11th ISCA Speech Synthesis Workshop (SSW 11)
26-28 August 2021, Budapest, Hungary

Text-to-Speech Synthesis Techniques for MIDI-to-Audio Synthesis

Erica Cooper[†]*, Xin Wang*[†]*, Junichi Yamagishi*

National Institute of Informatics, Chiyoda-ku, Tokyo, Japan
ecooper@nii.ac.jp, wangxin@nii.ac.jp, jyamagis@nii.ac.jp

Abstract

Speech synthesis and music audio generation from symbolic input differ in many aspects but share some similarities. In this study, we investigate how text-to-speech synthesis techniques can be used for piano MIDI-to-audio synthesis tasks. Our investigation includes Tacotron and neural source-filter waveform models as the basic components, with which we build MIDI-to-audio synthesis systems in similar ways to TTS frameworks. We also include reference systems using conventional sound modeling techniques such as sample-based and physical-modeling-based methods. The subjective experimental results demonstrate that the investigated TTS components can be applied to piano MIDI-to-audio synthesis with minor modifications. The results also reveal the performance bottleneck – while the waveform model can synthesize high quality piano sound given natural acoustic features, the conversion from MIDI to acoustic features is challenging. The full MIDI-to-audio synthesis system is still inferior to the sample-based or physical-modeling-based approaches, but we encourage TTS researchers to test their TTS models for this new task and improve the performance.

Index Terms: music audio synthesis, text to speech synthesis, deep learning, Tacotron

1. Introduction

Speech and music are human universals, and they have been the theme of numerous research topics in the social and natural sciences. From an engineering perspective, both speech and music information processing deal with symbolic and acoustic data, i.e., symbolic music notes or text, and acoustic music or speech audio signals. This similarity makes it possible to share methodologies across disciplines, especially those based on data-driven deep learning. For example, AI music composition, which learns a distribution of music notes to generate new songs, is based on language models for text data. Automatic audio transcription, which converts music into a sequence of notes, uses similar techniques to speech recognition such as Viterbi decoding and DNNs for classification tasks.

From the music notes to instrumental audio signals[1], however, cross-disciplinary techniques are less explored even though the concept is very similar to text-to-speech (TTS) synthesis. It has not been until recently that some deep learning models, such as WaveNet [1, 2], have been used in both tasks. While research in both fields has investigated other related models such as GAN [3, 4] and VAE [5, 6], most studies focus on music audio-to-audio mapping tasks. The question of how and to what extent TTS approaches can be applied to music audio generation remains to be explored.

This study is our initial step to address the aforementioned question. We focus on MIDI-to-audio synthesis for piano because of the available data, but the methodology is expected to be applicable to many other instruments. In Section 2, we compare TTS to music audio generation from MIDI and explain the possibility of using TTS methods for the MIDI synthesis task. In Section 3, we explain the MIDI-to-audio systems that use many components from TTS, including Tacotron [7] and neural source-filter (NSF) waveform model [8]. We introduce modifications to those components to account for the intrinsic differences between music and speech. Furthermore, we introduce MIDI-specific acoustic features and compare them with the Mel-spectrogram for MIDI-to-audio synthesis.

Based on a subjective evaluation, our study tentatively suggests that many TTS techniques can be adapted to music audio generation with slight modifications, and we observed trends similar to those in TTS. For audio generation, the NSF models, which were originally created for speech modeling, can produce high-quality polyphonic piano sound in the copy-synthesis scenario. For acoustic modeling, the Tacotron-based models demonstrated competitive performance to produce acoustic features from the MIDI piano roll input. However, the conversion from MIDI to acoustic features is the most challenging task, similar to the bottleneck in TTS. Hence, we encourage TTS researchers to extend their research outcomes to the music audio generation task.

2. Using TTS techniques for MIDI-to-audio synthesis

This paper focuses on music audio generation from music transcription data in MIDI format. As illustrated in Figure 1, the MIDI-to-audio synthesis process is similar to TTS since both convert symbolic data into audio signals.

In both cases, the front-end converts the input text or MIDI raw data into a representation as input to the acoustic model. In the case of statistical parametric TTS [9], it is a sequence of context vectors $\boldsymbol{x}_{1:N} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N)$, where N is the number of frames, and where the vector for the n-th frame is $\boldsymbol{x}_n$. Each $\boldsymbol{x}_n$ encodes linguistic information such as the phone and syllable identities. In the case of the MIDI-to-audio generation, the input MIDI contains messages that encode the time of note onset and offset, velocity, and other events. For processing using deep learning models, the MIDI input is usually represented as a piano roll[2] that can also be written as a sequence of vectors $\boldsymbol{x}_{1:N}$, and each $\boldsymbol{x}_n$ is a 128-dimensional vector in which each dimension encodes the velocity for one of the 128 MIDI notes[3]. Figure 2 illustrates the difference.

Accordingly, conversion from $\boldsymbol{x}_{1:N}$ to $\boldsymbol{o}_{1:T}$ can use similar models for both tasks. These methods can be grouped into two

[†]Equal contribution

[1]In this paper, we focus on musical instrument sounds rather than singing voice synthesis.

[2]Although it is called piano roll, it can be used for other instruments.

[3]The sustain pedal information can be reflected as elongation of notes, encoded by extending the note across multiple frames.

10.21437/SSW.2021-23

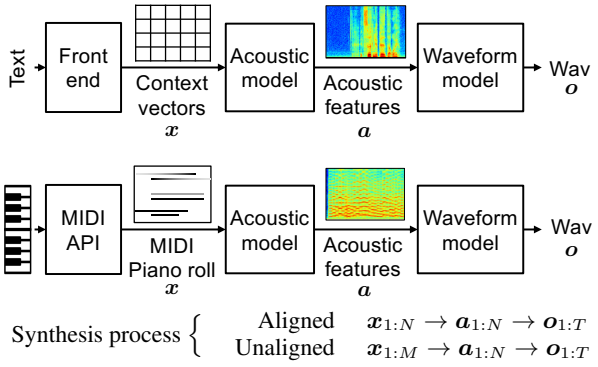

Figure 1: *Comparing TTS and MIDI-to-audio synthesis. Temporal length of synthesized waveform $\boldsymbol{o}_{1:T}$ and acoustic feature $a_{1:N}$ is related by the frame rate L, i.e., $T = N \times L$.*

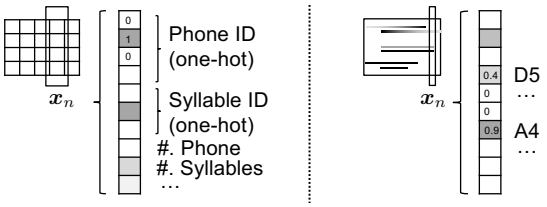

Figure 2: *Comparing contextual vector for statistical parametric TTS (left) and MIDI piano roll (right).*

categories from the perspective of MIDI-to-audio generation. The first group is for applications where the audio is required to be consistent with the timing information in the MIDI input. In implementation, the synthesis process has to be $\boldsymbol{x}_{1:N} \rightarrow \boldsymbol{a}_{1:N} \rightarrow \boldsymbol{o}_{1:T}$, where the MIDI piano roll and acoustic feature sequences have the same length N, and where the audio waveform length T is related to the feature length by a fixed frame shift L, i.e., $T = N \times L$. This pipeline is very similar to neural statistical parametric TTS [9], and we can use various types of neural networks [10, 11] for $\boldsymbol{x}_{1:N} \rightarrow \boldsymbol{a}_{1:N}$ and use neural waveform models [12, 13] or vocoders [14, 15] for $\boldsymbol{a}_{1:N} \rightarrow \boldsymbol{o}_{1:T}$. Note that $\boldsymbol{x}_{1:N}$ for TTS should have obtained the duration information from a duration model, while $\boldsymbol{x}_{1:N}$ for MIDI can retrieve the duration information from raw MIDI.

The second type of MIDI-to-audio application converts MIDI corresponding to a basic musical score into a professional "performance" with rich and dynamic expressivity, or transfers a particular "performance style" to the generated audio. In this case, $\boldsymbol{x}_{1:M}$ from input MIDI may not be aligned with the desired output sequence $\boldsymbol{a}_{1:N}$. Accordingly, the acoustic model should be capable of $\boldsymbol{x}_{1:M} \rightarrow \boldsymbol{a}_{1:N}$, and many attention-based sequence-to-sequence models [7, 16] are suitable for this task.

Note that it is also possible to use a single model to directly convert the the input piano roll into an audio waveform, which is similar to WaveNet for TTS [1].

3. Experimental systems for time-aligned MIDI to piano audio synthesis

In this paper, we focus on systems that convert time-aligned MIDI to musical instrument audio waveforms and choose piano as the target instrument. Figure 3 illustrates the systems as four groups. While all of them use an NSF-based waveform model, the first two types contain intermediate acoustic models, and the remaining two directly convert the input MIDI into an output waveform. Another difference among the systems is the type

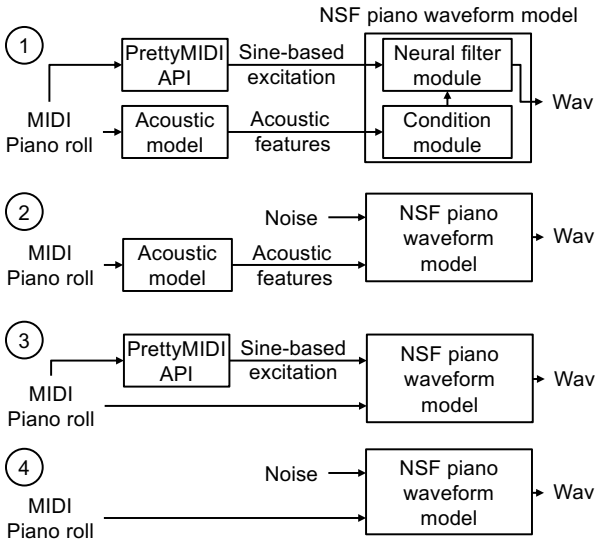

Figure 3: *Four types of experimental MIDI-to-audio systems using NSF piano waveform model.*

of the excitation signal, which will be detailed in Section 3.3. Note that the experimental systems introduced in this section are applicable to other monophonic or polyphonic instruments.

3.1. Acoustic models

For the systems that use separate acoustic models, we can use many types of neural networks that conduct $\mathbb{R}^{N \times D} \rightarrow \mathbb{R}^{N \times D'}$. Although it is originally designed for sequence-to-sequence mapping, we investigate variants of Tacotron, hoping that the outcome can accelerate our research on the unaligned MIDI-to-audio synthesis task in the near future.

Our Tacotron acoustic model implementation was based on [17]. Tacotron was modified to accept a sequence of 128-dimensional MIDI piano roll frames as input instead of a text or phoneme sequence. Tacotron's encoder learns an embedding that maps a symbolic token to an embedding vector, but since piano roll frames are not strictly symbolic but are already a meaningful vector representation of pitch and velocity, we replaced this embedding layer with a dense projection layer. Aside from these initial modifications of the encoder to accept MIDI piano roll input instead of text, the model architecture is otherwise the same as in [17]. The encoder consists of a CBHG module followed by a self-attention block. The outputs of both are input to the decoder, where the CBHG output is input to a forward attention mechanism and the self-attention output is received by an additive attention mechanism. The decoder consists of a decoder recurrent layer followed by self-attention, and finally, a post-net conducts spectral shaping and enhancement for the final spectrogram output.

Because music sequences are much longer than the typical short utterances used for training text-to-speech synthesis models, and in particular, the piano roll input representation is the same length as the output spectrogram, as opposed to text or phoneme sequences which are much shorter, we had to take a number of steps to fit the training sequences into memory and to learn alignments well. First, we found that it was necessary to segment the data into shorter sequences, starting with 200 frames. Next, we wished to reduce the autoregressive dependencies and instead force the model to rely more directly on the inputs. The prenet to the decoder receives

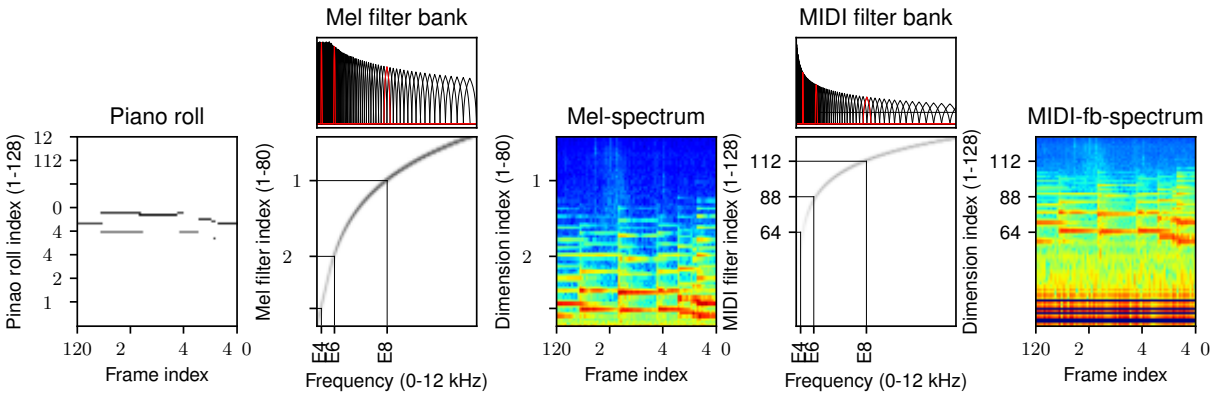

Figure 4: *Mel-spectrogram versus MIDI-filter-bank spectrogram*

the previous predicted spectrogram frame (or the previous actual spectrogram frame, when teacher-forcing is enabled), so we increased dropout at the prenet to the decoder from its default value of 0.5 to higher values of 0.9, 0.95, or 0.99, finding in initial experiments that the 0.99 dropout rate produced the best-aligned predictions. Next, we tried a downsampling approach where we downsampled the input piano roll sequence by a factor of either 2 or 4, and effectively downsampled the output spectrogram as well by setting the reduction factor to 2 or 4, so that the model predicts that many output frames at each timestep. We found that a downsampling factor of 4 produced the best alignments, and that combining this downsampling with the dropout to the decoder prenet at a rate of 0.99 produced stable alignments for sequences as long as 800 frames, so we chose this as our base model, which we call `taco2`.

As an additional method to force the model to use the input sequence more directly, we concatenated the current MIDI piano roll frame with the previous spectrogram frame at the decoder prenet (after that spectrogram frame has been dropped out). We call this model configuration `taco3`, and we warm-start its parameters from `taco2`. Finally, to see whether up-sampling the data again would improve the synthesis quality, model `taco3` uses none of these prenet or downsampling tricks, and its parameters are also warm-started from `taco2`.

As reference, we included a CNN-based network that conducts $\mathbb{R}^{N\times D} \rightarrow \mathbb{R}^{N\times D'}$. This network called PerformanceNet combines U-Net and multi-band convolution blocks to convert MIDI piano rolls into acoustic features and has shown good performance on MIDI-to-audio synthesis [18]. Samples from all systems can be heard online[4].

3.2. MIDI filter-bank features

Many TTS systems use Mel-spectrogram or Mel-cepstral coefficients as the output of the acoustic model, but the Mel scale may not be the best for music applications. Since each MIDI note d and its corresponding frequency f is related by

$$f = 2^{\frac{d-69}{12}} \times 440, \tag{1}$$

where 69 and 440 are the MIDI index and frequency value of note A4, respectively, we can define a new filter bank where the k-th filter is centered around $f = 2^{(k-69)/12} \times 440$ Hz. Figure 4 plots the resulting MIDI-based filter bank.

We apply the MIDI-based filter bank to extract low-dimensional spectral features in a similar manner to the Mel-spectrogram. Figure 4 also compares the Mel-spectrogram and MIDI-filter-bank-based spectra. Notice that the bars in the MIDI-filter-bank-based spectra resemble the corresponding piano roll. Note that the MIDI-filter-bank spectra have empty dimensions in the low frequency region because some of the filters do not cover any discrete frequency bin. These empty dimensions can be filled in if we increase the FFT points.

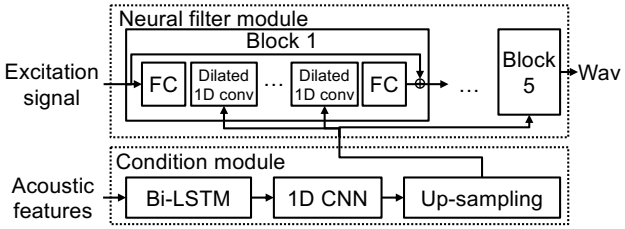

Figure 5: *NSF piano waveform model. Block in neural filter module is based on the simplified dilated-CNN filter block (cf. Figure.4 in [8]). FC denotes a fully-connected layer.*

3.3. NSF-based piano waveform model

Our NSF-based piano waveform model is based on the simplified NSF model [8]. As Fig. 5 illustrates, the NSF piano waveform model contains a condition module that transforms and up-samples the input frame-level acoustic features and a neural filter module that converts the up-sampled features and an excitation signal into an output waveform through multiple dilated convolution blocks.

A major difference between NSF waveform models for piano and speech is the source module. While the source module for speech produces an excitation signal from fundamental frequencies (F0s), such a module cannot be used for polyphonic piano sound. One solution is to use noise as the excitation (e.g., ② and ④ in Fig. 3). An alternative solution is to derive a sine-based excitation signal from the input time-aligned MIDI. In this paper, we use the PrettyMIDI synthesis API [19][5] to produce a polyphonic sine-based excitation signal (e.g., ① and ③ in Fig. 3) from the piano roll notes.

4. Experiments

4.1. Database and protocol

Experiments were conducted using the MAESTRO database (V2.0.0)[6] [2]. This is a large-scale database that contains

[4] https://nii-yamagishilab.github.io/samples-xin/main-midi2audio.html

[5] https://craffel.github.io/pretty-midi/

[6] https://magenta.tensorflow.org/datasets/maestro

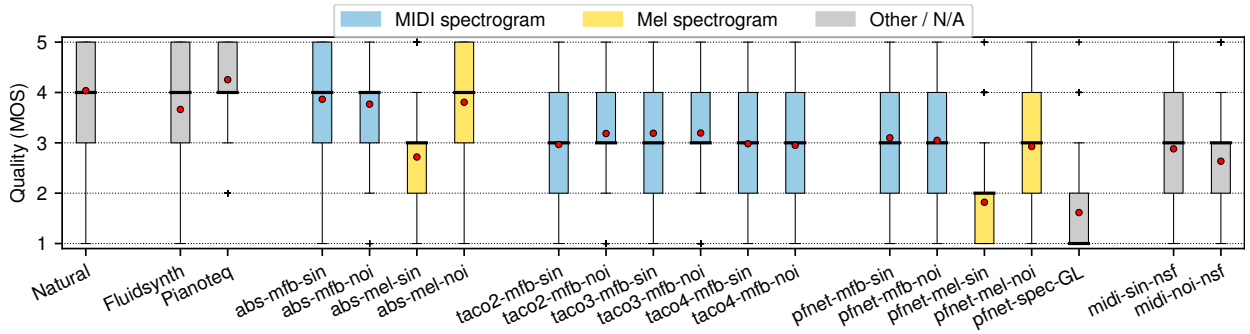

Figure 6: *Boxplots of MOS per system. Red dots denote mean of MOS.*

over 200 hours of piano performances and aligned MIDI data from the International Piano-e-Competition. Both the audio and MIDI data were recorded when the competing virtuoso pianists performed on concert-quality acoustic grand pianos with integrated MIDI capture and playback systems.

The experiments followed the official data protocol: a train set with 161.3 hours of data from 967 performances, a validation set with 19.4 hours of data from 137 performances, and a test set with 20.5 hours of data. Because it is impossible to evaluate the entire test set in subjective evaluation, 192 test segments were manually excerpted from the test set, and each test segment was less than 30 seconds in duration.

4.2. Experimental systems and feature configurations

Table 1 lists the systems investigated in the experiments. The first two are reference software synthesizers, and the next four are copy-synthesis systems that directly use natural acoustic features (i.e., Mel-spectrogram or MIDI-based filter bank features) as the input to the NSF model for piano waveform generation. They simulate the ideal case where acoustic models convert the input MIDI into the acoustic features perfectly. The next 11 systems are pipelines of an acoustic model, which is either a variant of the Tacotron or the PerformanceNet model, and the NSF waveform model. The last two experimental systems, namely `midi-sin-nsf` and `midi-noi-nsf`, directly convert the MIDI and the excitation signals into the waveform through NSF models.

We trained Tacotron models using the MIDI filter bank spectrogram as output, since we found initially that this produced better alignments than using Mel spectrograms.[7] The models were trained on segments of 800 frames using the Adam optimizer, a batch size of 4, and a learning rate of 0.0001. The base model `taco2` was trained for 550k steps until spectrogram loss on the development set had converged. The other two models `taco3` and `taco4` had their parameters warm-started from `taco2`, and were trained for an additional 250k steps and 50k steps to convergence, respectively. The PerformanceNet-based acoustic models (`PFNet`) were trained for 50 epochs using the Adam optimizer with a batch size of 16.

All the NSF models were trained using an Adam optimizer with a learning rate of 1×10^{-4} and [20]. The maximum number of training epochs is 20, and each epoch took around 24 hours. Due to the limited GPU memory size, the input and output data for NSF models was truncated into segments of 3s in duration, and the batch size was set to 5. During generation, the NSF models produced the waveform without truncation.

As a reference, the original PerformanceNet was included in the experiment. This system uses spectrograms as the acoustic feature and produces a waveform from the generated spectrogram through the Griffin-Lim (GL) algorithm [21]. Two software synthesizers were also included for reference: an open-source software called `Fluidsynth`[8] and a commercial one called `Pianoteq`[9]. While both produce piano audio given MIDI input, the former uses a sampling-based approach, and the latter is based on physical modeling of pianos.

Table 1: *Experimental systems. Pitch accuracy is measured using cross-entropy, the lower the better.*

System ID	Acoustic model	Acoustic feature	Excit. signal	Wave. model	Pitch mismatch note	Pitch mismatch chord	MOS (mean)
Natural	-	-	-	-	-	-	4.04
`Fluidsynth`	Sample-based MIDI-to-audio software				5.20	6.77	3.66
`Pianoteq`	Physical-model MIDI-to-audio software				4.82	6.50	4.25
`abs-mfb-sin`	-	midi-fb	sine	NSF	-	-	3.87
`abs-mfb-noi`	-	midi-fb	noise	NSF	-	-	3.77
`abs-mel-sin`	-	mel-spc.	sine	NSF	-	-	2.72
`abs-mel-noi`	-	mel-spc.	noise	NSF	-	-	3.81
`taco2-mfb-sin`	`taco2`	midi-fb	sine	NSF	4.61	6.34	2.97
`taco2-mfb-noi`	`taco2`	midi-fb	noise	NSF	4.66	6.36	3.18
`taco3-mfb-sin`	`taco3`	midi-fb	sine	NSF	4.78	6.48	3.19
`taco3-mfb-noi`	`taco3`	midi-fb	noise	NSF	4.89	6.53	3.19
`taco4-mfb-sin`	`taco4`	midi-fb	sine	NSF	4.86	6.39	2.98
`taco4-mfb-noi`	`taco4`	midi-fb	noise	NSF	4.97	6.42	2.95
`pfnet-mfb-sin`	PFNet	midi-fb	sine	NSF	5.59	7.14	3.10
`pfnet-mfb-noi`	PFNet	midi-fb	noise	NSF	5.78	7.26	3.05
`pfnet-mel-sin`	PFNet	mel-spec.	sine	NSF	5.66	7.17	1.82
`pfnet-mel-noi`	PFNet	mel-spec.	noise	NSF	5.74	7.25	2.93
`pfnet-spec-GL`	PFNet	spec.	-	GL	5.43	6.98	1.62
`midi-sin-nsf`	-	-	sine	NSF	4.32	6.40	2.88
`midi-noi-nsf`	-	-	noise	NSF	4.40	6.08	2.63

The audio waveforms from MAESTRO were down-sampled to 24kHz and encoded through 16-bit PCM. The Mel-spectrogram was then extracted using a frame length of 50ms, a frame shift of 12 ms, FFT with 2048 points, and 80 overlapped triangular filters evenly spaced on the Mel-frequency scale. The MIDI-based filter bank features were extracted using a similar configuration but with a filter bank based on the MIDI notes (Section 3.2). The spectrogram for `pfnet-spec-GL` used the original recipe [18]. The MIDI files were converted into 128-dimensional piano rolls using the PrettyMIDI API.

4.3. Subjective evaluation and results

We conducted a crowdsourced listening test to evaluate the quality of audio from our synthesizers, comparison synthesizers, and natural audio. We included 120 samples from each of the 20 systems, and obtained mean opinion score (MOS) ratings for each sample from five different listeners on a scale from 1 (very bad) to 5 (very good). Listeners were instructed

[7] Alignment errors were reduced from 21% using Mel spectrograms to 3% using MIDI spectrograms.

[8] https://www.fluidsynth.org/

[9] https://www.modartt.com/pianoteq

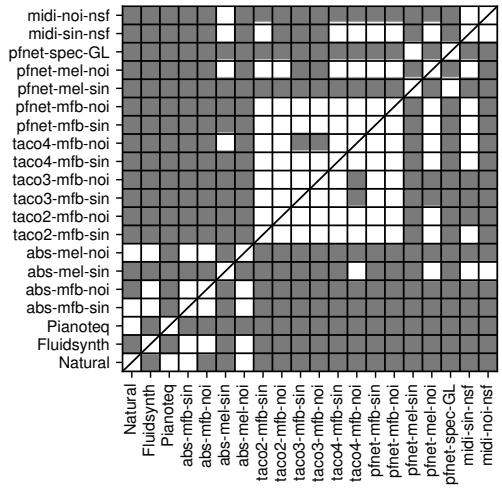

Figure 7: *Results of two-sided Mann-Whitney test with Holm-Bonferroni correction. A grey block indicates a statistically significant difference at* $\alpha = 0.05$.

to evaluate the overall quality of the piano sound subjectively. Each test set contained 30 different samples, balanced to contain at least one sample from each of the 20 systems. Listeners were permitted to complete up to 10 different sets. In total, we received ratings from 224 unique listeners. A box plot of the MOS ratings for each of the 20 systems can be seen in Figure 6, and significant differences from a Holm-Bonferroni-corrected two-sided Mann-Whitney U test at a level of $\alpha = 0.05$ are shown in Figure 7. The mean of MOS is also listed in Table 1.

We found that the `Pianoteq` physical synthesis was significantly preferred over every other synthesis method, and was even rated higher than natural piano audio (although not significantly so). Other systems which were not significantly different from natural audio were two of the analysis-by-synthesis systems, `abs-mfb-sin` and `abs-mel-noi`. There were not many statistically-significant differences between the Tacotron models, with the exception of `taco4-mfb-noi`, which was significantly worse than both of the `taco3` systems. As for PFNet-based systems, the two that used the MIDI filter bank representation had about equivalent performance to the Tacotrons with no significant differences, whereas `pfnet-mel-sin` and `pfnet-spec-GL` were significantly worse than all Tacotrons.

Comparing the standard Mel filter bank to the proposed MIDI filter bank, there are two significant differences favoring MIDI, `pfnet-mel-sin` vs. `pfnet-mfb-sin` and `abs-mel-sin` vs. `abs-mfb-sin`. For noise vs. sine wave excitation, there are two significant differences favoring noise, `pfnet-mel-sin` vs. `pfnet-mel-noi` and `abs-mel-sin` vs. `abs-mel-noi`.

4.4. Objective evaluation and results

We first measured the pitch accuracy of the synthesized audio. We trained a CNN-based F0 estimator called CREPE [22] on the MAESTRO training set. Although the original CREPE is designed for monophonic sound, it can be modified for polyphonic piano sound by replacing the target from one-hot vectors to multi-hot ones. During the pitch detection stage, we extract the pitch probability sequence $\boldsymbol{p}_{1:N} = (\boldsymbol{p}_1, \cdots, \boldsymbol{p}_N)$ from the input audio $\boldsymbol{o}_{1:T}$ by $\boldsymbol{p}_{1:N} = \texttt{CREPE}(\boldsymbol{o}_{1:T})$, where each $\boldsymbol{p}_n = [p_{n,1}, \cdots, p_{n,128}]$, and where $p_{n,k} \in (0, 1)$ indicates the probability of observing the k-th MIDI note at the n-th frame. Then, the cross entropy between $\boldsymbol{p}_{1:N}$ and the input piano roll $\boldsymbol{x}_{1:N}$ can be computed to measure the pitch mismatch $\texttt{CE} = -\sum_{n=1}^{N}\sum_{k=1}^{128} x_{n,k} \log p_{n,k}$. Hence, a lower `CE` indicate less severe mismatch.

We created piano rolls and synthesized audio for around 100 individual notes and chords and synthesized their audio. We measured cross entropy and listed results in Table 1. It can be observed that, when using separate acoustic and waveform models, the systems with sine excitation outperformed their counterparts using noise excitation, but the systems using only the NSF waveform model achieved lower `CE` values. Furthermore, Tacotron models have lower mismatches compared to PFNet systems using the same vocoder. However, lower pitch mismatch does not lead to higher MOS. This indicates that the perceptual quality of piano audio is not only affected by pitch accuracy. Another hypothesis is that amateur listeners may not be able to detect mild pitch mismatch.

5. Discussion

The evaluation results suggest that the physical-model-based approach outperformed the other deep-learning-based MIDI-to-audio systems and is even slightly better than the original audio in MAESTRO. The original audio was recorded over many years, and we observed that the MOS of the audio in year 2008 and 2014 were less than 4.0. This variation of quality may also affect the MAESTRO training set and the deep-learning-based models trained using this data. On the other hand, the physical-model-based approach is free from such artifacts in data. However, the physical model is the outcome of laborious analysis and simulation [23, 24, 25], which does not easily generalize to another piano type or instrument. In contrast, deep-learning-based models are more flexible, and this study showed examples of using TTS models for music generation.

Performance of the investigated deep-learning models is not satisfying yet. Since listeners may be more sensitive to the artifacts in music signals, this sets a high standard for producing natural audio but also leaves large room for improvement. One potential direction for future work is to combine data-driven techniques with physical models of piano sounds. Rather than learning a physical sound from scratch, sound physics may offer effective prior knowledge.

6. Conclusions

This study is our initial step to investigate the possibility of using TTS models for MIDI-to-audio synthesis. The two disciplines differ in many aspects but both deal with the mapping from one sequence of data into another. Based on the similarities and differences, we modified the TTS components, namely Tacotron and NSF, and introduced a MIDI filter-bank acoustic feature set for the MIDI-to-audio task, which improved alignments for Tacotron and resulted in more preferred audio. Based on subjective evaluation of the TTS-like systems, natural audio, physical piano model, and other reference systems, we observed promising results when using TTS components for MIDI-to-audio generation. We also identified the quality bottleneck when converting the MIDI input into acoustic features, a similar bottleneck to that in TTS. Hence, future work will explore more different types of acoustic and waveform models, and we encourage TTS researchers to extend their knowledge and practices to this challenging task of MIDI-to-audio generation.

7. Acknowledgements

A part of the computations were performed on the TSUBAME 3.0 supercomputer at Tokyo Institute of Technology. This work is supported by "TSUBAME Encouragement Program for Young/Female Users" of Global Scientific Information and Computing Center at Tokyo Institute of Technology and by "Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures" in Japan, and by JST CREST Grants (JPMJCR18A6 and JPMJCR20D3), JST AIP Challenge Grant, and MEXT KAKENHI Grants (18H04112, 21H04906, 21K11951, 21K17775), and a grant by KAWAI foundation for sound technology & music (year 2020, No.4), Japan.

8. References

[1] A. van den 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 preprint arXiv:1609.03499*, 2016.

[2] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *Proc. ICLR*, 2018.

[3] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial neural audio synthesis," in *Proc. ICLR*, 2019.

[4] 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 *Proc. NIPS*, 2019, pp. 14 910–14 921.

[5] F. Roche, T. Hueber, S. Limier, and L. Girin, "Autoencoders for music sound modeling: a comparison of linear, shallow, deep, recurrent and variational models," in *Proc. SMC*, 2019. [Online]. Available: http://www.gipsa-lab.fr/{∼}fanny.roche/SMC{_}2019.html

[6] A. Bitton, P. Esling, and T. Harada, "Vector-Quantized Timbre Representation," *arXiv preprint arXiv:2007.06349*, 2020.

[7] Y. Wang, R. J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: Towards End-to-End Speech Synthesis," in *Proc. Interspeech*, 2017, pp. 4006–4010.

[8] 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. [Online]. Available: https://ieeexplore.ieee.org/document/8915761/

[9] H. Zen, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *Proc. ICASSP*, 2013, pp. 7962–7966.

[10] H. Zen and A. Senior, "Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis," in *Proc. ICASSP*, 2014, pp. 3844–3848.

[11] Y. Fan, Y. Qian, F. Xie, and F. K. Soong, "TTS Synthesis with Bidirectional LSTM Based Recurrent Neural Networks," in *Proc. Interspeech*, 2014, pp. 1964–1968.

[12] A. Tamamori, T. Hayashi, K. Kobayashi, K. Takeda, and T. Toda, "Speaker-dependent WaveNet vocoder," in *Proc. Interspeech*, 2017, pp. 1118–1122.

[13] R. Prenger, R. Valle, and B. Catanzaro, "WaveGlow: A Flow-based Generative Network for Speech Synthesis," in *Proc. ICASSP*, 2019, pp. 3617–3621.

[14] H. Kawahara, I. Masuda-Katsuse, and A. de Cheveigne, "Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds," *Speech Communication*, vol. 27, pp. 187–207, 1999.

[15] M. Morise, F. Yokomori, and K. Ozawa, "WORLD: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Trans. on Information and Systems*, vol. 99, no. 7, pp. 1877–1884, 2016.

[16] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6706–6713.

[17] Y. Yasuda, X. Wang, S. Takaki, and J. Yamagishi, "Investigation of enhanced Tacotron text-to-speech synthesis systems with self-attention for pitch accent language," in *Proc. ICASSP*. IEEE, 2019, pp. 6905–6909.

[18] B. Wang and Y.-H. Yang, "PerformanceNet: Score-to-audio music generation with multi-band convolutional residual network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 1174–1181.

[19] C. Raffel and D. P. W. Ellis, "Intuitive analysis, creation and manipulation of midi data with pretty midi," in *Proc. ISMIR late breaking and demo papers*, 2014, pp. 84–93.

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

[21] D. Griffin and J. Lim, "Signal estimation from modified short-time Fourier transform," *IEEE Trans. ASSP*, no. 2, pp. 236–243, 1984.

[22] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "CREPE: A convolutional representation for pitch estimation," in *Proc. ICASSP*. IEEE, 2018, pp. 161–165.

[23] A. Chaigne and A. Askenfelt, "Numerical simulations of piano strings. I. A physical model for a struck string using finite difference methods," *The Journal of the Acoustical Society of America*, vol. 95, no. 2, pp. 1112–1118, 1994.

[24] N. Giordano and M. Jiang, "Physical Modeling of the Piano," *EURASIP Journal on Advances in Signal Processing*, vol. 2004, no. 7, p. 981942, dec 2004. [Online]. Available: https://asp-eurasipjournals.springeropen.com/articles/10.1155/S111086570440105X

[25] J. Chabassier and M. Duruflé, "Physical parameters for piano modeling," Ph.D. dissertation, INRIA, 2012.