An Audio-to-Audio Approach to Generate Bass Lines from Guitar's Chord Backing

Tomoo Kouzai[1] and Tetsuro Kitahara[1] *

[1]Graduate School of Integrated Basic Sciences, Nihon University
Setagaya-ku, Tokyo, Japan
{kouzai,kitahara}@kthrlab.jp

Abstract. In this paper, we address a system that generates a bass line from a chord backing played on the electric guitar in an audio-to-audio manner. Yielding bass lines for guitar chord backings would be helpful for amateur musicians composing band music. Conventional music arrangement systems targeted MIDI-like symbolic music representations, but accurately obtaining symbolic representations from guitars takes work. To solve this problem, we attempt an audio-to-audio approach; Once the user gives an audio recording of the guitar's chord backing, the system extracts some audio features (spectrogram, mel-spectrogram, or chromagram) and then generates an audio signal of bass lines using a convolutional neural network. The experimental results showed that the model with chromagrams generates bass lines the most robustly.

Keywords: music, CNN, guitar, band arrangements, audio-to-audio

1 Introduction

The electric guitar is one of the central instruments in light music, especially in band music. Therefore many amateur guitarists enjoy playing in a band. When they try to play their original songs in a band, a particular member (such as the guitarist) often composes a melody and a chord progression. They often collaboratively decide the phrases of instrumental parts (e.g., bass, drums). However, it is a challenging task because it requires musical knowledge, like typical phrases of each instrument. If the phrases of each instrument part can be automatically decided on a computer and the band members can listen to them, creating original songs may be more efficient.

Most of the existing studies on automatic music arrangements have been for the piano, such as piano arrangement from band or orchestra pieces[1, 2] and score reduction of piano pieces for beginners[3]. Although some studies targeting guitar, most of them are systems for arranging solo guitar scores, such as generating solo guitar scores

* This work was supported by JSPS Kakenhi Nos. JP22H03711 and JP21H03572.


This work is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

from orchestral scores[4]and from audio signals[5]. No research has been done on band arrangements that add drums, bass, or other sounds from the guitar sound.

Our goal is to develop a system that automatically makes band arrangements for a given song. This system is intended to be used by people who can play a simple backing for their original song on the guitar but cannot create phrases of other instruments, such as the bass and drums. As the first step, this paper addresses a system that generates a bass line for a given chord backing played on the guitar.

Most existing systems of music arrangement use MIDI-like symbol music representations, but it is not easy to accurately obtain a symbolic representation from recordings of guitar performances (Although there are commercial products of MIDI guitars, their audio-to-MIDI conversion is not necessarily accurate). We, therefore, adopt an audio-to-audio approach in which both inputs (guitar backings) and outputs (bass lines) are audio signals.

2 Proposed Method

Given an audio signal of a chord performance played on the electric guitar, our method generates an audio signal of a bass performance that fits the given guitar performance. For simplicity, the tempo and length are fixed (120 BPM and four measures in the current implementation). First, the given guitar signal is converted to a feature representation (i.e., spectrogram, mel-spectrogram, or chromagram). Then, it is segmented by 0.5 seconds, and each segment is input to a convolutional neural network (CNN), which generates a bass spectrogram. Finally, the bass spectrogram is converted to an audio signal. To train the CNN model, we use a pairwise dataset consisting of guitar feature representations and bass spectrograms.

2.1 Calculation of the spectrogram of the input sound source

The spectrogram is computed from a given guitar audio signal(and the bass source when learning) using the short-time Fourier transform (STFT) after downsampled to 22050 Hz. The Hann window is used. The window size is set to 2048, and the hop size is set to 1/1000 of the sampling frequency.

2.2 Feature extraction

We attempt three different feature representations:

- Spectrogram: The amplitude spectrogram obtained in Section 2.1 is used without conversions.
- Mel-spectrogram: This is calculated from the spectrogram using Librosa.
- Chromagram: This is also calculated from the spectrogram using Librosa. The hop size for the chromagram is set to 512.

Below, the models with a spectrogram, a mel-spectrogram, and a chromagram are called the *STFT model*, *Mel model*, and *Chroma model*, respectively.

2.3 Generation of bass spectrogram

The feature representation (spectrogram, mel-spectrogram, or chromagram) of the given guitar signal is converted into a spectrogram of a bass performance using a CNN model, because CNNs are widely used for analyzing spectrograms[6–11]. Our CNN model (Figure 1) consists of convolution layers and deconvolution layers as follows:

Convolution layers For the *STFT* model, the convolution layers consist of:

- 1st layer: The frequency axis of the guitar's feature representation is compressed to one dimension. The input spectrogram of dimension 1025×500 (frequency axis: 1025, time axis: 500) is compressed to a 1×500 matrix with a 1025×1 filter.
- 2nd layer: The 1×500 matrix is compressed to a 1×250 with a 1×2 filter.
- 3rd layer: The 1×250 matrix is compressed to a 1×50 with a 1×5 filter.
- 4th and later layers: A 1×2 filter and a 1×5 filter are alternately applied until a 1×5 matrix is obtained.

The number of filter channels in each layer is 1024. The stride is 1.No padding is used. A ReLU function is used for the activation.

For the *Mel* model, the filter size for the 1st layer is 128×1 because the input matrix size is 128×500. Apart from this, the same configurations are used.

For the *Chroma* model, the following convolution layers are used:

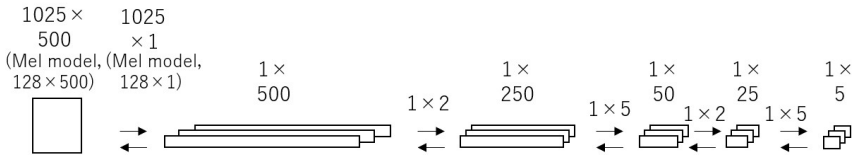
- 1st layer: The 12×22 chromagram is compressed to a 1×22 (filter size: 12×1).
- 2nd layer: The 1×22 matrix is compressed to a 1×11 matrix (filter size: 1×2).
- 3rd layer: The 1×11 matrix is compressed to a 1×2 matrix (filter size: 1×5).

Deconvolution layers The set of deconvolution layers generates a bass spectrogram independently of the feature representation used for guitar signals. It consists of:

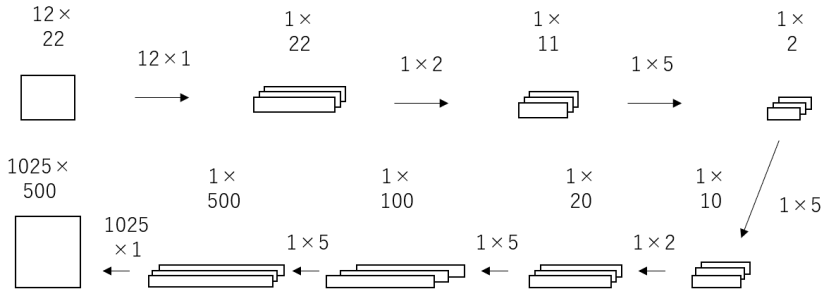
- Multiple decomposition layers with filter sizes of 1×5 and 1×2 are alternatively applied. These layers converts a 1×5 matrix (a 1×2 matrix for the *Chroma* model) to a 1×500 matrix.
- After that, a deconvolution layer expanding the frequency axis is applied. This layer has a filter size of 1025×1, which converts a 1×500 matrix to a 1025×500 matrix. This matrix represents a bass spectrogram.

2.4 Generation of the bass's audio signals

The audio signal of the bass part is obtained by using inverse Fourier transform and phase restoration on the spectrogram generated from the CNN. The Griffin-Lim algorithm is used for phase restoration. The number of iterations is 32, the window size is 2048, and the hop size is 1/1000 of the sampling frequency. To reduce impulsive noises, we use harmonic percussive source separation (HPSS) because impulsive noises are similar to percussive sounds.



(a) For the STFT and Mel models



(b) For the Chroma model

Fig. 1. Architecture of the CNN model. The numbers above the rectangles represent the shape of the data, and the numbers above the arrows represent the shape of the filter. Right-pointing arrows indicate the convolution layer and left-pointing arrows indicate the inverse convolution layer.

3 Experiment

We conducted an experiment to confirm whether an appropriate bass sound can be generated in several conditions.

3.1 Dataset

We made MIDI sequences that consisted of the guitar chord performances and bass lines using Cakewalk by BandLab. For simplicity, we only used four-bar chord progressions that consisted of one chord per measure. The guitar and bass performances are a sequence of eighth notes (for the bass, root eighth notes). Those MIDI sequences were converted to waveforms using software synthesizers (sforzando for the guitar and SI-Bass Guitar for the bass, included in Cakewalk by BandLab). The tempo for all sequences was set to 120 BPM. Based on these criteria, 20 pairs of guitar and bass signals were created. These pairs include those of the same chord progression but with different voicings. An example is shown in Fig.2. Out of them, 10 were allocated for training and 10 for testing.

3.2 Experimental conditions

The following three conditions were set.

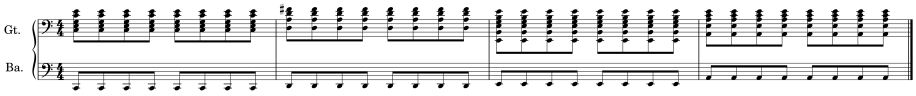


Fig. 2. Examples of guitar and bass scores created

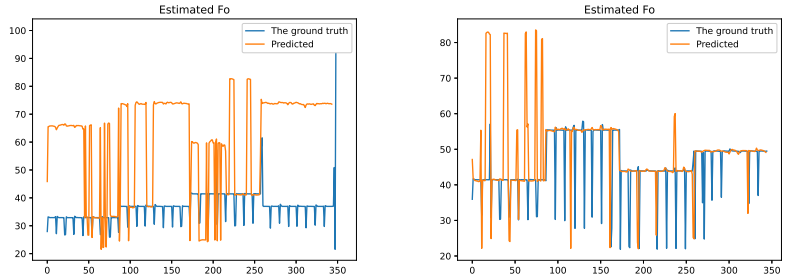


Fig. 3. Condition 1: F0 of generated bass lines with the Chroma model and the ground truth (Left: CDmEmD, the lowest accuracy; Right: EmAmFG, the highest accuracy)

Condition 1 The chord progressions or voicings are different between the training and test data, but all conditions in generating audio signals are the same.

Condition 2 In addition to Condition 1, the acoustic features are different between the training and test data. Specifically, a low-pass filter (setting: -3dB per octave increase) was applied to the test data.

Condition 3 The training data were those described above, while the test data was a recording of a performance by the first author on a real guitar. It was recorded using M-Audio's M-Track.

The generated bass signals were evaluated by calculating the ratio of *correct* frames. When the difference of the fundamental frequency (F0) at each frame from the signal given as the ground truth is lower than 50 cents, that frame is regarded as a *correct* frame. This ratio is called *accuracy* here. We also calculated *octave-ignored accuracy*, in which the difference of 1200 cents was considered correct.

3.3 Experimental results

The experimental results, listed in Table 1, can be summarized below[1].

Experimental condition 1 The model with the highest average accuracy, both with and without octave ignorance, was the Chroma model, and the model with the lowest average accuracy was the Mel model. When the octave is not ignored, the Mel model

[1] Audio samples are available at: `https://sites.google.com/kthrlab.jp/cmmr2023-kouzai`

Table 1. Accuracy and Octave-ignored accuracy for each test data

Condition	Filename	Accuracy			Octave-ignored accuracy		
		STFT	Mel	Chroma	STFT	Mel	Chroma
Condition1	A♯CDmEm_voicing	0.37	0.32	0.66	0.55	0.65	0.83
	EABmC♯m_voicing	0.39	0.20	0.64	0.48	0.29	0.67
	CDEmAm_voicing	0.39	0.49	0.53	0.53	0.57	0.81
	GABmD_voicing	0.55	0.54	0.62	0.62	0.56	0.77
	GCDEm	0.58	0.56	0.62	0.76	0.57	0.83
	CDmEmDm	0.42	0.21	0.17	0.66	0.26	0.77
	DmEmAmEm	0.57	0.40	0.32	0.65	0.40	0.84
	EmAmFG	0.59	0.39	0.81	0.69	0.42	0.87
	AmFGC	0.70	0.54	0.79	0.81	0.58	0.88
	FAmGDm	0.58	0.35	0.63	0.69	0.38	0.90
	Average	0.51	0.40	0.58	0.65	0.47	0.82
Condition2	A♯CDmEm_voicing	0.29	0.26	0.58	0.53	0.58	0.78
	EABmC♯m_voicing	0.22	0.26	0.62	0.27	0.41	0.75
	CDEmAm_voicing	0.17	0.30	0.51	0.29	0.43	0.82
	GABmD_voicing	0.28	0.49	0.67	0.34	0.51	0.74
	GCDEm	0.24	0.31	0.52	0.32	0.34	0.78
	CDmEmDm	0.15	0.11	0.23	0.20	0.10	0.85
	DmEmAmEm	0.23	0.17	0.22	0.27	0.18	0.85
	EmAmFG	0.35	0.26	0.68	0.43	0.28	0.88
	AmFGC	0.17	0.35	0.76	0.21	0.39	0.79
	FAmGDm	0.41	0.42	0.58	0.43	0.46	0.83
	Average	0.25	0.29	0.54	0.33	0.37	0.81
Condition3	CDEmAm_Audio	0.20	0.09	0.35	0.21	0.11	0.69

The name of the test data represents the chord progression. The same chord progression used for training, but with different voicing, was given "_voicing".

had the lowest accuracy among the three models in 6 out of the 10 data. When the octave is ignored, the Mel model had the lowest accuracy among the three models in 8 out of the 10 data.

Looking at Figure 3, which shows generated bass lines' F0 with the highest and lowest accuracy in the Chroma model, we can see that F0 is moving up and down. This is because the estimated F0s often contain double-pitch errors. In fact, the octave-ignored accuracy for these data is 0.77 and 0.6, respectively. For data containing three or more minor chords, the STFT model showed higher accuracy than the Chroma model, but again, the octave-ignored accuracy was high with the Chroma model.

Experimental condition 2 As in Condition 1, the model with the highest average accuracy with and without octave ignorance was the Chroma model. Especially for the Chroma model, the average accuracy was almost the same as for Condition 1. On the other hand, the model with the lowest average accuracy with and without octaves ignorance was the STFT model.

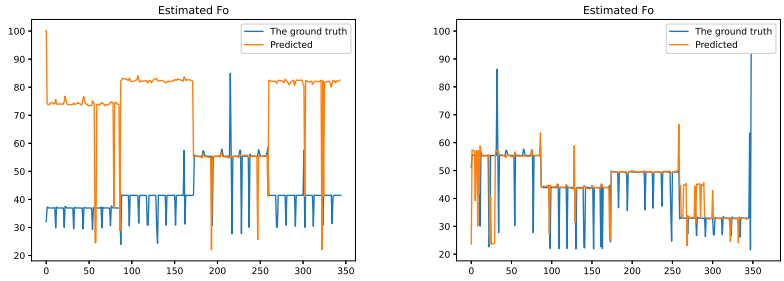


Fig. 4. Condition 2: F0 of generated bass lines with the Chroma model and the ground truth (Left: DmEmAmDm, the lowest accuracy; Right: AmFGC, the highest accuracy)

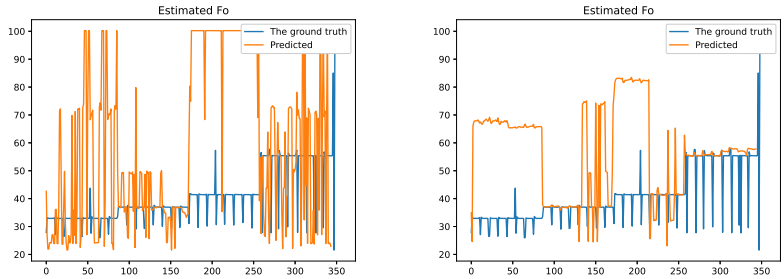


Fig. 5. Estimation of the fundamental frequency of CDEmAm_audio in the lowest accuracy Mel model and the highest accuracy Chroma model

Compared to Condition 1, the average accuracy for the STFT model dropped by more than 0.2, while the average accuracy for the Chroma model did not drop as much. This would be because the chromagram is a robust feature to timbral changes caused by the low-pass filter. Compared to Condition 1, the accuracy for data with many minor chords was lower for all models, especially for the STFT model; the accuracy dropped to less than half of the accuracy in Condition 1.

Figure 4 showed that the F0 fluctuates less up and down than in Condition 1. Instead, for DmEmAmEm, the double pitch was stably estimated. This is why the octave-ignored accuracy is high (0.85) while the accuracy is low (0.22).

Experimental condition 3 Although the accuracy was lower than in conditions 1 and 2, the model with the highest accuracy was the Chroma model, while the model with the lowest accuracy was the Mel model. The Mel model generated no harmonic tone in the first two measures. Because there were no harmonic tones, the F0 estimator showed erroneous values, as shown in Figure 5. This is why this model showed the lowest accuracy. With the Chroma model, bass-like harmonic tones were generated but were slightly distorted. This distortion caused double-pitch errors in F0 estimation; in fact, the accuracy and octave-ignored accuracy had a large difference.

4 Conclusion

In this paper, we proposed a method for generating bass signals from given guitar signals using a convolutional neural network. The experimental results show that the accuracy of the model using the chromagram is the best in all conditions, while the accuracy of the model using the mel-spectrogram and STFT is considerably low for guitar signals with a low-pass filter.

However, these models have been tested only with simple bass lines that consist of only root notes. To enable to generate more complex bass lines, the models need to learn various bass lines, ranging from rhythmic to melodious ones, played in real songs. To achieve this, we must consider longer contexts in the models. Therefore, we would like to extend our models, for example by increasing context layers.

References

1. S. Onuma, M. Hamanaka.: Piano Arrangement System Based on Composers' Arrangement Processes, ICMC, pp.191–194 (2010)
2. E. Nakamura, S. Sagayama.: Automatic Piano Reduction from Ensemble Scores Based on Merged-Output Hidden Markov Model, ICMC, pp.298–305 (2015)
3. M. Terao, Y. Hiramatsu, R. Ishizuka, Y. Wu, K. Yoshii.: Difficulty-Aware Neural Band-To-Piano Score Arrangement Based on Note-and Statistic-Level Criteria, ICASSP, pp.196–200 (2022)
4. S. Ariga, S. Fukayama, M. Goto.: Song2Guitar: A Difficulty-aware Arrangement System for Generating Guitar Solo Covers from Polyphonic Audio of Popular Music, ISMIR, pp568–574 (2017)
5. Daniel R. Tuohy, Walter D. Potter.: GA-based Music Arranging for Guitar, Congress on Evolutionary Computation, pp.1065–1070 (2006)
6. A. Ferraro, D. Bogdanov, X. Serra, Jay H. Jeon, J. Yoon.: How Low Can You Go? Reducing Frequency and Time Resolution in Current CNN Architectures for Music Auto-tagging, EUSIPCO2020 (2020)
7. M. Dong.: Convolutional Neural Network Achieves Human-level Accuracy in Music Genre Classification, arXiv:1802.09697 (2018)
8. Y. Gong, S. Khurana, A. Rouditchenko, J. Glass.: CMKD: CNN/Transformer-Based Cross-Model Knowledge Distillation for Audio Classification, arXiv:2203.06760 (2022)
9. G. Jiang, A. Biswas, C. Bergler, A. Maier.: InSE-NET: A Perceptually Coded Audio Quality Model based on CNN, AES (2021)
10. C. Ji, Y. Pan.: Infant Vocal Tract Development Analysis and Diagnosis by Cry Signals with CNN Age Classification, SpeD, pp.37–41 (2021)
11. Ephrem A. Retta, R. Sutcliffe, E. Almekhlaf, Yosef K. Enku, E. Alemu, Tigist D. Gemechu, Michael A. Berwo, M. Mhamed, J. Feng.: Kinit Classification in Ethiopian Chants, Azmaris and Modern Music: A New Dataset and CNN Benchmark, arXiv:2201.08448 (2022)