The rest of the paper is organized as follows. In Section II, we summarize the hypernetwork model and its extension with evolutionary scheme. We describe methods for learning and generating music with EHNs in Section III and presents experimental results in Section IV. Concluding remarks are given in Section V.

II. The Hypernetwork Model

A hypernetwork is a weighted random hypergraph in which higher-order interactions of vertices are explicitly represented in hyperedges. Each hyperedge is associated with a weight representing the strength of the association across the vertices forming the hyperedge [1].

Formally, a hypergraph is an undirected graph $G=(V,E)$ whose edges connect a non-null number of vertices, where $V=\{v_1,v_2,\ldots,v_n\}$ is a set of vertices, $E=\{e_1,e_2,\ldots,e_m\}$ is a set of hyperedge. A hyperedge $e_i=\{v_{i1},v_{i2},\ldots,v_{ik}\}$ contains one or more vertices and the number of vertices, k in this case, is called as the order of the hyperedge (Fig. 1).

A hypernetwork is defined as $H=(X,E,W)$ where *X*, *E*, and *W* are the sets of vertices, edges, and weights, respectively. Both its structure (hyperedge compositions) and parameters (hyperedge weights) are learned by molecular evolutionary processes using the primitive operations of matching, selection, and amplification of hyperedges.

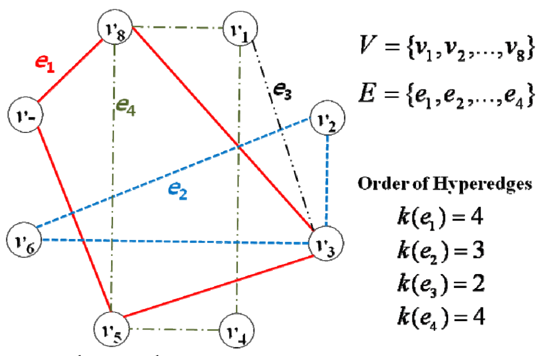

Fig. 1. Sample hypergraph with eight vertices and four hyperedges

For a data set, one attribute corresponds to one vertex of a hypernetwork. Given an input data, hyperedges are constructed by random sampling of the attributes. By means of hyperedges, a hypernetwork contains compressed information of the input data.

We can do binary classification with following simple, but effective algorithm (Fig. 2).

1. Present an input pattern x
2. Extract all hyperedges that are matched to the input pattern x
3. Count the number of each class in the extracted hyperedges
4. Classify the input pattern x as the class that has the highest count in Step 3

Fig. 2. Simple binary classification algorithm with hypernetworks

Learning a hypernetwork corresponds to searching combinatorial spaces which are represented by a library of hyperedges and learning step contains concepts of evolutionary computation. The main evolutionary feature in hypernetwork learning is the evolution of the structure of network. Other fundamental features of evolutionary computation, like population encoding, selection, mutation, and fitness function, are contained either implicitly or explicitly [5].

III. Learning Music by Evolving Hypernetworks with Music Corpus

A. *Creating hypernetworks from music corpus*

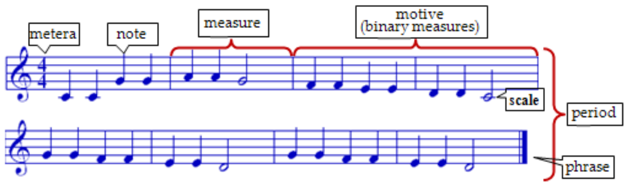

Fig. 3. Basic elements that constitute a score

Fig. 3 shows the basic music words. We use a monophonic melody in MIDI format and we only consider notes in the music in this paper. A note can be interpreted as a tuple <pitch, duration> and it constitutes the unit block of order 1 in a hyperedge.

When a music corpus is given, we build a library of hyperedges by sampling consecutive notes randomly. We allow various numbers of notes in a hyperedge, i.e. we use mixed ordered library.

B. *Training hypernetworks with evolution*

The purpose of training is to build a hypernetwork model that reproduces the melodies of the music in the training. Formally, we can consider the music generation as a conditional probability prediction problem. Given a cue $\mathbf{x}_c$ $=(x_1, x_2, \ldots, x_s)$, we want to get prediction $\hat{\mathbf{x}}_p=(\hat{x}_{s+1},\hat{x}_{s+2},\ldots,\hat{x}_m)$ that maximizes the conditional probability:

$$p(\hat{x}_{s+1},\hat{x}_{s+2},\ldots,\hat{x}_m \mid x_1,x_2,\ldots,x_s). \qquad (1)$$

In the hypernetwork model, the joint distribution of attributes x is roughly expressed as the distribution of hyperedge weights and learning a model that gives better prediction consist in adjusting this distribution from the training data. To optimize weight distribution, we use an evolutionary algorithm using population coding with a library of hyperedges and evolution of this library- by step-wise adjustment of weights.

In the training step, we set a sliding window of length *m,* for each score in the corpus, for each position, set the prefix of length *s* as a cue, and try to predict remaining (*m-s*) notes based on current library. Weights of hyperedges that participate in prediction are either increased or decreased according to the weight update scheme in Fig. 4. Hyperedges

```
Weight update scheme:
  - If the predicted note is correct, then do nothing
  - If incorrect,
      ■ Decrease the weight of the selected hyperedge by
          its order $k$
      ■ Choose randomly one hyperedge which have
          correct prediction from the set of matched
          hyperedges and then increase its weight by its
          order $k$

Replacement scheme:
  - Remove predefined ratio of hyperedges of which
     weights are lowest in the library
  - Randomly sample hyperedges with order $k$ from the
     pool of hyperedges in the corpus according to the
     probability of order $k$ in the current library, i.e., $prob(k)$
  - update $prob(k)$
```

Fig. 4. Schemes for weight update and hyperedge replacement. Replacement scheme is same with [3].

with lower order represents globally frequent patterns in the data and those with higher order represents local specific patterns. We apply order-proportional reward or penalty. When the sliding window has swept the whole music in the given training corpus, or after one epoch, we replace hyperedges that have low weights according to the replacement scheme. The whole algorithm for training hypernetworks for music generation is summarized in Fig. 5.

C. Music completion as validation

With the trained hypernetwork and given fragment of melody as a cue, we generate music by continuously predicting notes.

For validation or test set, we calculate the success rate for each position after the cue, i.e., s+1, s+2, …, m-th position in the total prediction $\hat{\mathbf{x}}_p = (\hat{x}_{s+1}, \hat{x}_{s+2}, ..., \hat{x}_m)$. As sliding window moves along all the melodies in the set, we sum success count for each point and divide the sum by the total count of trials.

IV. Experimental Results

A. Experimental setup

We designed two experiments to check the performance of our method as summarized in Table 1. Task 1 was conducted for predicting the sequence of melody. Four distinct sets of songs in MIDI format are used, which consist of two sets of Korean pop songs and two sets of folk songs (Table 2). We extracted the monophonic melody track manually and normalized the playing speed as 100 beats per minute. Duration of notes is quantized so that the melody is expressed as a sequence of notes.

Parameters for the evolutionary hypernetwork we applied are described in Table 3.

```
Input:
  Training set – a set of monophonic scores
  Parameters for an EHN

Output:
  A library with modified set of weights
  Success rate of generation for the training set

Initialization:
  Build a library of hyperedges by random selection from the
  training set
  Give each hyperedge equal weight $w_0$
  Set the ratio of each order uniformly

Main training loop:
For each melody in the training set
  Do
    Set a sliding window and the initial cue

    For $i=s+1$ to $m$
      - Gather every hyperedges that meet the matching
         condition on current cue.
      - Select the last note in the hyperedge with the
         highest weight as our prediction for the current
         position
      - Matching condition: if prefix with length ($k$-1) of a
         hyperedge of order $k$ matches exactly to the suffix
         of the cue, we take it as a candidate of matching
         edges
      - Update weights according to the 'weight update
         scheme'
      - Extend cue by appending predicted note to the
         current cue
    End For

    Shift the sliding window by one note
  Until the end of the melody

  Replace hyperedges According to the 'replacement
  scheme'
End For
```

Fig. 5. Algorithm to train an EHN to generate music

Table 1. Tasks that will be used for checking the performance of our method

ID	Description	Main parameters
Task 1	Learning and recalling short fragments	Length of the cue: 8 notes Length of recalled melody: 6
Task 2	Long generation with cue	Length of the cue: 8 Length of generated melody : 20

Table 2. Description of the music corpus for the experiment.

Index	Abbr.	Genre	Description	Count
1	K-Kim.	K-Pop	Gunmo Kim's songs	15
2	K-Shin.	K-Pop	Seoung-Hun Shin's songs	15
3	Scot.	Folk Songs	Scotch folk songs	15
4	Amer.	Folk Songs	American folk songs	15

Table 3. Parameter setting for the experiment

- Size of a library = 5000
- Order of hyperedge in the library $k = \{2,3,4,5,7,9\}$
- Replacement rate = 5%
- Iteration count for the training (epochs) = 50
- Default weight of a hyperedge = 1000

B. *Results for learning to recall short fragments*

We train a hypernetwork with the setting of eight-note cue and six-note prediction. For the evaluation, we calculate the success rate per each position. The success rate of each position increases concurrently (Fig. 6, Table 4). Even though we did continuous prediction, i.e. predicted note is added to the cue for the next prediction, the success rate for the sixth position is 88% on average. This result can be explained as the effect of self-organized combination of low-order hyperedges which contains short-term patterns that are frequent in the given set and higher-order hyperedges that represents specific patterns.

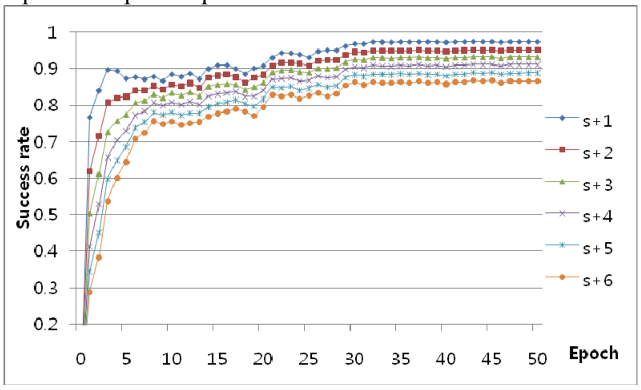

Fig. 7. Success rates (y-axis) of recall for each position (s+1~s+6) vs. epochs (x-axis) for the K-Shin set.

Table 4. Success rate per each position on each music set

Location / Music set	Success Rate(%)					
	s+1	s+2	s+3	s+4	s+5	s+6
K-Kim.	97.30	95.15	93.06	91.19	89.31	87.35
K-Shin.	97.36	95.15	93.32	91.22	88.94	86.75
Schot.	97.49	95.51	93.58	91.45	89.33	87.28
Amer.	98.38	97.25	95.87	94.55	93.23	91.92
Average	97.63	95.77	93.96	92.10	90.20	88.33

C. *Results for generating long melodies with cue*

Using each set of music in Table 2, we trained a hypernetwork relatively and then tried to generate long sequence against given cues. One of the major purposes of this experiment is to check if different hypernetworks generate different style.

It is hard to check the style of generated music or the similarity just based on the score. So we applied human interpretation for the generated music. Out of twenty generated melodies with random cues extracted from the songs that are not contained in the corpus, five people agreed that thirteen melodies show the original style of songs that each hypernetwork has learned (data are not shown).

Fig. 7 shows one example of long melody generation based on the same cue and two different hypernetworks. The cue is an eight-note-long fragment from "Swanee River", one of American folk songs. Score A is the generated melody by the hypernetwork which learned from American folk songs and Score B is generated by the hypernetwork that learned Scotland folk songs. In the score A, the generated melody is different from the original, but it contains the style of banjo or accordion accompaniment when we here it. On the other hand, scores C and D show the style of Korean ballad.

D. *Analysis of the library of trained hypernetwork*

In this section, we inspect the trained hypernetwork at the library level.

Fig. 8 shows how the count for each order of hyperedges which got more weights changes during learning in the library. Initially, the count of orders is set uniformly. But as learning starts, the distribution becomes unstable immediately, and especially the count of order 2 hyperedges changes drastically. After enough learning, 38 epochs in this case, the count distribution seems to become stable and converged. At this point, two lowest orders (2 and 3) change

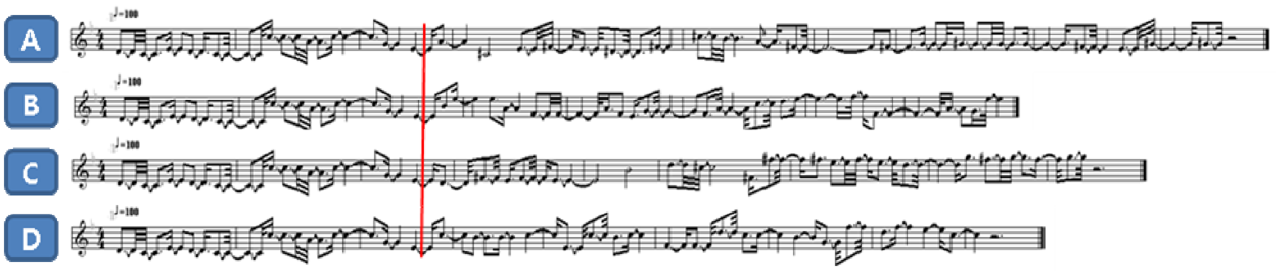

Fig. 6. Scores generated by EHNs that learned Amer. (A), Scot. (B), K-Kim (C), and K-Shin (D) with the cue (left side of the bar in the middle) from "Swanee River", the famous American folk song.

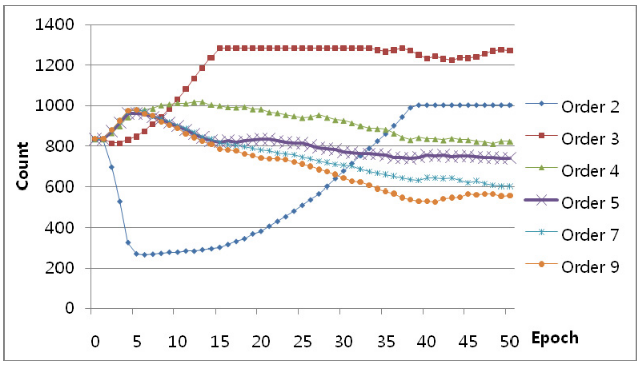

Fig. 8. Distribution of hyperedges of which weights are larger than the initial weight in hypernetwork trained by K-Shin

their rank in the ratio at this time. Other orders take rank according to their length and the count of hyperedges gets lower for these higher orders than the initial count.

This behavior can be interpreted that lower order hyperedges have not only high probability to be matched to the cue but also high probability to predict incorrectly. It can also be implied that after covering exception of general patterns by higher order hyperedges, lower order hyperedges can get higher weight than default weight.

On the initial phase of learning, the current model frequently fails to generate note sequences of full length, six in our experiment. This failure occurs when there is no matching hyperedge in current library. On this phase, replacement does significant role for learning. After enough replacement and weight update, we can expect that hyperedges with lower weights do not contain significant information on the learned music and replacement does not produce information loss any more. Fig. 9 shows this phenomenon. Before 33-rd epoch, failure occurs frequently, but after this point the model hardly fails in generation. This result seems to be deeply related to the change of amount of order 2 hyperedges. Lower order hyperedges serve the role of remembering and making general patterns. Therefore the adaptability of this model depends on the number of lower order hyperedges especially order 2 and order 3 in this experiment.

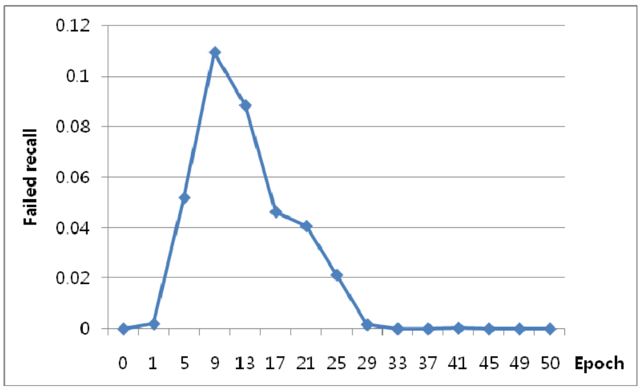

Fig. 9. The rate of failed recalls from hypernetwork trained by K-shin

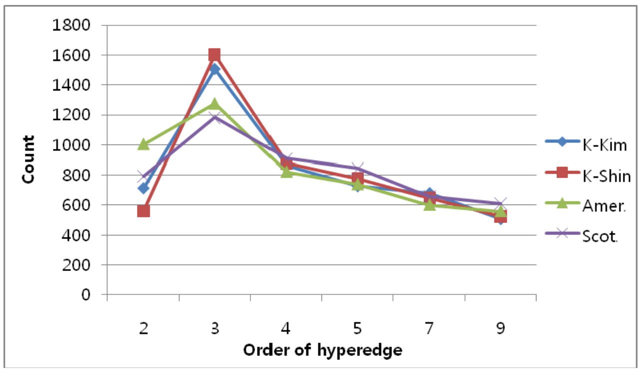

Fig. 10. The number of high weight hyperedges for each order with four different music corpora.

Fig. 10 shows the distribution of hyperedges per each order after enough training on four set of songs for our experiments.

In all data set, order 3 hyperedges is dominant. Evolutionary hypernetwork model seems to automatically find important order 3 hyperedges which contain core information on the melodic structure. According to the Narmour's theory, three successive notes are important for melody prediction and based on this theory I-R model is widely used for describing melodies[16] [17].

In summary, by extracting and learning with various length note-fragments under the ENH framework and our algorithm, we get following results:

- 88% of recall rates of the original six-note-long melody when a fragment of eight-note-long notes is given as a cue
- Generation of long melody with the learned style from the given music corpus

And these results can be explained from the following features of EHNs:

- With mixed order library of hyperedges, short-term and long-term sequential patterns can be extracted and combined to generate music with various styles
- we can extract a set of arguments that constitutes melodic context in music, especially three-note-long melodies which is known as the core in melodic perception of human

V. Concluding Remarks

In this study, we proposed a novel music generation model based on learning examples using evolutionary hypernetworks. Our model can learn patterns from songs and generate melodies with similar style to the learned data. Our model generates various styles of music not by changing algorithm or parameters but by changing training data set. In addition, people can analyze the pattern of music set and can extract a set of arguments that constitutes melodic structures in music by inspecting hyperedges in hypernetworks.

For macro-level human-like composition, it is necessary to learn global patterns. We are considering two approaches as our future works; using hyperedges which have non-contiguous patterns for encoding wide range pattern and additional information such as chord sequence, note density sequence.

REFERENCES

[1] B.-T. Zhang, "Hypernetworks: A molecular evolutionary architecture for cognitive learning and memory," *IEEE Computational Intelligence Magazine*, vol. 3, no. 3, pp. 49-63, 2008.

[2] B.-T. Zhang and J.-K. Kim, "DNA hypernetworks for information storage and retrieval," *Preliminary Proceedings of the Twelfth International Meeting on DNA Computing (DNA 12)*, pp. 283-292, 2006.

[3] J.-K. Kim, B.-T. Zhang, "Evolving hypernetworks for pattern classification," *IEEE Congress on Evolutionary Computation (CEC 2007)*, pp.1856-1862, 2007.

[4] S. Kim, S.-J. Kim, and B.-T. Zhang, "Evolving hypernetwork classifiers for microRNA expression profile analysis," *IEEE Congress on Evolutionary Computation (CEC 2007)*, pp.313-319, 2007.

[5] J.-W. Ha, J.-H. Eom, S.-C. Kim, and B.-T. Zhang, "Evolutionary hypernetwork models for aptamer-based cardiovascular disease diagnosis," *The Genetic and Evolutionary Computation Conference (GECCO 2007)*, pp. 2709-2716, 2007.

[6] E. Bautu, S. Kim, A. Bautu, H. Luchian, and B.-T. Zhang, "Evolving hypernetwork models of binary time series for forecasting price movements on stock markets," *IEEE Congress on Evolutionary Computation (CEC 2009),* 2009 (accepted).

[7] C. Raphael. "A probabilistic expert system for automatic musical accompaniment," *Journal of Computational and Graphical Statistics*, vol. 10, no. 3, pp. 467–512, 2001.

[8] R. B. Dannenberg, C. Raphael: "Music score alignment and computer accompaniment," *Commun. ACM*, vo. 49, no. 8, pp. 38-43, 2006.

[9] I. Simon, D. Morris, and S. Basu, "MySong: Automatic Accompaniment Generation for Vocal Melodies," *The twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pp. 725-734, 2008.

[10] A. T. Cemgil and S. J. Godsill. "Probabilistic Phase Vocoder and its application to Interpolation of Missing Values in Audio Signals," *The 13th European Signal Processing Conference*, 2005.

[11] M. Mozer, "Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing," *Connection Science*, vol. 6, pp. 247-280, 1994.

[12] D. Eck, J. Schmidhuber, "Finding temporal structure in music: Blues improvisation with LSTM recurrent networks," In Proc. *of the IEEE workshop on Neural Networks for Signal Processing*, pp. 747-756, 2002.

[13] J.-F. Paiement, Y. Grandvalet, and S. Bengio, "Predictive models for music," *IDIAP Research Report*, 08-51, 2008.

[14] D. Stammen and B. Pennycook, "Real-time recognition of melodic fragments using the dynamic timewarp algorithm," *ICMC*, pp. 232-235, 1993.

[15] T. Eerola and P. Toiviainen, *MIDI Toolbox: MATLAB Tools for Music Research*. University of Jyväskylä: Kopijyvä, Jyväskylä, Finland, 2004. Available at http://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/miditoolbox/

[16] E. Narmour, "The Analysis and Cognition of Basic Melodic Structures : The Implication- realization Model," Univeristy of Chicago Press , Chicago, 1990

[17] E. G. Schellenberg, "Expectancy in melody: tests of the implication-realization model", *Cognition*, vol. 58, no. 1, pp. 75-125, 1996.