deepGTTM-III: Multi-task Learning with Grouping and Metrical Structures

Masatoshi Hamanaka[1], Keiji Hirata[2], and Satoshi Tojo[3]

[1] RIKEN
masatoshi.hamanaka@riken.jp
[2] Future University Hakodate
hirata@fun.ac.jp
[3] JAIST
tojo@jaist.ac.jp

Abstract. This paper describes an analyzer that simultaneously learns grouping and metrical structures on the basis of the generative theory of tonal music (GTTM) by using a deep learning technique. GTTM is composed of four modules that are in series. GTTM has a feedback loop in which the former module uses the result of the latter module. However, as each module has been independent in previous GTTM analyzers, they did not form a feedback loop. For example, deepGTTM-I and deepGTTM-II independently learn grouping and metrical structures by using a deep learning technique. In light of this, we present deepGTTM-III, which is a new analyzer that includes the concept of feedback that enables simultaneous learning of grouping and metrical structures by integrating both deepGTTM-I and deepGTTM-II networks. The experimental results revealed that deepGTTM-III outperformed deepGTTM-I and had similar performance to deepGTTM-II.

Keywords: A generative theory of tonal music (GTTM), grouping structure, metrical structure, deep learning

1 Introduction

Our main goal was to develop a system that enabled a time-span tree of a melody to be automatically acquired on the basis of the generative theory of tonal music (GTTM) [1]. GTTM is composed of four modules, each of which assigns a separate structural description to a listener's understanding of a piece of music. These four modules sequentially output a grouping structure, metrical structure, time-span tree, and prolongational tree. The grouping structure is intended to formalize the intuitive belief that tonal music is organized into groups that are in turn composed of subgroups. These groups are presented graphically as several levels of arcs below a music staff. The metrical structure describes the rhythmical hierarchy of a piece of music by identifying the position of strong beats at the levels of a quarter note, half note, measure, two measures, four measures, and so on. Strong beats are illustrated as several levels of dots below the music staff (Fig. 1).

The time-span tree provides performance rendering [2], music reproduction [3], and a summarization of the music [4]. This summarization can be used as a representation of a search, resulting in music retrieval systems. It can also be used for melody morphing, which generates an intermediate melody between two melodies in systematic order [5, 6]. These systems presently need a time-span tree analyzed by musicologists because previous analyzers [7, 8] have not performed optimally.

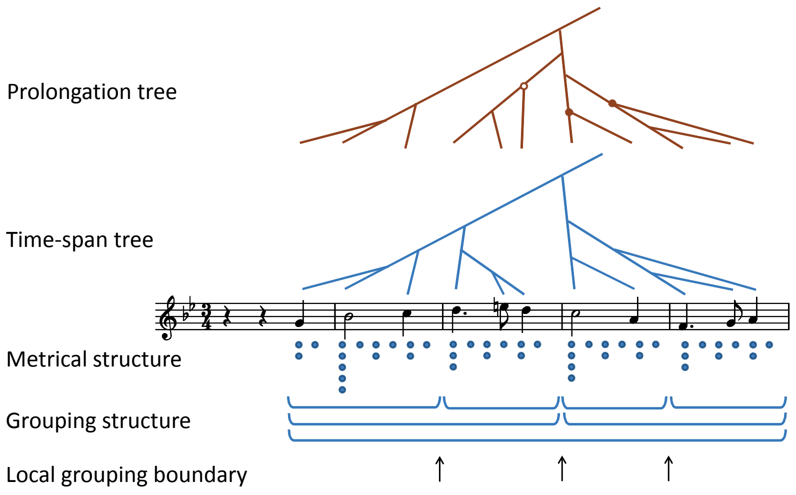

Fig. 1. Grouping structure, metrical structure, time-span tree, and prolongational tree.

There are three significant problems when implementing GTTM on a computer.

- **Conflict between rules.**
  There are two types of rules in GTTM: well-formedness rules (WFRs) and preference rules (PRs). WFRs are necessary conditions to assign a structure and restrictions to these structures. When more than one structure can satisfy the WFRs, PRs indicate the superiority of one structure over another.
  Because there is no strict order of applying PRs, a conflict between rules often occurs when applying them, which results in ambiguities in analysis. Figure 2 outlines an example of the conflict between metrical preference rules (MPRs) 5c and 5a. The MPR5c states that a relatively long slur results in a strong beat, and MPR5a states that a relatively long pitch-event results in a strong beat. Because metrical WRF 3 (MWFR3) states that strong beats are spaced either two or three beats apart, a strong beat cannot be perceived at both onsets of the first and second notes.

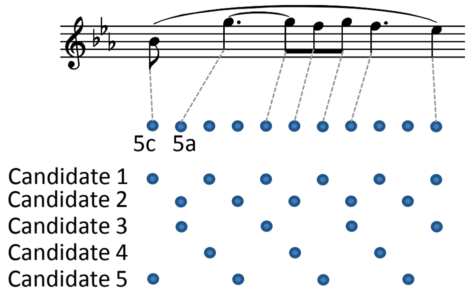

Fig. 2. Conflict between two metrical preference rules

We developed an automatic time-span tree analyzer (ATTA) [7] that had 46 adjusted parameters to control the strength of each rule. In other words, the ATTA we developed enabled us to control the priority of rules, which enabled us to obtain extremely accurate groupings and metrical structures. However, we needed musical knowledge like that of musicologists to properly tune the parameters.

The full ATTA (FATTA) [8] does not have to tune the parameters because it automatically calculates the stability of structures and optimizes the parameters so that the structures are stable. FATTA obtains excellent analysis results for metrical structures but unacceptable results for grouping structures and time-span trees.

The deep layered network in our deepGTTM enables us to learn the priority of rules.

- **Difficult to integrate bottom up and top down processes.**
GTTM rules include bottom up and top down rules. For example, GPR2 is a bottom up rule that prescribes the relationship of onset (attack) and offset (release) timings, and a grouping boundary. In contrast, GPR5 is a top down rule that prefers that a group be divided into two subgroups of the same length.

The ATTA and FATTA frequently output incorrect higher level hierarchical structures even when a low-level structure is correct because they only use a bottom up process.

In contrast, we also developed analyzers that only use a top down process called σGTTM [9], σGTTMII [10], and σGTTMIII [11]. σGTTM and σGTTMII can detect the local grouping structure in GTTM analysis by combining the GTTM with statistical learning. However, σGTTM and σGTTMII are only suitable for grouping structures and cannot acquire time-span trees. σGTTMIII enabled us to automatically analyze time-span trees by learning with a time-span tree of 300 pieces from the GTTM database [12] on the basis of probabilistic context-free grammar (PCFG). σGTTMIII performed the best at acquiring time-span trees. However, these analyzers [7–11] do not perform sufficiently well for use in application systems [2–6].

The deep layered network in our deepGTTM learns both top down and bottom up rules from learning data.

- **Feedback loops.**
  The four modules in the GTTM are in series, i.e., the latter module uses the result from the former module. The GTTM also has a feedback loop in which the former module uses the result from the latter module. For example, GPR7 (time-span and prolongational stability) prefers a grouping structure that results in a more stable time-span and/or prolongation reduction. Another example is that MPR9 (time-span interaction) prefers metrical analysis that minimizes conflict in the time-span reduction. However, as each module has been independent in previous GTTM analyzers, they did not form a feedback loop. For example, deepGTTM-I [13] and deepGTTM-II [14] independently learned the grouping and metrical structures by using a deep learning technique.

  Figure 3 summarizes the theory reported by Lerdahl and Jakendoff [1]. The bottom right has the preference rules. If we naïvely implement this theory, the analysis process is endless looping when the output is divergence.

In light of this, we present deepGTTM-III, which is a new analyzer that solves these problems and enables simultaneous learning of grouping and metrical structures by integrating both networks of deepGTTM-I and deepGTTM-II. The deep layered network learns the priority of rules that solves the conflict between rules. The network also learns both bottom up and top down rules. By integrating both deepGTTM-I and deepGTTM-II networks, the integrated network possesses information to acquire the metrical structure and the grouping structure. Therefore, the information on acquiring the metrical structure can be used for acquiring the grouping structure, and vice versa. Therefore, a feedback loop is implicitly constructed inside the network.

The network was pre-trained by using 15,000 pieces of music formatted in musicXML that were acquired by Web crawling. We used 300 pieces from the GTTM database: 200 for fine-tuning and 100 for evaluation [12]. The experimental results indicated that the integrated network outperformed the independent network.

The paper is organized as follows. Section 2 describes related work, and Section 3 explains our GTTM analyzers: deepGTTM-I, II, and III. Section 4 explains how we evaluated the performance of deepGTTM-I, II, and III, and Section 5 concludes with a summary and an overview of future work.

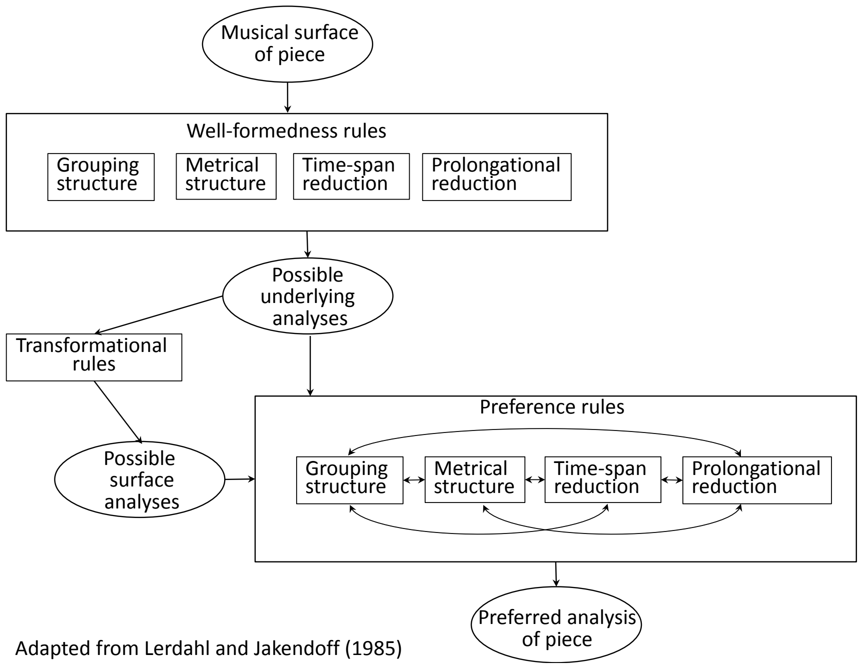

Fig. 3. Summary of Lerdahl and Jakendoff's theory.

2 Related Work

Deep learning has recently been used for tasks in the area of music information retrieval [15–19] and has demonstrated its potential to solve various kinds of tasks in the area. An automatic tagging system using a fully convolutional network was developed that predicts high-level information about a music clip, such as emotion, genre, and instrumentation [15]. An automatic chord detection system using the bottleneck architecture of a deep layered network outperformed previous systems based on support vector machines (SVMs) and hidden Markov models (HMMs) [16]. An automatic system of chord estimation based on a hybrid Gaussian HMM and deep learning approach enabled very large chord progressions to be estimated [17]. A music recommendation system using deep convolutional neural networks to predict latent factors from music audio demonstrated that deep convolutional neural networks significantly outperformed more traditional approaches [18]. Automatic polyphonic music transcription using a supervised neural network model performed the best across the two most common unsupervised acoustic models [19]. These systems [15–19] replaced other machine learning techniques with deep learning and performed better than traditional machine learning techniques.

The traditional machine learning techniques cannot work well in our task of acquiring hierarchical musical structures. In other words, only deep learning enables the relationship between input scores and output structures to be learned. Direct learning between inputs to output does not work well because they have gaps that are too wide. Therefore, we prepared two steps for learning. First, the network learns individual rule applications. The deep layered network can easily learn rules in GTTM. After the rule applications are learned, the network can learn the relationship between input scores and output structures. That is, the network gains musical knowledge by learning GTTM rules.

3 deepGTTM-I, II, and III

deepGTTM-I, II, and III are GTTM analyzers based on deep learning. deepGTTM-I analyzes the local grouping boundaries of a grouping structure [13], and deepGTTM-II analyzes the metrical structure [14]. This paper presents deepGTTM-III, which integrates deepGTTM-I and II.

There are three main advantages of using deep learning for GTTM analysis.

- **Learning applications of both bottom up and top down rules**
  Previous analysis systems based on GTTM were constructed by human researchers or programmers. Some rules in GTTM are very ambiguous, and their implementations might differ depending on the person. However, deepGTTM is a learning based system where the quality of the analyzer depends on the training data and trained network. The input of the network includes the score information of the whole analysis area to learn both bottom up and top down rules.

- **Learning priority of rules**
  σGTTM and σGTTMII do not work well because they only determine the priority of rules from applied rules because the priority of rules depends on the context of a piece. The input of the network in deepGTTM, on the other hand, is the score, and the network learns the priority of the rules as the weight and bias of the network on the basis of the context of the score.

- **Feedback loop in deep layered network**
  There are several kinds of feedback processes in deepGTTM-III because a deep layered network is trained by multi-task learning with grouping and metrical structures. When the grouping structure is learned, important information for acquiring grouping and metrical structures is propagated because deepGTTM-III shares hidden nodes for acquiring both grouping and metrical structures. Similarly, when the metrical structure is learned, important information on acquiring the grouping structure is also propagated.

3.1 Structure of Network

We used a deep belief network (DBN) [20] for deepGTTM-I, II, and III. Figure 4 outlines the structure for the DBN of deepGTTM-I. The input of the DBN was

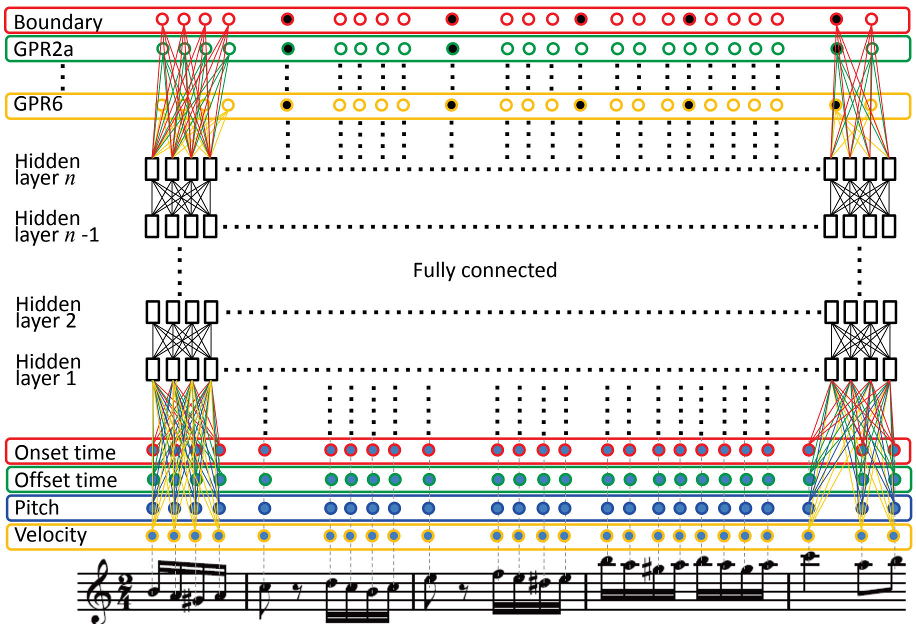

Fig. 4. DBN for deepGTTM-I.

the onset time, offset time, pitch, and velocity of note sequences from musicXML. All inputs were normalized from zero to one. The output of DBN formed multi-tasking learning, which had 10 outputs: 9 kinds of grouping preference rules (GPR2a, 2b, 3a, 3b, 3c, 4, 5, 6, and 7) and local grouping boundary.

Figure 5 outlines the structure for deepGTTM-II. The inputs of deepGTTM-II are the onset time, offset time, pitch, velocity, and the grouping structure manually analyzed by musicologists. Each hierarchical level of the grouping structure is separately input by a note neighboring the grouping boundary as one; otherwise, it is zero. There are eight outputs of deepGTTM-II that enable multi-task learning in each hierarchical level of the metrical structure, i.e., seven MPRs (MPR2, 3, 4, 5a, 5b, 5c, and 5d), and one level of the metrical structure. Individual outputs have two units, e.g., rules that are not applicable (=0) and rules that are applicable (=1), or weak beats (=0) and strong beats (=1). A metrical structure consists of hierarchical levels, and we added one hidden layer to generate the next structure level. We used logistic regression to connect the final hidden layer (n,n+1,..., n+h) and outputs. All outputs shared the hidden layers from one to the final hidden layer.

Figure 6 outlines the structure for the DBN we called deepGTTM-III to generate a grouping and metrical structure. deepGTTM-III has the same input as deepGTTM-I, and its output is the same as the merged output of deepGTTM-I and II.

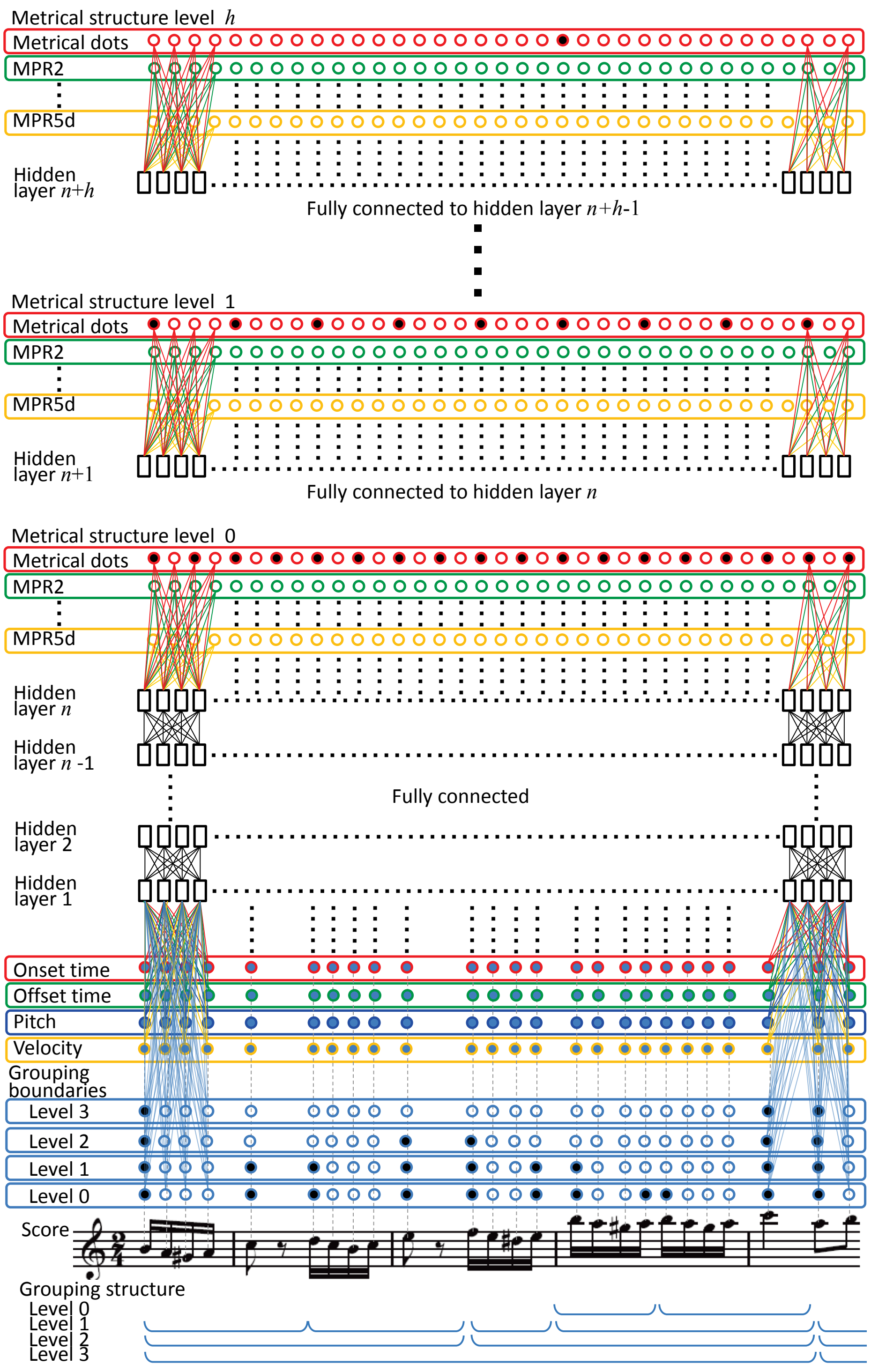

Fig. 5. DBN for deepGTTM-II.

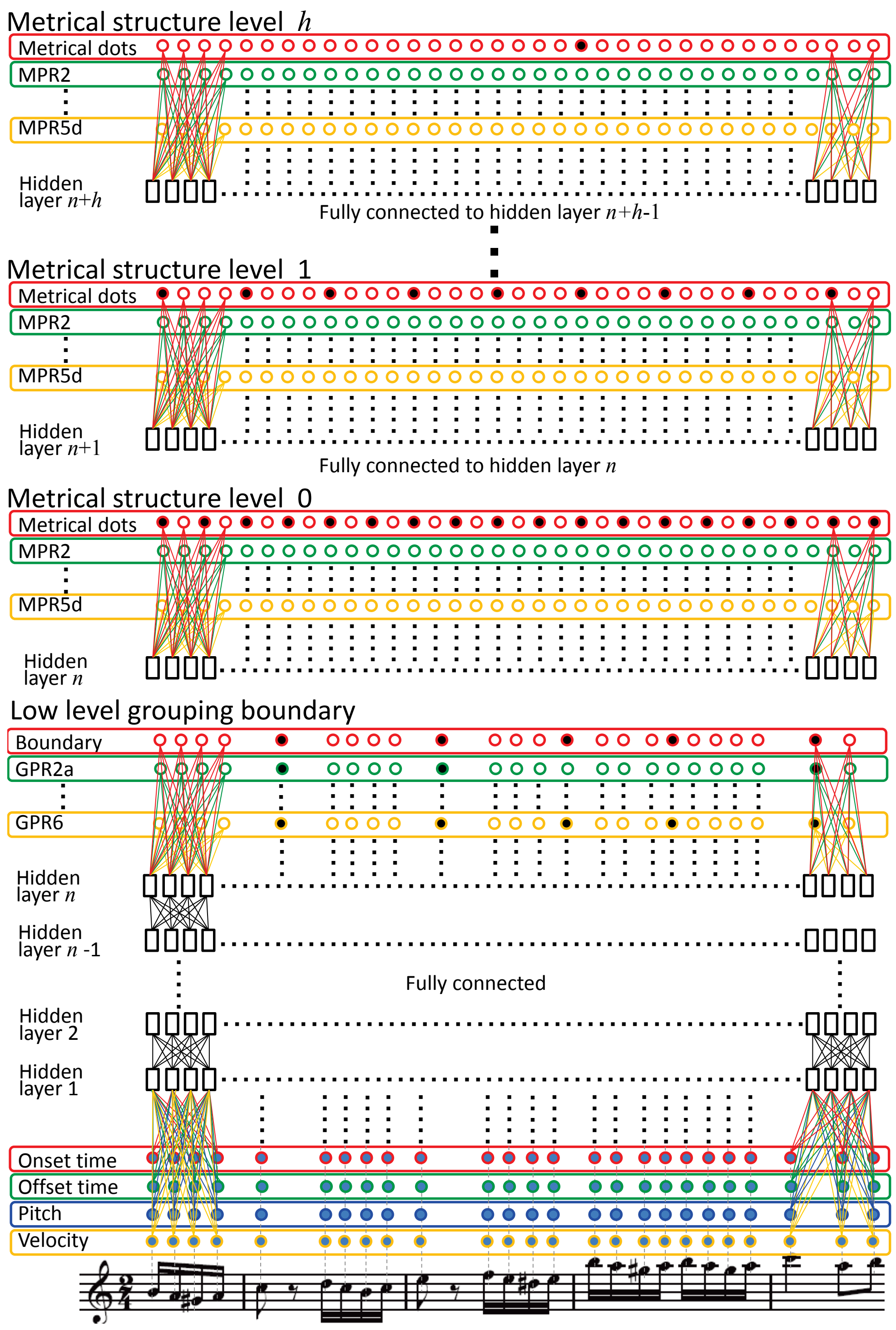

Fig. 6. DBN for deepGTTM-III.

3.2 Learning Networks

This section describes how we learned the local grouping boundaries and metrical structure by using deep layered networks.

Pre-training. The network learned the features of the music in pre-training. As a large scale dataset with no labels was needed, we collected 15,000 pieces of music formatted in musicXML from Web pages that were linked to the musicXML page of MakeMusic Inc. [21]. The musicXMLs were downloaded in three steps.

1). A Web autopilot script made a list of URLs that were most likely files of musicXMLs from five links on the musicXML page of MakeMusic Inc.
2). The files in the URL list were downloaded after URLs that were clearly not musicXMLs had been omitted.
3). All the downloaded files were opened using the script, and files that were not musicXML were deleted.

Each network of deepGTTM-I, II, and III was pre-trained by using a restricted Boltzmann machine.

Learning rules application and structure. The network in fine-tuning learned with a labeled dataset. We had 300 pieces with a labeled dataset in the GTTM database, which included musicXMLs with positions of local grouping boundaries, positions of dots for each hierarchy of the metrical structure, and positions to which the grouping and metrical preference rules were applied. However, these 300 pieces were insufficient for deep learning.

Consequently, we constructed a half-labeled dataset. We automatically added the labels of six applied rules of GPR2a, 2b, 3a, 3b, 3c, and 3d, and MPR3, 5a, 5b, 5c, 5d because these rules could be uniquely applied as a score. We used our ATTA to add labels to these rules.

We also artificially increased the labeled dataset because the 300 pieces in the GTTM database were insufficient for training a deep layered network. First, we transposed the pieces for all 12 keys. We then changed the length of note values to two times, four times, eight times, a half time, a quarter time, and an eighth time. Thus, the total labeled dataset had 25, 200 (= 300x12x7) pieces.

The priority of rules and grouping and metrical structures were learned by back propagation of the deep layered network using the half-labeled dataset and labeled dataset. deepGTTM-I and II had very complex networks. The fine-tuning of local grouping boundaries and one level of the metrical structure involved multi-task learning. The fine-tuning of each PR also involved multi-task learning. Therefore, the fine-tuning of PRs involved multi-dimensional multi-task learning. The processing flow for the learning of a GPR or local grouping boundaries had four steps. The order of music pieces was changed at every epoch in all steps.

1). The order of the pieces of training data was randomly shuffled, and a piece was selected from top to bottom.
2). The note transition of the selected piece was randomly shuffled and a note transition was selected from top to bottom.

3). Back propagation from output to input was carried out on the basis of whether the note transition had a boundary, or the rule was applied (=1) or not (=0).
4). The next note transition or the next piece in steps 1 and 2 was repeated.

The processing flow for the learning of an MPR or metrical dots involved four steps.

1). The order of the music pieces of training data was randomly shuffled, and a piece was selected from top to bottom.
2). The beat positions of the selected piece were randomly shuffled, and a beat position was selected from top to bottom.
3). Back propagation from output to input was carried out on the basis of whether the beat position had a strong beat, or the rule was applied (=1) or not (=0).
4). The next piece in step 1 or the next beat position in step 2 was repeated.

The processing flow for the multidimensional multi-task learning of PRs involved three steps.

1). The order of PRs was randomly shuffled, and a rule was selected from top to bottom.
2). Multi-task learning of the selected PR was carried out.
3). The next rules in step 1 were repeated.

Simultaneous learning of grouping and metrical structures in deepGTTM-III. The deep layered network of deepGTTM-III was trained by using a multi-task learning technique for the grouping and metrical structures. The main difference in learning and acquiring the metrical structure in deepGTTM-II and deepGTTM-III is that the grouping structure is needed in the input data for deepGTTM-II but not for deepGTTM-III. In other words, deepGTTM-III predicts low level grouping boundaries by itself and uses the information on predicting the low level grouping boundaries to predict the metrical structure.

The deepGTTM-I and II networks first learn individual rule applications by fixed numbers of epochs and then learn the structure on the same epochs. They then repeat learning of rule applications and structure learning. In contrast, deepGTTM-III repeats learning of GPR applications, grouping structures, MPR applications, and metrical structures by using fixed numbers of epochs. As all learning processes of grouping and metrical structures interact in the deepGTTM-III network, the feedback loop described in Section 3 is implicitly formed.

A GPR and an MPR are sometimes learned complementarily when the rules are learned because some GPRs and MPRs are very similar. For example, GPR6 (parallelism) prefers to form parallel parts of a group, where two or more segments of the music can be construed as parallel, and MPR1 (parallelism) prefers a parallel metrical structure, where two of more groups or parts of groups can be construed as parallel.

In another example, consider a sequence of four notes: n1, n2, n3, and n4. GPR2b (Attack-Point) states the transition n2-n3 may be heard as a group boundary if the interval between the attack points of n2 and n3 is greater than that between the attack points of n1 and n2 and that between the attack points of n3 and n4. MPR5a prefers a metrical structure in which a relatively strong beat occurs at the inception of a relatively long pitch event.

4 Experimental Results

We evaluated deepGTTM by using 100 music pieces from the GTTM database; the remaining 200 pieces were used to train the network. The F-measure was given by the weighted harmonic mean of precision P (proportion of selected dots that were correct) and recall R (proportion of correct dots that were identified).

Table 1 compares the results for deepGTTM-III with those for deepGTTM-I and deepGTTM-II for a network that had 11 layers with 3000 units. The results indicate that deepGTTM-III obtained a higher F-measure in acquiring local grouping boundaries than deepGTTM-I. However, deepGTTM-III obtained an F-measure in acquiring a metrical structure similar to that of deepGTTM-II, which was slightly higher than that of deepGTTM-III. We used the correct grouping structure in the GTTM database because deepGTTM-II needs a grouping structure for input to the network. In contrast, as deepGTTM-III does not need a grouping structure, it operates efficiently even when there is no correct grouping structure.

Table 1. Performance of deepGTTM-I, II, and III.

	Low level grouping boundary		Metrical structure	
Melodies	deepGTTM-III	deepGTTM-I	deepGTTM-III	deepGTTM-II
1. Grande Valse Brillante	0.80	0.79	0.93	0.94
2. Moments Musicaux	0.80	0.81	0.99	1.00
3. Turkish March	0.77	0.76	0.96	0.98
4. Anitras Tanz	0.78	0.76	0.90	0.90
5. Valse du Petit Chien	0.80	0.78	0.99	0.99
	:	:	:	:
Total (100 melodies)	0.81	0.78	0.94	0.96

5 Conclusion

We presented deepGTTM-III, which integrates a grouping structure analyzer called deepGTTM-I and a metrical structure analyzer called deepGTTM-II. Whereas deepGTTM-I and deepGTTM-II have to independently learn grouping and metrical structures, deepGTTM-III learns them simultaneously. The experimental results indicated that deepGTTM-III obtained a higher F-measure in acquiring local grouping boundaries than deepGTTM-I and had a similar F-measure to deepGTTM-II in acquiring the metrical structure. This work was one step in implementing a generative theory of tonal music (GTTM) based on deep learning. We plan to implement time-span reduction analysis on the basis of deep learning in the future.

Acknowledgments. This work was supported by JSPS KAKENHI Grant Numbers 17H01847, 25700036, 16H01744, and 23500145.

References

1. Lerdahl, F. and Jackendoff, R.: A Generative Theory of Tonal Music. MIT Press (1985).
2. Hirata, K. and Hiraga R.: Ha-Hi-Hun plays Chopin's Etude. In Working Notes of IJCAI-03 Workshop on methods for automatic music performance and their applications in a public rendering contest (2003).
3. Hirata, K., Matsuda, S., Kaji K., and Nagao K.: Annotated Music for Retrieval, Reproduction, and Sharing. In: Proceedings of the 2004 International Computer Music Conference (ICMC2004), pp. 584–587 (2004).
4. Hirata K. and Matsuda S.: Interactive Music Summarization based on GTTM. In: Proceeding of the 2002 International Society for Music Information Retrieval Conference (ISMIR2002), pp. 86–93 (2002).
5. Hamanaka, M., Hirata, K., and Tojo, S.: Melody Morphing Method based on GTTM. In: Proceedings of the 2008 International Computer Music Conference (ICMC2008), pp. 155–158 (2008).
6. Hamanaka, M., Hirata, K., and Tojo, S.: Melody Extrapolation in GTTM Approach. In: Proceedings of the 2009 International Computer Music Conference (ICMC2009), pp. 89–92 (2009).
7. Hamanaka, M., Hirata, K., and Tojo, S.: Implementing 'a generative theory of tonal music'. Journal of New Music Research, 35(4), 249–277 (2006).
8. Hamanaka, M., Hirata, K., and Tojo, S.: FATTA: Full automatic time-span tree analyzer. In: Proceedings of the 2007 International Computer Music Conference (ICMC2007), pp. 153–156 (2007).
9. Miura, Y., Hamanaka, M., Hirata, K., and Tojo, S.: Decision tree to detect GTTM group boundaries. In: Proceedings of the 2009 International Computer Music Conference (ICMC2009), pp. 125–128 (2009).
10. Kanamori, K. and Hamanaka, M.: Method to Detect GTTM Local Grouping Boundaries based on Clustering and Statistical Learning. In: Proceedings of the 2014 International Computer Music Conference (ICMC2014), pp. 125–128 (2014).
11. Hamanaka, M., Hirata, K., and Tojo, S.: *sigma*GTTM III: Learning-based Time-span Tree Generator Based on PCFG. In: Proceedings of the 11th International Symposium on Computer Music Multidisciplinary Research (CMMR 2015), pp. 303–317 (2015).
12. Hamanaka, M., Hirata, K., and Tojo, S.: Musical Structural Analysis Database Based on GTTM. In: Proceeding of the 2014 International Society for Music Information Retrieval Conference (ISMIR2014), pp. 325–330 (2014).

13. Hamanaka, M., Hirata, K., and Tojo, S.: deepGTTM-I: Local Boundary Analyzer based on a Deep Learning Technique. In: Proceedings of the 12th International Symposium on Computer Music Multidisciplinary Research (CMMR 2016), pp. 8–20 (2016).
14. Hamanaka, M., Hirata, K., and Tojo, S.: deepGTTM-II: Automatic Generation of Metrical Structure based on Deep Learning Technique. In: Proceedings of 13th Sound and Music Computing Conference (SMC2016), pp. 203–210 (2016).
15. Choi, K., Fazekas, G., and Sandler, M.: Automatic Tagging Using Deep Convolutional Neural Networks. In: Proceeding of the 2016 International Society for Music Information Retrieval Conference (ISMIR2016), pp. 805–811 (2016).
16. Zhou, X. and Lerch, A.: Chord Detection Using Deep Learning. In: Proceeding of the 2015 International Society for Music Information Retrieval Conference (ISMIR2015), pp. 52–58 (2015).
17. Deng, J. and Kwok, Y.: Hybrid Gaussian-HMM-Deep Learning Approach for Automatic Chord Estimation with Very Large Vocabulary, In: Proceeding of the 2016 International Society for Music Information Retrieval Conference (ISMIR2016), pp. 812–818 (2016).
18. Oord, A., Sander, D., and Benjamin, S.: Deep content-based music recommendation. In: Proceeding of the Advances in Neural Information Processing Systems 26 (NIPS 2013), pp. 2643–2651 (2013).
19. Sigtia, S., Benetos, E., and Dixon, S.: An End-to-End Neural Network for Polyphonic Piano Music Transcription, IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), 24 (5), 927–939 (2016).
20. Hinton, G. E., Osindero, S., and Teh, Y. W.: A fast learning algorithm for deep belief nets. Neural Comp. 18, pp. 1527–1554 (2006).
21. MakeMusic Inc., "Finale," 2018, http://www.finalemusic.com/.