error in frequency-domain for certain step sizes and that those steps were similarly difficult for our baseline algorithms.

5.3 Signal to Noise

When listening to our generated transpositions, it is clear that our approach generates significantly more noise than the two baseline algorithms. To assess the noisiness of our generated signal, we calculate the signal to noise ratio (SNR), using the approach described in Sect. 4.2. We compare the average noise of our transpositions against the baseline algorithms in Table 3. Here we confirm that our approach generates significantly more noise than the other approaches.

For all waveform types we narrowly achieve a positive SNR dBc value, indicating there is more signal than noise in our generated transpositions. We also observe that we performed marginally better with the sine and triangle waveforms compared to the square or sawtooth waveforms. This implies, unsurprisingly, that our approach works better with simpler timbres and struggles more with complex timbres with a high number of frequency components. The baseline

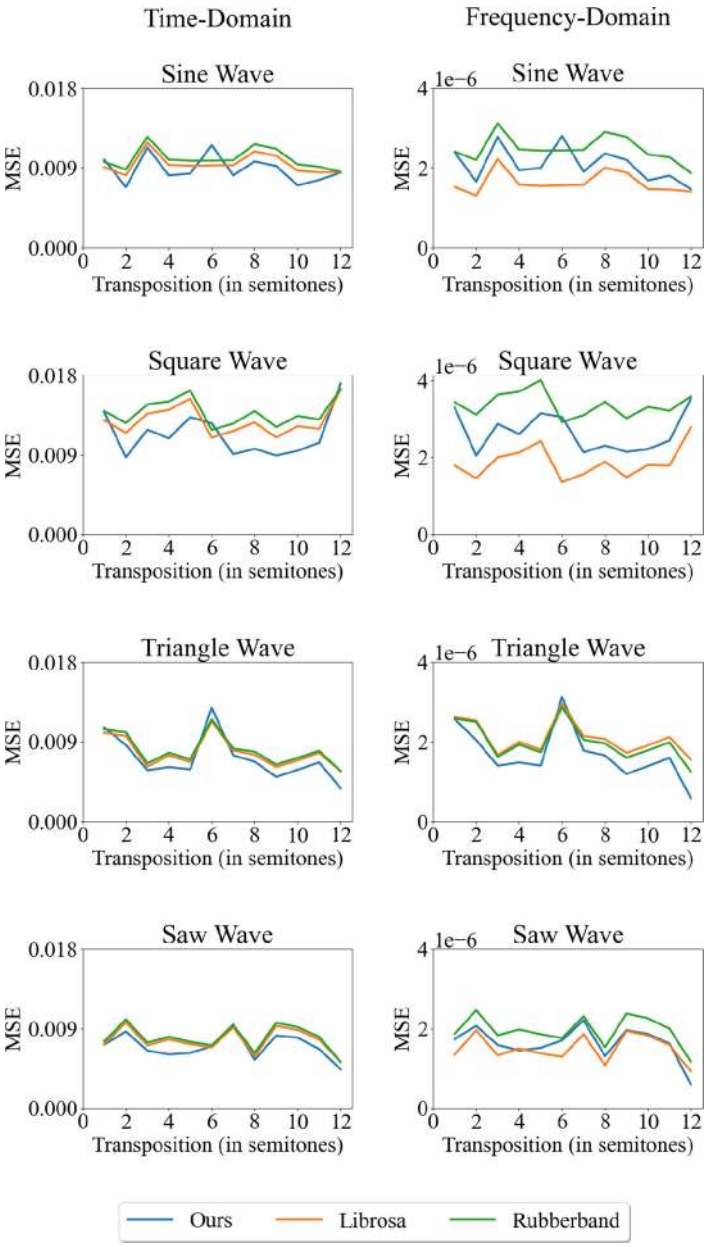


Fig. 2. Error by transposition size in semitones and for each waveform type. The left column shows MSE of the time domain signal and the right column shows MSE of the signal in the frequency domain.

algorithms also demonstrate a similar trend of increased noise for the square and sawtooth waveforms. However, some of the difference here can be attributed to our algorithm for calculating the SNR (see Sect. 4.2), which considers anything above the sixth harmonic partial to be noise. In all cases, the baseline approaches generated significantly less noise than our approach.

Table 3. Average Signal to Noise Ratio (SNR) of each model in dBc over the different transposition step sizes. Positive values of SNR indicate there is more signal than noise.

Model	Sine	Square	Triangle	Saw	Overall
Our Approach	2.65	0.95	2.49	0.23	**1.58**
Librosa	44.55	11.53	29.23	8.65	**23.49**
Rubberband	41.05	8.93	27.41	6.31	**20.93**

To better understand the source of the noise of our generated transpositions, we manually inspected individual examples. We discuss one such example in detail here. For each model type, we provided a $F_0 = 28.5$ Hz waveform as input. These models generated output 57 Hz for the sine (SNR of 11.16 dBc), square (7.50), triangle (10.65), and sawtooth (6.86) waveforms. Time domain representations of the generated signals are shown in Fig. 3, overlaid with the expected waveform shape. We 57 Hz to better visually highlight the waveform shape over the long period of a low frequency signal. We note that this noisy trend occurs for all generated output across the frequency range.

In all cases we readily and visually observe that our approach generates a lot of noise. The shape of the signal of the learned transposition is jagged and erratic as compared to the expected waveform shape. There are many errant peaks in the time domain signal. The unrefined shape of the learned waveform is the primary reason the generated transpositions result in such a noise signal. Despite the noise, we can visually confirm that our signals do attempt to match the waveform shape. Furthermore, there are clear differences between the shapes of our generated waveforms. For this and many examples, our model had the most difficulty with the square wave, given the richness of its harmonic series.

It is possible that with deeper model topologies or additional training time, our signals would become smoother and thus reduce some of the noise. However, we also compared the transpositions of the baseline algorithms for these tasks. In Fig. 4 we illustrate that the Rubberband and Librosa also do not precisely match the expected waveshape for complex timbres of the square and sawtooth waveform, contributing to some noise in their generated transpositions.

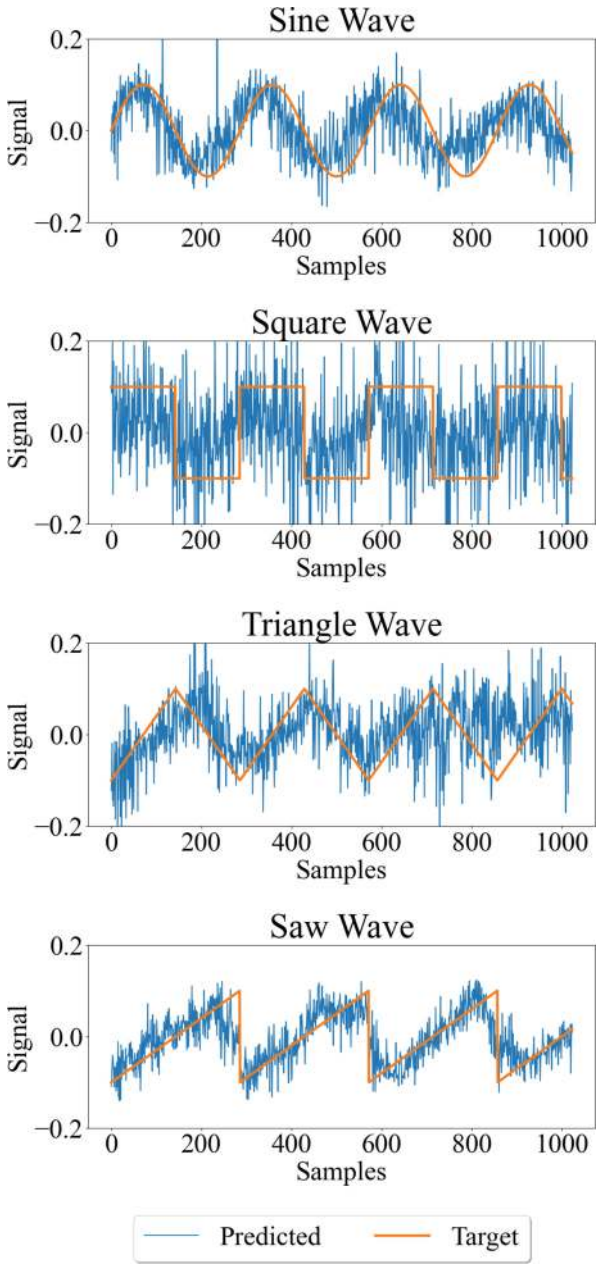


Fig. 3. Example of one octave ($s = 12$) transposition 57 Hz, comparing the generated signal (blue) to the true signal (orange). (Color figure online)

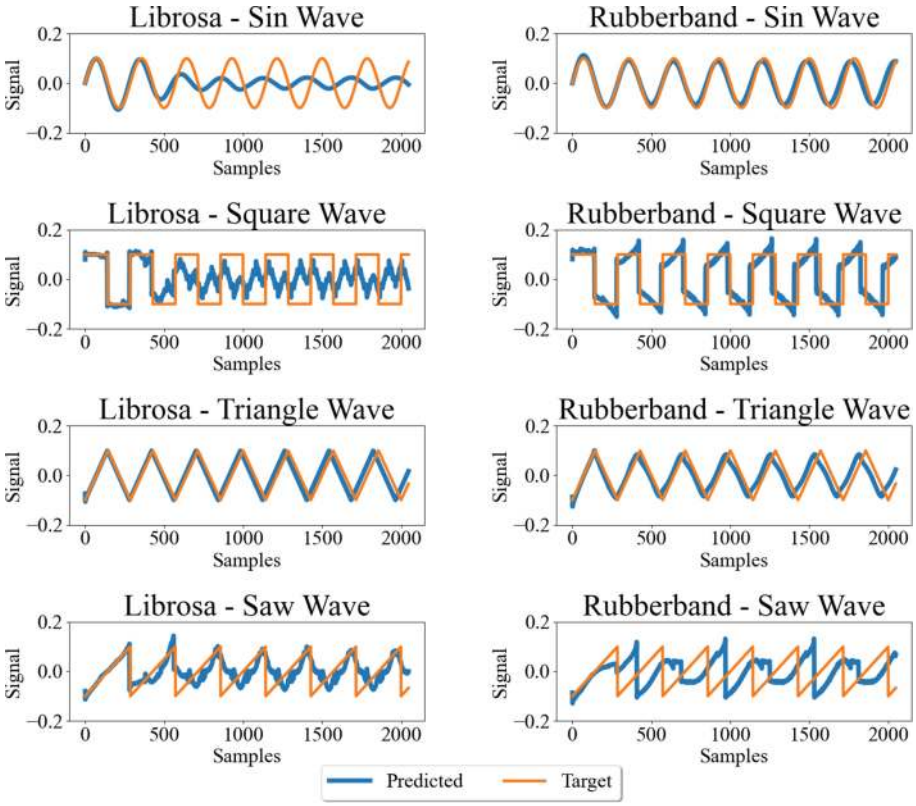


Fig. 4. In some cases, the Librosa (left) and Rubberband (right) pitch shifting methods also struggle to capture complex timbre accurately for 57 Hz example. The target is shown in orange and the pitch-shifted signal is shown in blue. (Color figure online)

6 Discussion

In this work we explore automatic transposition directly from raw audio signals. This novel approach uses deep-learning to approximate Fourier processes to transpose audio directly without pre-processing, filtering, explicit use of the fast Fourier transform, feature extraction, or resampling. Although our generated transpositions are quite noisy, they do accurately transpose the frequency content by the intended number of semitones. Moreover, our approach readily transposes complex timbres to map each harmonic partial to its appropriate transposed frequency. These results demonstrate the feasibility of an approach to transpose audio directly from the raw time domain signal.

6.1 Limitations and Future Work

Despite these encouraging initial results, there remain many research questions, challenges, and opportunities to improve our transpositions. Here we discuss the limitations of our approach in the context of our plans for future work.

One of the primary limitations of our approach is the amount noise found in the output signal, as compared to other approaches. Much of this noise results from the overall jagged shape of our generated signal, which are imperfect approximations of the true waveforms. To improve our approximation, we will explore other network architectures, including deeper recurrent networks, convolutional networks, and differential DSP models [6]. Furthermore, we will explore dilated convolution, as used by Wavenet [18], to improve the shape of our generated output. Additionally, we will investigate post-processing approaches to reduce noise, such as the use of machine learning models to filter noise as well as signal processing techniques to smooth our generated waveform. The noise in our signal appears to be Gaussian distributed and would be a good candidate for existing denoising algorithms. Lastly, we will investigate how the signal to noise ratio is affected by the transposition size.

In this initial exploration, we limited ourselves to four basic waveform shapes. In future work, we will investigate this approach on more complex examples, including both speech and polyphonic music. We will consider large established datasets like Nsynth [7], which contains MIDI pitch references from which we can estimate fundamental frequencies [14]. Towards our goal of creating a generalizable transposition model, we will also investigate transposing real world datasets of sounds and music. For these, we will not have an example of the expected transposition and thus must consider other evaluation strategies.

A persistent challenge in generative audio tasks using deep learning is measuring the efficacy of the approach and the quality of the generated audio. In the task of vocal correction, one study found a lack of generalizability of existing metrics used between studies [14]. In this study, we synthetically generated a dataset of audio signals such that we would have the input and transposed output signals. This enabled us to directly compare the time-domain signal of our true and generated outputs using mean-squared error. As future work, we will consider other evaluation approaches and metrics to enable examination of the efficacy of our system in which the correct transposed audio is not available.

First, we will evaluate the correctness of the frequencies of the transposed output. In the present study, we were unable to achieve sufficient resolution in the spectral domain given our design choice of one-second examples and a 16 kHz sampling rate. As future work, we will train our models with longer 10 s audio examples and with a higher sampling rate to ensure we have the fidelity to identify the fundamental frequency with a resolution of 0.1 Hz. However, this will significantly increase model training time. Second, we will design a subjective listening test to evaluate the efficacy of our approach to transpose complex pieces of music in which we do not have ground-truth comparisons.

6.2 Novelty and Applications

There are many useful real-world applications for an intelligent system that can perform transposition in near real-time. If a robust solution was available, it could perform pitch corrections or intonation adjustments during live concert performances. Automatic music transposition would enable near real-time audio transpositions from one musical key to another. This could serve as useful tool for musicologists, composers, arrangers, and those learning new pieces of music. Additionally, such a system would be of great use to recording engineering and music producers who could quickly transpose audio samples to other keys.

Unlike other approaches our approach could readily be adapted to apply non-linear frequency transpositions. Models could be trained to map from any individual frequency to any desired frequency. This could be used to transpose between different tuning systems, such as transforming equal temperament to just temperament. Moreover, such a system enables the possibility to remap timbre directly, perhaps allowing the timbral transformation from one instrument to another, or the conversion of one spoken voice to sound like another.

We motivate our work by the desire to perform non-linear frequency transpositions to improve the intelligibility of music for those that listen through cochlear implants. If there existed a mapping between true frequencies and those pitches the patient perceives, collected through patient-specific tests by an audiologist, researchers could investigate timbral filtering of sounds to improve perception of complex sounds. For example, such a system could simplify complex timbres by attenuating certain and very specific frequencies. In such a way, researchers could begin adapting sound directly to the patient's perception, which results from the surgeon's placement of electrodes along the cochlear during implantation.

6.3 Conclusion

In this study, we demonstrate the feasibility of a deep learning approach to transpose raw audio signals without explicit spectral analysis or manipulation. By training neural networks with generated waveforms covering the range of possible frequencies, we are able to automatically transpose signals with fundamental frequencies not explicitly included in the training set. For a model trained for a specific transposition step size and waveform shape, we successful transpose the signal to the desired frequencies and timbre, while approximating the original waveform shape. Although our current approach produces considerably more noise compared to established algorithms, we identify numerous avenues to improve the audio quality of our generated transpositions. This study represents the first investigation of a general transposition engine trained on simple waveforms and able to transpose to any non-linear spectral mapping.

References

1. Bengio, Y., LeCun, Y., et al.: Scaling learning algorithms towards AI. Large-Scale Kernel Mach. **34**(5), 1–41 (2007). https://doi.org/10.7551/mitpress/7496.003.0016
2. Briot, J.-P., Pachet, F.: Deep learning for music generation: challenges and directions. Neural Comput. Appl. **32**(4), 981–993 (2018). https://doi.org/10.1007/s00521-018-3813-6
3. Choi, K., Fazekas, G., Cho, K., Sandler, M.: A tutorial on deep learning for music information retrieval. arXiv preprint arXiv:1709.04396 (2017). https://doi.org/10.48550/arXiv.1709.04396
4. Disch, S., Edler, B.: Frequency selective pitch transposition of audio signals. In: 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 29–32. IEEE (2011). https://doi.org/10.1109/ICASSP.2011.5946320
5. Dolson, M.: The phase vocoder: a tutorial. Comput. Music. J. **10**(4), 14–27 (1986). https://doi.org/10.2307/3680093
6. Engel, J., Hantrakul, L.H., Gu, C., Roberts, A.: DDSP: differentiable digital signal processing. In: International Conference on Learning Representations (2020). https://openreview.net/forum?id=B1x1ma4tDr
7. Engel, J., et al.: Neural audio synthesis of musical notes with wavenet autoencoders. In: International Conference on Machine Learning, pp. 1068–1077. PMLR (2017). https://doi.org/10.48550/arXiv.1704.01279
8. Hernandez-Olivan, C., Beltran, J.R.: Music composition with deep learning: a review. arXiv preprint arXiv:2108.12290 (2021). https://doi.org/10.48550/arXiv:2108.12290
9. Jawahir, A., Haviluddin, H.: An audio encryption using transposition method. Int. J. Adv. Intell. Inform. **1**(2), 98–106 (2015). https://doi.org/10.26555/ijain.v1i2.24
10. Khalil, R.A., Jones, E., Babar, M.I., Jan, T., Zafar, M.H., Alhussain, T.: Speech emotion recognition using deep learning techniques: a review. IEEE Access **7**, 117327–117345 (2019). https://doi.org/10.3390/app9194050
11. Lawlor, B., Fagan, A.D.: A novel efficient algorithm for music transposition. Organ. Sound **4**(3), 161–167 (2000). https://doi.org/10.1017/S135577180000306X
12. Lin, S., Liu, N., Nazemi, M., Li, H., Ding, C., Wang, Y., Pedram, M.: FFT-based deep learning deployment in embedded systems. In: 2018 Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 1045–1050. IEEE (2018). https://doi.org/10.23919/DATE.2018.8342166
13. Luo, Y.J., Chen, M.T., Chi, T.S., Su, L.: Singing voice correction using canonical time warping. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 156–160. IEEE (2018). https://doi.org/10.1109/ICASSP.2018.8461280
14. Luo, Y.J., Lin, Y.J., Su, L.: Toward expressive singing voice correction: On perceptual validity of evaluation metrics for vocal melody extraction. arXiv preprint arXiv:2010.12196 (2020). https://doi.org/10.48550/arXiv.2010.12196
15. Mathieu, M., Henaff, M., LeCun, Y.: Fast training of convolutional networks through FFTs. arXiv preprint arXiv:1312.5851 (2013). https://doi.org/10.48550/arXiv.1312.5851
16. Moulines, E., Charpentier, F.: Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. Speech Commun. **9**(5–6), 453–467 (1990). https://doi.org/10.1016/0167-6393(90)90021-Z

17. Nye, M., Saxe, A.: Are efficient deep representations learnable? arXiv preprint arXiv:1807.06399 (2018). https://doi.org/10.48550/arXiv.1807.06399
18. van den Oord, A., et al.: Wavenet: a generative model for raw audio. arXiv preprint arXiv:1609.03499 (2016). https://doi.org/10.48550/arXiv.1609.03499
19. Peeters, G., Richard, G.: Deep learning for audio and music. In: Benois-Pineau, J., Zemmari, A. (eds.) Multi-faceted Deep Learning, pp. 231–266. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-74478-6_10
20. Purwins, H., Li, B., Virtanen, T., Schlüter, J., Chang, S.Y., Sainath, T.: Deep learning for audio signal processing. IEEE J. Sel. Top. Sig. Process. **13**(2), 206–219 (2019). https://doi.org/10.1109/JSTSP.2019.2908700
21. Rosenzweig, S., Schwär, S., Driedger, J., Müller, M.: Adaptive pitch-shifting with applications to intonation adjustment in a cappella recordings. In: 2021 24th International Conference on Digital Audio Effects (DAFx), pp. 121–128. IEEE (2021). https://doi.org/10.23919/DAFx51585.2021.9768268
22. Roucos, S., Wilgus, A.: High quality time-scale modification for speech. In: ICASSP'85. IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 10, pp. 493–496. IEEE (1985). https://doi.org/10.1109/ICASSP.1985.1168381
23. Schedl, M.: Deep learning in music recommendation systems. Front. Appl. Math. Stat. 44 (2019). https://doi.org/10.3389/fams.2019.00044
24. Verhelst, W., Roelands, M.: An overlap-add technique based on waveform similarity (WSOLA) for high quality time-scale modification of speech. In: 1993 IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 2, pp. 554–557. IEEE (1993). https://doi.org/10.1109/ICASSP.1993.319366
25. Wager, S., Tzanetakis, G., Wang, C.i., Kim, M.: Deep autotuner: a pitch correcting network for singing performances. In: ICASSP 2020–2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 246–250. IEEE (2020). https://doi.org/10.1109/ICASSP40776.2020.9054308
26. Zhou, F., Torre, F.d.l.: Canonical time warping for alignment of human behavior. In: Proceedings of the 22nd International Conference on Neural Information Processing Systems. NIPS 2009, pp. 2286–2294. Curran Associates Inc., Red Hook (2009). https://doi.org/10.5555/2984093.2984349
27. Zou, F., Shen, L., Jie, Z., Zhang, W., Liu, W.: A sufficient condition for convergences of Adam and RMSPROP. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11127–11135 (2019). https://doi.org/10.48550/arXiv.1811.09358