DeepGTTM-II: Automatic Generation of Metrical Structure based on Deep Learning Technique

Masatoshi Hamanaka
Kyoto University
hamanaka@kuhp.kyoto-u.ac.jp

Keiji Hirata
Future University Hakodate
hirata@fun.ac.jp

Satoshi Tojo
JAIST
tojo@jaist.ac.jp

ABSTRACT

This paper describes an analyzer that automatically generates the metrical structure of a generative theory of tonal music (GTTM). Although a fully automatic time-span tree analyzer has been developed, musicologists have to correct the errors in the metrical structure. In light of this, we use a deep learning technique for generating the metrical structure of a GTTM. Because we only have 300 pieces of music with the metrical structure analyzed by musicologist, directly learning the relationship between the score and metrical structure is difficult due to the lack of training data. To solve this problem, we propose a multidimensional multitask learning analyzer called deepGTM-II that can learn the relationship between score and metrical structures in the following three steps. First, we conduct unsupervised pre-training of a network using 15,000 pieces in a non-labeled dataset. After pre-training, the network involves supervised fine-tuning by back propagation from output to input layers using a half-labeled dataset, which consists of 15,000 pieces labeled with an automatic analyzer that we previously constructed. Finally, the network involves supervised fine-tuning using a labeled dataset. The experimental results demonstrated that the deepGTTM-II outperformed the previous analyzers for a GTTM in F-measure for generating the metrical structure.

1. INTRODUCTION

We propose an analyzer for automatically generating a metrical structure based on a generative theory of tonal music (GTTM) [1]. A 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 output a grouping structure, metrical structure, time-span tree, and prolongational tree. As the acquisition of a metrical structure is the second step in the GTTM analysis, an extremely accurate analyzer makes it possible to improve the performance of all later analyzers.

We previously constructed several analyzers that enabled us to acquire a metrical structure such as the automatic time-span tree analyzer (ATTA) [2] and fully automatic time-span tree analyzer (FATTA) [3]. However, the performance of these analyzers was inadequate in that musicologists had to correct the boundaries because of numerous errors.

For this paper, we propose the deepGTTM-II with which we use deep learning [4] to improve the performance of generating a metrical structure from a score. Unsupervised training in the deep learning of deep-layered networks called pre-training aids in supervised training, which is called fine-tuning [5].

Our goal was to develop a GTTM analyzer that enables us to output the results obtained from analysis that are the same as those obtained by musicologists based on deep learning by learning the analysis results obtained by musicologists. We had to consider three issues in constructing such a GTTM analyzer.

- **Multi-task learning**
  A model or network in a simple learning task estimates the label from an input feature vector. However, metrical strength of each beat can be found in every beats. Therefore, we consider a single learning task as estimating whether one beat can be a strong beat or weak beat. Then, a problem in detecting a metrical structure can be solved by using multi-task learning.

  Subsection 4.3 explains multi-task learning by using deep learning.

- **Hierarchical metrical structure**
  A hierarchical metrical structure is generated by iterating the choice of the next level structure. The next level structure is recursively generated using the previous structure. However, when we use learning with a single standard network of deep learning, it is difficult to lean a higher level structure because many network representations are used for learning a lower level structure

  Subsection 4.2 explains how to learn a higher level structure.

- **Large scale training data**
  Large-scale training data are needed to train a deep-layered network. Labels are not needed for pre-training the network. Therefore, we collected 15,000 pieces of music formatted in musicXML from Web pages that were introduced in the MusicXML page of MakeMusic Inc [6]. We needed labeled data to fine-tune the network. Although we had 300 pieces

with labels in the GTTM database [7], this number was too small to enable the network to learn.

Subsection 4.1 explains how we collected the data and how we got the network to learn effectively with a small dataset.

- **GTTM rules**
  A GTTM consists of several rules, and a beat that is applied to many rules tends to be strong in metrical structure analysis. As a result of analysis by musicologists, 300 pieces in the GTTM database were not only labeled with the correct metrical structure but also labeled with applied positions of metrical preference rules. Therefore, the applied positions of metrical preference rules were helpful clues for estimating whether one beat can be strong or weak.

  Subsection 4.2 explains how the network learned with the metrical preference rules.

- **Sequential vs. recurrent models**
  There are two types of models, i.e., recurrent and sequential, that can be used for analyzing a hierarchical metrical structure. The recurrent neural network provides recurrent models, which are suitable for analyzing a metrical structure in which cyclical change results in strong and a weak beats. However, the recurrent neural network is difficult to train and training time is very long. On the other hand, sequential models, such as deep belief networks (DBN), are not suitable for detecting the repetition of strong beats. However, the DBN is very simple and performs well in detecting the local grouping boundary of the GTTM in deepGTTM-I.

  Therefore, we choose the DBN for analyzing the metrical structure of a piece. Subsection 4.2 explains how the DBN is trained for analyzing metrical structure.

The results obtained from an experiment suggest that our multi-dimensional multi-task learning analyzer using deep learning outperforms the previous GTTM analyzers in obtaining the metrical structure.

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

2. RELATED WORK

We briefly look back through the history of cognitive music theory. The imprecation realization model (IRM) proposed by Narmour abstracts and expresses music according to symbol sequences from information from a score [8, 9]. Recently, the IRM has been implemented on computers and its chain structures can be obtained from a score [10]. On the other hand, the *Schenkerian* analysis analyzes deeper structures called “Urline” and “Ursatz” from the music surface [11]. Short segments of music can be analyzed through Schenkerian analysis on a computer [12]. There is another approach that constructs a music theory for adopting computer implementation [13, 14].

The main advantage of analysis by a GTTM is that it can acquire tree structures called time-span and prolongation trees. A time-span or prolongation tree provides a summarization of a piece of music, which can be used as the representation of an abstraction, resulting in a music retrieval system [15]. It can also be used for performance rendering [16] and reproducing music [17]. The time-span tree can also be used for melody prediction [18] and melody morphing [19].

The metrical structure analysis in a GTTM is a kind of beat tracking. Current methods based on beat tracking [20–23] can only acquire the hierarchical metrical structure in a measure because they do not take into account larger metrical structures such as two and four measures.

Our ATTA [2] by integrating a grouping structure analyzer and metrical analyzer. The metrical structure analyzer has 18 adjustable parameters. It enables us to control the priority of rules, which enables us to obtain extremely accurate metrical structures. However, we need musical knowledge like that which musicologists have to properly tune the parameters.

Our FATTA [3] does not have to tune parameters because it automatically calculates the stability of structures and optimizes the parameters to stabilize the structures. However, its performance for generating a metrical structure is lower than that of the ATTA.

The σGTTM [24] enables us to automatically detect local grouping boundaries by using a decision tree. The σGTTMII [25] involves clustering steps for learning the decision tree and outperforms the ATTA if we can manually select the best decision tree. The σGTTMIII [26] enables us to automatically analyze time-span trees by learning with a time-span tree of 300 pieces of music from the GTTM database [7] based on probabilistic context-free grammar (PCFG). The pGTTM [27] also uses PCFG, and we used it to attempt unsupervised learning. The main advantages of σGTTMIII and pGTTM are that they can learn the context in difference hierarchies of the structures (e.g., beats are important in the leaves of time-span trees, or chords are important near the roots of the trees.). However, none of these analyzers [7, 24, 25, 27] can generate the metrical structure.

On the other hand, our deepGTTM-I [28] outperforms the ATTA, FATTA, σGTTM, and σGTTMII in detecting local grouping boundaries by introducing deep learning for GTTM analysis. However, it also cannot acquire the hierarchical grouping structure.

In light of this, we introduce a deep learning analyzer for generating the hierarchical metrical structure of a GTTM.

3. GTTM AND ITS IMPLEMENTATION PROBLEMS

Figure 1 Shows a grouping structure, metrical structure, time-span tree, and prolongational tree. 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 mea-

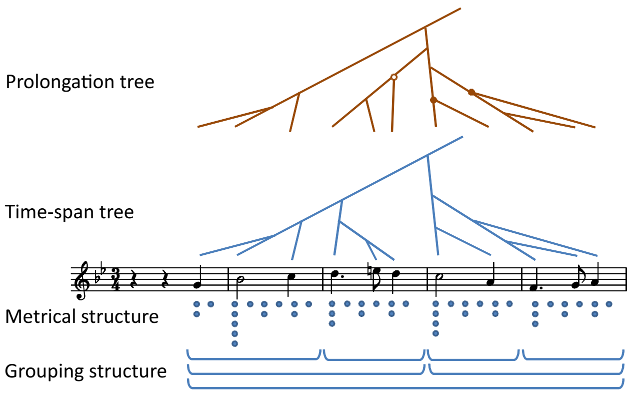

Figure 1. Grouping structure, metrical structure, time-span tree, and prolongation tree

sures, and so on. Strong beats are illustrated as several levels of dots below the music staff.

3.1 Metrical Preference Rules

There are two types of rules in a GTTM, i.e., ”well-formedness” and ”preference”. Well-formedness rules are necessary for the assignment of a structure and restrictions on the structure. When more than one structure satisfies the well-formedness rules, the preference rules indicate the superiority of one structure over another.

There are ten metrical preference rules (**MPRs**): **MPR1** (parallelism), **MPR2** (strong beat early), **MPR3** (event), **MPR4** (stress), **MPR5** (length), **MPR6** (bass), **MPR7** (cadence), **MPR8** (suspension), **MPR9**(time-span interaction), and **MPR10** (binary regularity). **MPR5** has six cases: (a) pitch-event, (b) dynamics, (c) slur, (d) articulation, (e) repeated pitches, and (f) harmony.

3.2 Conflict Between Rules

Because there is no strict order for applying **MPR**s, a conflict between rules often occurs when applying them, which results in ambiguities in analysis.

Figure 2 shows an example of the conflict between **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 well-formedness rule 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.

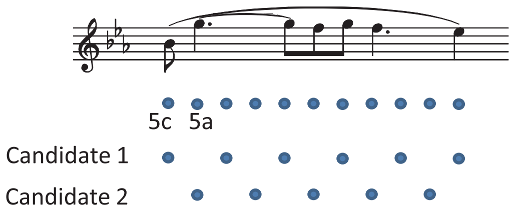

Figure 2. Example of conflict between MPRs

A beat that is applied to many rules tends to be strong in the analysis of a metrical structure. However, the number of rules cannot be determined because the priority of rules differs depending on the context of a piece.

We expect to learn the rule application and priority of rules by inputting a whole song with labels of the applied rules to a deep layered network.

3.3 Ambiguous Rule Definition

Some rules in a GTTM are expressed with ambiguous terms. For example **MPR5** is defined as follows.

> The **MPR5 (Length)**, preference for a metrical structure in which a relatively strong beat occurs at the inception of either
>
> a. a relatively long pitch-event,
>
> b. a relatively long duration of a dynamic,
>
> c. a relatively long slur,
>
> d. a relatively long pattern of articulation,
>
> e. a relatively long duration of a pitch in the relevant levels of the time-span reduction, or
>
> f. a relatively long duration of a harmony in the relevant levels of the time-span reduction (harmonic rhythm)

The term ”relatively” in this sense is ambiguous. Another example is that a GTTM has rules for selecting proper structures when discovering similar melodies (called parallelism) but does not define similarity. For example **MPR1** is defined as follows.

> The **MPR1 (Parallelism)**, where two or more groups or parts of groups can be construed as parallel, they preferably receives a parallel metrical structure.

3.4 Context Dependency

To solve the problems discussed in Subsections 3.2 and 3.3, we proposed the machine executable extension of GTTM (exGTTM) and ATTA [2]. Figure 3 is an example of an application of **MPR4**, **5a**, **5b**, and **5c** in the exGTTM and ATTA. By configuring the threshold parameters $T^j (j = 4, 5a, 5b, \; and \; 5c)$, we can control whether each rule is applicable. However, proper values of the parameter depend on the piece of music and on the level of hierarchy in the metrical structure. Therefore, the automatic estimation of proper values of the parameters is difficult.

3.5 Less Precise Explanation of Feedback Link

A GTTM has various feedback links from higher-level structures to lower-level ones. For example **MPR9** is defined as follows.

> The **MPR9 (Time-span Interaction)** has preference for a metrical analysis that minimizes conflict in the time-span reduction.

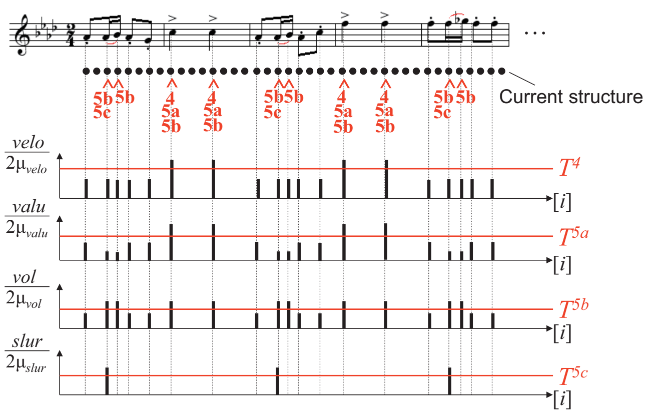

Figure 3. Application of MPR4, 5a, 5b, and 5c in ATTA

However, no detailed description and only a few examples are given. Other feedback links in the GTTM rules are not explicit. For example, analyzing the results of a time-span tree strongly affects the interpretation of chord progression, and various rules are related to chord progression, e.g., **MPR7 (Cadence)** requires a metrical structure in which cadences are metrically stable.

For complete implementation of a GTTM based on deep learning, we have to introduce the feedback link by using recurrent neural network; however, we do not focus on the feedback link in this paper.

4. DEEPGTTM-II: METRICAL STRUCTURE ANALYZER BASED ON DEEP LEARNING

We adopted deep learning to analyze the structure of a GTTM and solve the problems described in Subsections 3.2, 3.3, and 3.4. There are two main advantages in adopting deep learning.

- Learning rule applications
  We constructed a deep-layered network that can output whether each rule is applicable on each level of beat by learning the relationship between the scores and positions of applied MPRs with deep learning.

  Previous analysis systems based on a GTTM were constructed by a researchers and programmers. As described in Subsection 3.3, some rules in a GTTM are very ambiguous and the implementations of these rules might differ depending on the person.

  However, the deepGTTM-II is a learning-based analyzer the quality of which depends on the training data and trained network.

- Learning priority of rules
  Our FATTA does not work well because it only determines the priority of rules from the stability of the structure because the priority of rules depends on the context of a piece of music. The input of the network in the deepGTTM-II, on the other hand, is the score and it learns the priority of the rules as the weight and bias of the network based on the context of the score.

This section describes how we generated a metrical structure by using deep learning.

4.1 Datasets for training

Three types of datasets were used to train the network, i.e., a non-labeled dataset for pre-training, half-labeled dataset, and labeled dataset for fine-tuning (Fig. 4).

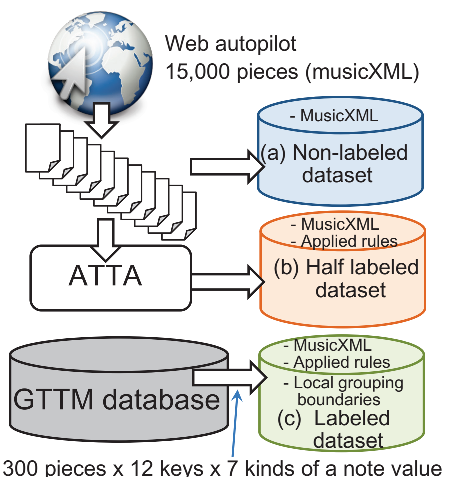

Figure 4. Non-labeled, half-labeled, and labeled datasets

(a) **Non-labeled dataset**. The network in pre-training learned the features of the music. A large-scale dataset with no labels was needed. Therefore, we collected, 15,000 pieces of music formatted in musicXML from Web pages that were introduced on the musicXML page of MakeMusic Inc. [11] (Fig. 4a). The musicXMLs were downloaded in the following three steps.

  (1) Web autopilot script made a list of urls that probably downloaded musicXMLs in five links from the musicXML page of MakeMusic Inc.

  (2) The files in the url list were downloaded after they had been omitted because they were clearly not musicXML.

  (3) All the downloaded files were opened using the script, and files that were not musicXML were deleted.

(b) **Half Labeled Dataset**. The network in fine-tuning learned with the labeled dataset. We had 300 pieces of music with a labeled dataset in the GTTM database, which included musicXML with a metrical structure, and positions to which the MPRs were applied. However, 300 pieces were insufficient for deep learning. Consequently, we constructed a half-labeled dataset. We automatically added the labels of the seven applied rules of **MPR2**, **3**, **4**, **5a**, **5b**, **5c**, and **5d**. These rules can be uniquely applied from a score when we give the threshold values. We used our ATTA to add labels to these rules (Fig. 4b). With the ATTA, the strength of the beat dependent on each MPR can be expressed as

${D_i}^j (j = 2, 3, 4, 5a, 5b, 5c,\ and\ 5d,\ 0 \leq {D_i}^j \leq 1)$. For example, **MPR4** is defined in a GTTM as follows.

> The **MPR4 (Event)**, preference for a metrical structure in which beats of level L_i that are stressed are strong beats of L_i.

We formalized ${D_i}^4$ as follows.

$$ {D_i}^4 = \begin{cases} 1 & velo_i > 2 \times \mu_{velo} \times T^4 \\ 0 & else, \end{cases} \tag{1} $$

where $velo_i$ is the velocity of a note from beat i, μ_{velo} is the average of $velo_i$, and T^j $(0 \leq T^j \leq 1)$ are the threshold parameters to control the those that determines whether the rules are applicable (${D_i}^j = 1$) or not (${D_i}^j = 0$). We used 1 as the threshold parameter value ($T^j = 1,\ where\ j = 2, 3, 4, 5a, 5b, 5c,\ and\ 5d$).

(c) **Labeled dataset**. We collected 300 pieces of 8-bar-long, monophonic, classical music and asked people with expertise in musicology to analyze them manually with faithful regard to the MPRs. These manually produced results were cross-checked by three other experts.

We artificially increased the labeled dataset because 300 pieces of music 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 the note values to two times, four times, eight times, half time, quarter time, and eighth time. Thus, the total labeled dataset had 25,200 (= 300x12x7) pieces (Fig. 4c).

4.2 Deep Belief Network

We used a deep belief network (DBN) to generate a metrical structure. Figure 6 outlines the structure for this DBN. The input of the DBN is the onset time, offset time, pitch, and velocity of note sequences from musicXML and grouping structure manually analyzed by musicologists. Each hierarchical level of the grouping structure is separately inputted by a note neighboring the grouping boundary as 1; otherwise, 0.

The output of the DBN formed multi-tasking learning, which had eight outputs in each hierarchical level of the metrical structure, such as seven types of MPRs (**MPR2**, **3**, **4**, **5a**, **5b**, **5c**, and **5d**) and one level of the metrical structure. Individual outputs had two units, e.g., rules that were not applicable (=0) and rules that were 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 layer from 1 to the final hidden layer. The network was learned in the four steps below. The order of music pieces was changed at every epoch in all steps.

(a) **Pre-training hidden layers from 1 to n.** Unsupervised pre-training was done by stacking restricted Boltzmann machines (RBMs) from the input layer to the hidden layer n. Pre-training was repeated for a hundred epochs using 15,000 pieces in a non-labeled dataset.

(b) **Fine-tuning of MPR 2, 3, 4, 5a, 5b, 5c, and 5d.** After pre-training, the network involved supervised fine-tuning by back propagation from output to input layers. The fine-tuning of MPR2, 3, 4, 5a, 5b, 5c, and 5d were repeated for one hundred epochs using 15,000 pieces in the half-labeled dataset.

(c) **Fine-tuning of one level of metrical structure.** After learning the MPRs, the network involved supervised fine-tuning by back propagation using the labeled dataset of 25,200 pieces at a level of the metrical structure.

(d) **Repeat pre-training and fine-tuning for next level of metrical structure.** If the metrical structure has a next level (more than two dots), add one hidden layer and pre-train the hidden layer using the non-labeled dataset then repeat (b) and (c).

4.3 Multi-dimensional multi-task learning

The DBN we introduced in Subsection 4.2 was a very complex network. The fine-tuning of one level of the metrical structure was multi-task learning. The fine-tuning of each metrical preference rule also involved multi-task learning. Therefore, the fine-tuning of MPRs involved multi-dimensional multi-task learning (Fig. 5).

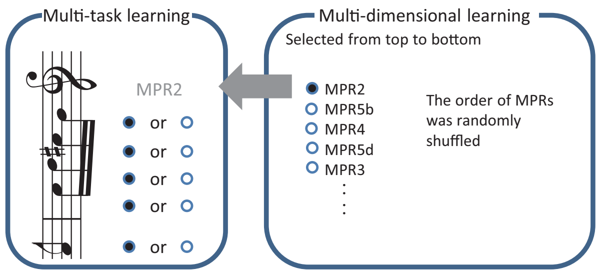
Multi-task learning
MPR2
or
or
or
or
or
Multi-dimensional learning
Selected from top to bottom
MPR2
MPR5b
MPR4
MPR5d
MPR3
The order of MPRs was randomly shuffled

Figure 5. Multi-dimensional multi-task learning

Multi-task learning. The processing flow for the multi-task learning of an MPR or metrical dots involved four steps.
Step 1: The order of the music pieces of training data was randomly shuffled and a piece was selected from top to bottom.
Step 2: The beat position of the selected piece was randomly shuffled and a beat position was selected from top to bottom.
Step 3: Back propagation from output to input was carried out in which the beat position had a strong beat or the rule was applied (=1) or was not (=0).
Step 4: The next beat position or the next piece in steps 2 and 3 was repeated .

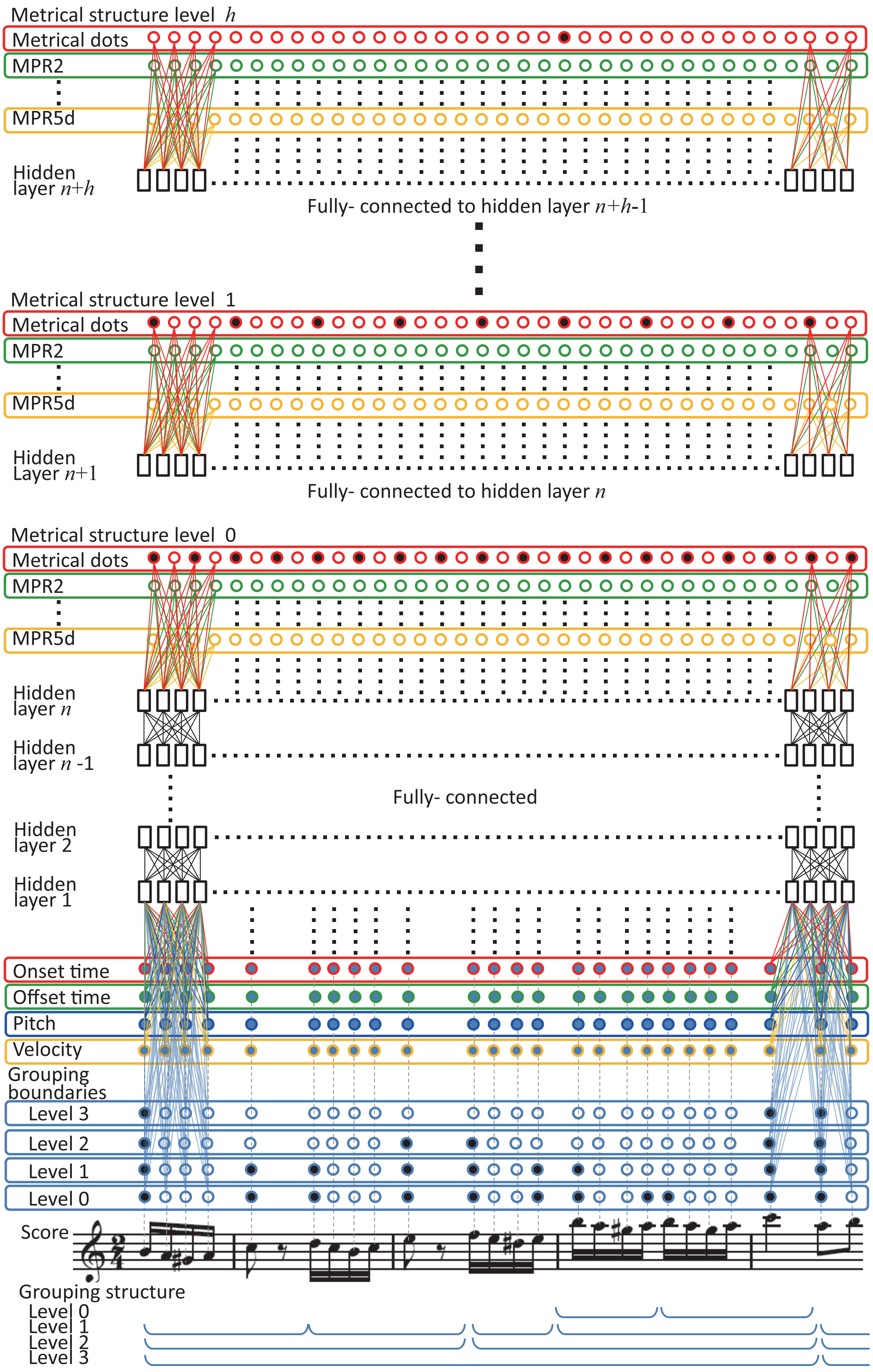

Figure 6. Deep belief network for generating metrical structure

Multidimensional multi-task learning. The processing flow for the multi-dimensional multi-task learning of MPRs involved the following three steps.
Step 1: The order of MPRs was randomly shuffled and a rule was selected from top to bottom.
Step 2: Multi-task learning of the selected MPR was carried out.
Step 3: The next rules in step 1 were repeated.

5. EXPERIMENTAL RESULTS

We evaluated the F_{measure} of the deepGTTM-II by using 100 music pieces from the GTTM database, where the remaining 200 pieces were used to train the network. The F_{measure} is given by the weighted harmonic mean of precision P (proportion of selected dots that are correct) and recall R (proportion of correct dots that were identified).

$$F\quad \text{measure} = 2 \times \frac{P \times R}{P + R} \tag{2}$$

Table 1 summarizes the results for a network that had 11 hidden layers with 3000 units. The ATTA had adjustable parameters and its performance changed depending on the parameters. For the default parameter, we use the middle value of the range of the parameter [2]. The FATTA had no parameters for editing.

The results indicate that the deepGTTM-II outperformed FATTA and ATTA with both default parameters and configured parameters in term of the F_{measure}. This results show that the deepGTTM-II performed extremely robustly.

6. CONCLUSIONS

We developed a metrical structure analyzer called deepGTTM-II that is based on deep learning. The following three points are the main results of this study.

- **Music analyzer based on Deep Learning**
  It has been revealed that deep learning is strong for various tasks. We demonstrated that deep leaning is also strong for music analysis. We will try to implement other music theory based on deep learning. Although we collected 300 pieces of music and analyzed the results of a GTTM by musicologists, the 300 labeled datasets were not sufficient for learning a deep-layered network. We therefore used our previous a GTTM analyzer called ATTA to prepare three types of datasets, non-labeled, half labeled, and labeled, to learn the network .

- **High-accuracy GTTM analyzer without manual editing**
  Previous GTTM analyzers, such as ATTA and σGTTM, require manual editing; otherwise, the performance will be much worse. The F_{measure}s of GTTM analyzers without manual editing, such as FATTA, σGTTM, σGTTMIII, and pGTTM, is too low (under 0.8). On the other hand, the deepGTTM-II shows extremely high performance, which indicates the possibility of its practical use in GTTM applications [15–19,29]. We plan to implement the entire GTTM analysis process based on deep learning.

- **Multi-dimensional multitask learning**
  We proposed multi-dimensional multi-task learning analyzer that efficiently learns the hierarchical level of the metrical structure and MPRs by sharing the network. Multi-dimensional multi-task learning is expected to be applied to other data that have a hierarchy and time series such as film [30] and discussion [31]. After a network that had 11 layers with 3000 units had been learned, the deepGTTM-II outperformed the previously developed analyzers for obtaining a metrical structure in terms of the F_{measure}.

This work was one step in implementing a GTTM by using deep learning. The remaining steps are to implement time-span reduction analysis and prologational reduction analysis of a GTTM based on deep learning. There are two problems as follows. One is generating tree structures because time-span and prolongation tree structures are more complex than a hierarchical metrical structure. The other problem is the lack of training samples because there are many combinations of tree structures and an unlearned sample sometimes appear in test data. We will attempt to solve these problems and make it possible to construct a complete GTTM system based on deep learning.

In the current stage, we cannot understand the details on why deep learning works extremely well for metrical analysis in a GTTM. Thus, we also plan to analyze a network after a metrical structure is learned.

7. REFERENCES

[1] F. Lerdahl and R. Jackendoff, *A Generative Theory of Tonal Music*, ser. Mit Press series on Cognitive theory and mental representation. MIT Press, 1985.

[2] M. Hamanaka, K. Hirata, and S. Tojo, “Implementing ’a generative theory of tonal music’,” *JNMR*, vol. 35, no. 4, pp. 249–277, 2006.

[3] M. Hamanaka, K. Hirata, and S. Tojo, “Fatta: Full automatic time-span tree analyzer,” in *Proceedings of ICMC2007*, 2007, pp. 153–156.

[4] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[5] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, “Why does unsupervised pre-training help deep learning?” *JMLR*, vol. 11, pp. 625–660, 2010.

[6] MakeMusic Inc., “Finale,” 2016, http://www.finalemusic.com/.

Melodies	deepGTTM-II	ATTA (Default prameters)	ATTA (Configured parameters)	FATTA
1. Grande Valse Brillante	0.94	0.88	0.93	0.88
2. Moments Musicaux	1.00	0.95	1.00	1.00
3. Trukish March	0.98	0.91	0.96	0.96
4. Anitras Tanz	0.90	0.82	0.86	0.82
5 Valse du Petit Chien	0.99	0.87	0.92	0.95
	:	:	:	:
Total (100 melodies)	0.96	0.84	0.90	0.88

Table 1. Performance of deepGTTM-II, ATTA, and FATTA

[7] M. Hamanaka, K. Hirata, and S. Tojo, "Music structural analysis database based on gttm," in *Proceedings of ISMIR2014*, 2014, pp. 325–330.

[8] E. Narmour, *The Analysis and Cognition of Basic Melodic Structures: The Implication-realization Model*. University of Chicago Press, 1990.

[9] E. Narmour, *The Analysis and Cognition of Melodic Complexity: The Implication-Realization Model*. University of Chicago Press, 1992.

[10] S. Yazawa, M. Hamanaka, and T. Utsuro, "Melody generation system based on a theory of melody sequences," in *Proc. of ICAICTA2014*, 2014, pp. 347–352.

[11] S. Heinrich, *Free Composition: New Musical Theories and Fantasies*. Pendragon Pr, 5 2001.

[12] A. Marsden, "Software for schenkerian analysis," in *Proc. of ICMC2011*, 2011, pp. 673–676.

[13] D. Temperley, *The Cognition of Basic Musical Structures*. MIT Press, 2004.

[14] F. Lerdahl, *Tonal Pitch Space*. Oxford University Press, USA, 2001.

[15] K. Hirata and S. Matsuda, "Interactive music summarization based on generative theory of tonal music," *JNMR*, vol. 5, no. 2, pp. 165–177, 2003.

[16] K. Hirata and R. Hiraga, "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, pp. 72–73.

[17] K. Hirata and S. Matsuda, "Annotated music for retrieval, reproduction," in *Proc. of ICMC2004*, 2004, pp. 584–587.

[18] M. Hamanaka, K. Hirata, and S. Tojo, "Melody expectation method based on gttm and tps," in *Proc. of ISMIR2008*, 2008, pp. 107–112.

[19] M. Hamanaka, K. Hirata, and S. Tojo, "Melody morphing method based on gttm," in *Proc. of ICMC2008*, 2008, pp. 155–158.

[20] D. Rosenthal, "Emulation of human rhythm perception," *CMJ*, vol. 16, no. 1, pp. 64–76, 1992.

[21] M. Goto, "An audio-based real-time beat tracking system for music with or without drum-sounds," *JNMR*, vol. 30, no. 2, pp. 159–171, 2001.

[22] S. Dixon, "Automatic extraction of tempo and beat from expressive performance," *JNMR*, vol. 30, no. 1, pp. 39–58, 2001.

[23] M. Davies and S. Bock, "Evaluating the evaluation measures for beat tracking," in *Proc. of ISMIR2014*, 2014, pp. 637–642.

[24] Y. Miura, M. Hamanaka, K. Hirata, and S. Tojo, "Decision tree to detect gttm group boundaries," in *Proc. of ICMC2009*, 2009, pp. 125–128.

[25] K. Kanamori and M. Hamanaka, "Method to detect gttm local grouping boundarys based on clustering and statistical learning," in *Proc. of SMC2014*, 2014, pp. 1193–1197.

[26] M. Hamanaka, K. Hirata, and S. Tojo, "σgttmiii: Learning based time-span tree generator based on pcfg," in *Proc. of CMMR2015*, 2015, pp. 303–317.

[27] E. Nakamura, M. Hamanaka, K. Hirata, and K. Yoshii, "Tree-structured probabilistic model of monophonic written music based on the generative theory of tonal music," in *Proc. of ICASSP2016*, 2016, pp. 276–280.

[28] M. Hamanaka, K. Hirata, and S. Tojo, "deepgttm-i: Local boundaries analyzer based on deep learning technique," in *Proc. of CMMR2016*, 2016, pp. 6–20.

[29] M. Hamanaka, M. Yoshiya, and S. Yoshida, "Constructing music applications for smartphones," in *Proc. of ICMC2011*, 2011, pp. 308–311.

[30] S. Takeuchi and M. Hamanaka, "Structure of the film based on the music theory," in *JSAI2014*, 2014, 1K5-OS-07b-4 (in Japanese).

[31] T. Oshima, M. Hamanaka, K. Hirata, S. Tojo, and K. Nagao, "Development of discussion structure editor for discussion mining based on muisc theory," in *IPSJ SIG DCC*, 2013, 7 pages (in Japanese).