Predicting Missing Music Components with Bidirectional Long Short-Term Memory Neural Networks

I-Ting Liu and Richard Randall

***Abstract*—**Successfully predicting missing components from complex multipart musical textures has attracted researchers of music information retrieval and music theory. Solutions have been limited to either two-part melody and accompaniment (MA) textures *or* four-part Soprano-Alto-Tenor-Bass (SATB) textures. This paper proposes a robust framework applicable to both textures using a Bidirectional Long-Short Term Memory (BLSTM) recurrent neural network. The BLSTM system was evaluated using frame-wise accuracies on the Nottingham Folk Song dataset and J. S. Bach Chorales. Experimental results demonstrated that BLSTM significantly outperforms other neural-network based methods by 4.6% on average for four-part SATB and two-part MA textures. The high accuracies obtained with BLSTM on both textures indicated that it is the most robust and applicable structure for predicting missing components from multi-part musical textures.

***Keywords*—** bidirectional long-short term memory neural network, multipart musical textures, music information retrieval

I. Introduction

THIS paper presents a method for predicting missing components from complex multipart musical textures. Specifically, we examine two-part melody and accompaniment (MA) and Soprano-Alto-Tenor-Bass (SATB) chorale textures. We treat each voice as a part (*e.g.* the melody of the MA texture or the Soprano of the SATB texture) and the problem we address is given an incomplete texture, how successfully can we generate the missing part. This project proposes a robust approach that is capable of handling both textures elegantly and has applications to any style of music. Predictions are made using a Bidirectional Long-Short Term Memory (BLSTM) recurrent neural network that is able to learn the relationship between components, and can thus be trained to predict missing components. This work demonstrates the capability of the BLSTM system by conducting experiments on the two tasks mentioned above with two distinct datasets.

Analyzing music with the aid of computer programs has attracted researchers of music information retrieval and music theory over the past twenty years. Music (especially western tonal music) has always been regarded as a kind of art with rigorous formalization [1]. Various complex rules regulate how notes can be and cannot be played together in complex multipart textures. Such rules change over time and are subject to multiple factors [2]. As artificial intelligence and machine-learning research advances, it is natural that computer scientists apply such technique to music analysis in order to elucidate these rules [3]. Two popular tasks investigated in this area are (1) generating chord accompaniments for a given melody in a two-part MA texture and (2) generating a missing voice for an incomplete four-part SATB texture. Successfully accomplishing either task manually is time-consuming and requires considerable style-specific knowledge and the applications discussed below are designed to automate and help non-professional musicians compose and analyze music.

Approaches that treat these problems can be categorized into two types according to the level of human engagement in discovering and applying music rules. Early works that handle incomplete four-part SATB textures were mostly knowledge-based models. Steels [4] proposed a representation system to encode musical information and exploit heuristic search, which takes the form of if-then musical rules that specify solutions under different conditions to generate voices. Ebcioglu built CHORAL, a knowledge-based system that includes over 350 rules modeling the style of Johann Sebastian Bach [5]. Due to the large number of rules involved, some studies modeled the problem as a constraint satisfaction problem, as was used by Pachet and Roy [6] on four-part textures and Ramirez et al. [7] on two-part textures. Knowledge-based genetic algorithms were also used as an alternative method to represent the rules. McIntyre [8] implemented a system that harmonizes user-defined melody in Baroque style, and Hall [9] presented a system that selects combinations of attributes to model the harmonization of J. S. Bach's chorales. Freitas and Guimaraes also implemented a system based on genetic algorithms in [10]. The fitness function and genetic operators rely on "music knowledges" to suggest chord progressions for given melodies.

While rules in knowledge-based systems have to be manually encoded into these systems, rules in probabilistic models and neural networks can be derived by training corpora without human intervention by the models. Hidden Markov Models (HMM) are one of the most common probabilistic models for the task of generating a chord sequence given melodies for two-part textures [11][12]. In HMM, a pre-selected dataset is used to train a transition probability matrix, which represents the probability of changing from one chord to another, and a melody observation matrix, the probability of encountering

R. Randall is with the School of Music and the Center for the Neural Basis of Cognition, Carnegie Mellon University, Pittsburgh, PA 15213 USA (412-268-4785; email: randall@cmu.edu).

I-T. Liu is with School of Music, Carnegie Mellon University, Pittsburgh, PA 15213 USA. (e-mail: itingl@andrew.cmu.edu).

each note when different chords are being played. The optimal chord sequence is then generated using dynamic programming, or Viterbi Algorithm. HMM are also used by Allan [13] [14] to harmonize four-part chorales in the style of J. S. Bach. In addition to HMM, Markov Model and Bayesian Networks are alternative models used for four-part textures by Biyikoglu [15] and Suzuki, et al. [16]. Raczynski, et al. [17] proposed a statistical model that combines multiple simple sub-models. Each sub-model captures different music aspects such as metric and pitch information, and all of them are then interpolated into a single model. Paiement, et al. [18] proposed a multi-level graphical model, which is proved to capture the long-term dependency among chord progression better than traditional HMM. One drawback of probabilistic models is that they cannot correctly handle data that are not seen in training data. Chuan and Chew [19] reduced this problem by using a hybrid system for style-specific chord sequence generation with statistical learning approach and music theory. In [20], Chuan compared and evaluated rule-based Harmonic Analyzer [21], probabilistic-model based MySong [18], and the hybrid system proposed in [19] on the task of style-specific chord generation for melodies.

Neural networks have also been used. Gang, et al. [22] were one of the earliest that used neural networks to produce chord harmonization for given melodies. Sequential neural network consisted of a sub-net that learned to identify chord notes for the melody in each measure, and the result was fed into the network to learn the relationship between melodies and chords. The network was later adopted in real-time application [23][24]. Consisting of 3 layers, the input layer takes pitch, metric information, and the current chord context, and the output layer predicts the next chord. Cunha, et al. [25] also proposed a real-time chord harmonization system using multi-layer perceptron (MLP) neural networks and a rule-based sequence tracker that analyzes the structure of the song in real-time, which provides additional information on the context of the notes being played.

Hoover, et al. [26] used two Artificial Neural Networks (ANN) to model the relationship between melodies and accompaniment as a function of time. The system was later extended to generate multi-voice accompaniment by increasing the size of the output layer in [27]. Bellgard and Tsand [3] trained an effective Bolzmann machine and incorporated external constraints so that harmonization follows the rules of chorales. Fuelner developed a feed-forward neural network that harmonizes melodies in specific styles in [28]. De Prisco, et al. [29] proposed a neural network that finds appropriate chords to harmonize given bass lines in four-part SATB chorales by combining three base networks, each of which models contexts of different time lengths.

Although these previous studies provide valuable insights, a number of constraints exist in their applications. Most rules encoded in knowledge-based systems are style-specific, making them hard to apply to other types of music efficiently. Probabilistic models and neural networks, on the other hand, provide a much more adaptable solution that can be applied to music of different styles by learning rules from different styles of training data. Nevertheless, many of the probabilistic models can only handle music pieces of fixed length. In addition, the transition matrix of probabilistic models has to be learned using specific music representation (*e.g.* chords) and cannot be generalized to other representations (*e.g.* SATB). Moreover, probabilistic models tend to ignore long-term dependency among music components as they mainly focus on local transitions between two consecutive components. Existing studies using neural networks captured long-term dependencies in music and also are capable of dealing with music pieces of arbitrary lengths. However, neural networks have been notoriously hard to train, and their ability to utilize long-term information was limited until the introduction of Long-Short Term Memory (LSTM) cells.

The current study builds on the LSTM model and has the advantages of several other models without their restrictions. Like probabilistic models, neural networks can learn music "rules" of different styles of music from training data. Yet unlike probabilistic models where a transition matrix among chords is required, neural networks enable the model to deal with flexible polyphonic accompaniment that does not necessarily have to be in the form of chords. Such structure also allows the model to incorporate additional musical quantities easily by adjusting the number of input neurons. Finally, adding *bidirectional links* and LSTM cells improves a neural network's ability to employ additional timing information. All of the above contributes to the fact that the proposed BLSTM model is flexible and effective in generating the missing component in an incomplete texture.

II. Background

A. Feed-Forward Neural Network

The most common neural network is a feed-forward multi-layered perceptron (MLP) network as shown in Fig. 1, which consists of three layers: an input layer, a hidden layer, and an output layer. The network is usually trained via back-propagation. Feed-forward neural networks were used extensively in music-related research such as harmony generation [22][25], onset detection [30], and algorithmic composition [31].

Though being the simplest among various neural network structures, feed-forward neural networks could not effectively capture rhythmic patterns and music structures in music owing to the fact that they do not have a mechanism to keep track of the notes played in the past. Since each input frame is processed individually, feed-forward neural networks are totally deterministic unless the context is provided to the network such as using a sliding window as in [30].

B. Recurrent Neural Network

Another way to present past information is to add recurrent links to the network, resulting in a recurrent neural network (RNN). A RNN has at least one feedback connection from one or more of its units to another unit, forming cyclic paths in the network. Fig. 2 shows a simple example of a RNN with an input layer, a hidden layer, and a output layer with recurrent links from the output layer to the input layer. RNNs are known to be able to approximate dynamical systems due to internal states that act as internal memory to process sequence of inputs through time. The fact that music is a complex structure that has both short-term and long-term dependency just as language models makes RNN an ideal structure for solving music-related problems. Mozer [32] used a fully-connected RNN to generate music note-by-note. Boulanger, Lewandowski, *et al*. also developed an RNN-based model to recognize chords in audio files [33] and construct polyphonic music [34] by using restricted Boltzmann machine (RBM) and recurrent temporal RBM (RTRBM).

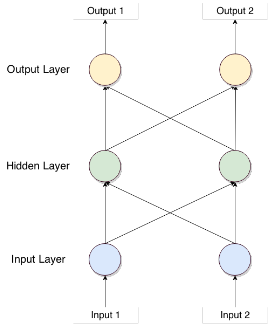

Fig. 1. A Multi-layer Feed-forward Neural Network

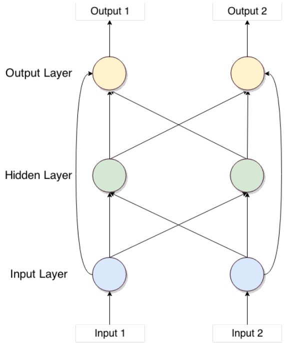

Fig. 2 A Multi-layer Recurrent Neural Network

C. Bidirectional Recurrent Neural Network

Standard RNNs process inputs in temporal order, and their outputs are mainly based on previous context. One way to include future information in the network is to use a bidirectional recurrent neural network (BRNN). Fig. 3 shows the structure of a BRNN with two hidden layers (one forward states, one backward states) unfolded in time. In a BRNN, two separate hidden layers are used, both connected to the same inputs and outputs. One of the layers processes inputs forward in time, while the other one processes inputs backward. Therefore, for each point of a given sequence, it could access complete temporal information before and after. BRNN has been used successfully to classify speech data in [35]. Eyben, *et al*. [36] also achieved impressive results on onset detection using BRNN. For the complete algorithm for training BRNN, please refer to [37].

D. Long Short Term Memory (LSTM)

Although BRNNs have access to both past and future information, they have been notoriously hard to train because of "vanishing gradients," [38] a problem commonly seen in RNNs when training with gradient based methods. Gradient methods, such as Back-Propagation Through Time (BPTT) [39], Real-Time Recurrent Learning (RTRL) [40] and their combinations, update the network by flowing errors "back in time." As the error propagates from layer to layer, it tends to either explode or shrink exponentially depending on the magnitude of the weights. Therefore, the network fails to learn long-term dependency between inputs and outputs. Tasks with time lags that are greater than 5-10 time steps are already difficult to learn, not to mention that dependency of music usually spans across tens to hundreds of notes in time, which contributes to music's unique phrase structures. Long short-term memory (LSTM) [38] algorithm was designed to tackle the error-flow problem.

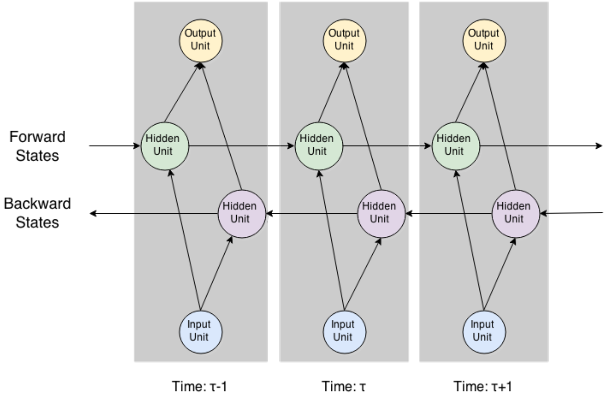

Fig. 3 A Bidirectional Recurrent Neural Network (BRNN) unfolded in time. Image from [37].

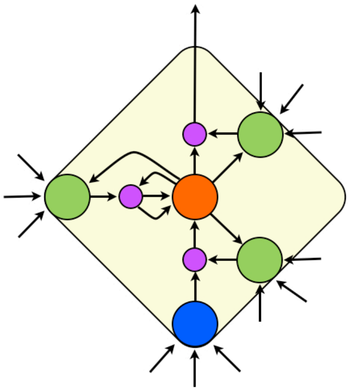

Fig. 4 A LSTM block that contains one linear cell (orange) and three non-linear gating units (green). Image from [41]

In a LSTM hidden layer, fully-connected memory blocks replace nonlinear units that are often used in feed-forward neural network. Fig. 4 shows an LSTM block. The core of a memory block is a linear cell (orange) that sums up the inputs, which has a self-recurrent connection of fixed weight 1.0, preserving all previous information and ensuring they would not vanish as they are propagated in time. A memory block also contains three sigmoid gating units: input gate, output gate, and forget gate. An input gate learns to control when inputs are allowed to pass into the cell in the memory block so that only relevant contents are remembered; an output gate learns to control when the cell's output should be passed out of the block, protecting other units from interference from current irrelevant memory contents; a forget gate learns to control when it is time to forget already remembered value, *i.e.* to reset the memory cell. When gates are closed, irrelevant information does not enter the cell and the state of the cell is not altered. The outputs of all memory blocks are fed back recurrently to all memory blocks to remember past values.

In this project, we use Bidirectional Recurrent Neural Networks as their recurrent links grant the network access to both information in the past and in the future. We also use LSTM cells in the network to avoid vanishing gradient problems, the details of which are covered below.

III. Methods

A. Music Representation

MIDI files are used as input in both training and testing phases in this project. Multiple input and output neurons are used to represent different pitches. At each time, the value of the neuron associated with the particular pitch played at that time is 1.0. The values of the rest of the neurons are 0.0. We avoid distributed encodings and other dimension reduction techniques and represent the data in this simple form because this representation is common and assumes that neural networks can learn a more distributed representation within hidden layers. Monophonic inputs are used because of their adaptability to both monophonic and polyphonic data. We leave further consideration and evaluation of distributed representation to future work.

The music is split into time frames and the length of each frame depends on the type of music. Finding missing music components can then be formulated as a supervised classification problem. For a song of length t_1, for every time t from t_0 to t_1, given input $x(t)$, the notes played at time t, find the output $y(t)$, which is the missing component we try to predict. In other words, for two-part MA textures, $y(t)$ is the chord played at time t, while for four-part SATB textures, $y(t)$ is the pitch of the missing part at time t.

B. Generating Accompaniment in Two-Part MA Texture

1) Input and Output

The MIDI files are split into eighth-note fractions. The inputs at time t, $x(t)$, are the notes of the melody played at time t. Instead of representing the notes by their MIDI number, which spans the whole range of 88 notes on a keyboard, we used pitch-class representation to encode pitches into their corresponding pitch-class number. Pitch class, also known as "chroma," is the set of all pitches regardless of their octaves. That is, all C notes (C*0*, C*1*. . . C$_n$, *etc.*) are all classified as pitch-class C. All notes are represented with one of the 12 numbers corresponding to the 12 semitones in an octave. In addition to pitch-class information, two additional values are added as inputs: Note-Begin unit and Beat-Onunit. In order to be able to tell when a note ends, a Note-Begin unit is used to differentiate two consecutive notes of the same pitch from one note that is held for two time frames as was done by [31]. If the note in the melody begins at time n, the value of the Note-Begin unit is 1.0; if the note is sounding, but is held from previous time n or is not played at all, the value of the unit is 0.0. The Beat-Onunit, on the other hand, provides metric information to the network. If the time t is on a beat, the value of the Beat-Onunit is 1.0, otherwise 0.0. If it is at rest, the values of all input neurons are 0.0. The time signature information is obtained via meta-data in MIDI files.

The outputs at time t, $y(t)$, is the chord played at time t. We limit chord selection to major, minor, diminished, suspended, and augmented triads as in [12], resulting in 52 chords in total. The output units represent these 52 chords in a manner similar to the input neurons: the value of the neuron corresponding to the chord played at that time has a value of 1.0, and the values of the rest of the neurons are all 0.0.

2) Training the Network

The input layer has 14 input neurons: 12 neurons for each pitch in the pitch-class, one neuron for note-begin and one for Beat-Onunit. The network consists of two hidden layers for both forward and backward states, resulting in four hidden layers in total. In every hidden layer are 20 LSTM blocks with one memory cell. The output layer uses the softmax activation function and cross entropy error function as in [35]. Softmax function is a standard function for multi-class classification that squashes a K-dimensional vector x in the range of (0,1), which takes the form

$$\sigma(x)_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_j}}, \text{ for } j = 1, ..., K \quad .$$

The softmax function ensures that all the output neurons sum to one at every time step, and thus can be regarded as the probability of the output chord given the inputs at that time. Each music piece is presented to the network one at a time, frame-by-frame. The network is trained via standard gradient-descent Back-Prorogation. A split of data is used as the validation set for early stopping in order to avoid over-fitting of the training data. If there is no improvement on the validation set for 30 epochs, training is finished and the network setting with the lowest classification error on the validation set is used for testing.

3) Markov Model as Post-Processing

The network trained in III-B2 can then be used to predict the chord associated with each melody note by choosing the output neuron that has the highest activation at each time point. However, the predicted chord at each time is independent of the chord predicted in the previous and succeeding time. While there are forward and backward links in the hidden layers of the network, there is no recurrent connections from the final neuron output to the network. The chord might sound good with the melody, but the transition from one chord to another might not make sense at all. In fact, how one chord transitions from and to the other typically follows specific chord-progression rules depending on different music styles. A bi-gram Markov Model is thus added to learn the probability of transitioning from each chord to possible successors independent of the melody, which will be referred to as the transition matrix. The transition matrix is smoothed using linear interpolation with a unigram model. The model also learns the statistics of the start chords.

Instead of selecting the output neuron with the highest activations, the first k neurons with the highest activations are chosen as candidates. Dynamic programming is then used to determine the optimal chord sequence among the candidates using the previously learned transition matrix.

C. Generating the Missing Part in SATB Textures

1) Input and Output

We sample the melody at every eighth note for similar reasons as explained by [29]. Notes that are shorter in length are considered as passing notes and are ignored here. The inputs at time *t*, $x(t)$, are the pitches of the notes played at time *t*, spanning the whole range of 88 notes (A0, C8) on a keyboard, resulting in an 88-dimensional vector. If a note *i* is played at time *t*, the value of the neuron associated with the particular pitch is 1.0, *i.e.*, $x_i(t)$=1.0. The number of non-zero elements in $x(t)$, which are the number notes played each time, ranges from one to three, depending on the number of voices present.

For the task of predicting the missing voice, either Soprano, Alto, Tenor or Bass, in a four-part texture where the other three voices are present, the input is three-part polyphonic music. In this case, there are at most three non-zero elements in x_t for every time *t*. If the task is to predict one missing voice given only one of the three other voices, there is at most one non-zero element in $x(t)$. Note that we do not add any additional information to the network about which voice is missing nor which voice(s) are given; the network induces such knowledge according to the input data and the target output data. The reason why we do not represent the notes with their pitch-class profile as we did when handling two-part MA texture is that the network depends on octave information to identify which voice the notes belong to. The outputs at time *t*, $y(t)$, is the predicted missing note at time *t*, which falls in the pitch range of any of the four voices, depending on the task specified by our training data. Similarly, the value of the neuron associated with the particular pitch played at the time *t* is 1.0, otherwise 0.0.

2) Training the Network

The network structure is the same as the one used in Section III-B2 except that the number of input neurons and output neurons are 88, and that we use 20 LSTM blocks for the first hidden layer and 50 LSTM blocks for the second hidden layer. Similar to what we did for two-part MA textures, each music piece is presented to the network one at a time, frame-by-frame. If the task is to generate one missing voice given any of the three other voices, then the three present voices are given to the network individually as if they are independent melodies. In this case, each music piece is actually presented to the network three times in total, and each time only one of the three voices is presented. Training is finished if there is no improvement on the validation data for 30 epochs.

3) Predict Missing Voice with the Trained Network

The trained network is ready to predict the missing voice by doing an 88-class classification on the input voice. At each time frame, the neuron with the highest activations is selected, and the pitch it represents is considered as the pitch of the missing voice.

IV. EVALUATION

A. Generating Missing Accompaniment in MA Texture

1) Dataset

The system's performance on two-part MA textures is evaluated using the Nottingham Dataset [42] transcribed from ABC format, which is also used in [34] for composing polyphonic music. The dataset consists of 1024 double-track MIDI files, with melody on one track and accompaniment on the other. The length of the pieces ranges from 10 seconds to 7.5 minutes, the median being 1 minute and 4 seconds. Those without accompaniment and those whose accompaniment are more complicated than simple chord progressions are discarded, resulting in 962 MIDI files comprising more than 1000 minutes, in total. Songs not in the key of C major nor A minor (874 of them) were transposed to C major/A minor after probabilistically determining their original key using Krumhansl-Schmuckler key-finding algorithm [43][44].

The chords were annotated at every beat or at every quarter note. Seventh chords were reduced to triads, and rests were replaced with previous chords. 60% of the dataset is selected randomly as training data, 20% as validation data, and 20% as testing data. Training finishes when validation accuracy does not improve for 30 epochs. All results for the training and testing sets were recorded at the time when the classification error on the validation set is lowest.

2) Effects of Including Metric Information in Input

Since the network learns the input melody as a sequence in time and has no access to information other than pitches, we added Beat-On flag to a frame when it is on a beat according to the time signature meta-data in MIDI files (Group iii and iv). We also added Note-Begin (Group ii and iv) to differentiate two consecutive notes of the same pitch from two distinctive notes, as mentioned in Section III-B1.

All three groups were sampled every eighth note, and the MIDI note range (50, 95) was used as the input range. Table I shows the classification accuracy of the three groups as well as the one where neither flag is provided as a reference. Two groups where Beat-On flag is added, Group iii and iv, perform significantly better than the groups without the beat information (Group i). This is consistent with the fact that chords always change on a beat or multiples of a beat. Therefore, such information is crucial to the timing of chord changes in the network. Note-Begin, on the other hand, does not seem to improve the accuracy, which is due to the fact that whether the note is held from the previous time or it is newly started does not affect chord choices.

TABLE I
CLASSIFICATION ACCURACY OF THE DATASET WHEN A NOTE-BEGIN FLAG, BEAT-ONFLAG, AND BOTH FLAGS ARE ADDED TO THE INPUTS.

	Training Set	Test Set
(i) Pitch Information only	72.88%	68.54
(ii) Note-Begin	72.11%	68.86
(iii) Beat-On	75.82%	70.34
(iv) Note-Begin and Beat-On	75.76%	**70.61**

TABLE II
CLASSIFICATION ACCURACY OF THE DATASET WHEN USING VARIOUS REPRESENTATIONS OF PITCHES AT VARIOUS SAMPLING RATES.

	Training Set	Test Set
(i) 8th Note + Melody Range	75.76%	70.65 %
(ii) 8th Note + Pitch Class	73.13%	**72.05 %**
(iii) 16th Note + Melody Range	73.10%	69.50 %
(iv) 16th Note + Pitch Class	74.02%	70.67 %

3) Choice of Data Representations

This experiment discusses how to represent data to the network to achieve the best performance. To see how different resolutions of the melody affects the chord prediction result, we evaluated the performance of the system using different frame lengths. "8th Note" or "16th Note" indicates the melodies and accompaniments were sampled every eighth note or sixteenth note. To see how melodies are represented to the network affects the performance, we represented the input to the network using only the actual pitch range that melody notes are played in, which is MIDI note 50 (D3) to 95 (B6) (Groups i and iii - "Melody Range"), and using pitch class representation (Groups ii and iv - "Pitch Class").

Since the network learns the input melody as a sequence in time and has no access to information other than pitches, we added Beat-On flags to a frame when it is on a beat according to the time signature in meta-data in MIDI files. We also added Note-Begin flags to differentiate two consecutive notes of the same pitch from two distinctive notes. Representing the melodies with their pitch-class number at every 8th note (Group ii) could correctly predict the missing chords approximately 72% of the time when both Note-Begin and Beat-On information are available, and significantly outperforms other representations. Table II shows the result.

4) Comparison with Other Approaches

We compared the architecture used in this paper with four other neural network architectures: Unidirectional LSTM, Bidirectional recurrent neural network (BRNN), Unidirectional recurrent neural network (RNN), and Multi-layer perceptron network (MLP). Neurons in BRNN, RNN and MLP networks were sigmoid neurons. The size of the hidden layers were selected so that the number of weights are approximately the same (around 32,000) for all of the networks as in [35]

Table III shows the classification accuracy and the number of epochs required to converge. All groups were sampled at every eighth note, and were provided with both metric information, (Note-On and Beat-On), during training and testing. Using approximately same number of weights, BLSTM performs significantly better than other neural networks and also converges the fastest.

TABLE III
CLASSIFICATION ACCURACY OF THE DATASET USING DIFFERENT NEURAL NETWORK ARCHITECTURES.

Network	Training Set	Test Set	Epochs
BLSTM	**75.76%**	**71.13 %**	103
LSTM	71.51%	67.57 %	130
BRNN	68.77%	68.86 %	136
RNN	68.33%	66.58 %	158
MLP	55.16%	54.66 %	120

B. Finding the Missing Part in Four-Part SATB Textures

1) Dataset

We evaluated our approach using 378 of J. S. Bach's four-part chorales acquired from [45]. MIDI files were all multi-tracked, one voice on each track. The average length of the pieces is approximate 45 seconds, the maximum and minimum being 6 minutes to 17 seconds, respectively. Among all chorales, 102 pieces are in minor mode. All of the chorales were transposed to C major/A minor using Krumhansl-Schmuckler key-finding algorithm [43]. As in section IV-A, 60% of the files were used as training set, 20% as test set, and 20% as validation set, resulting in 226, 76, 76 pieces respectively.

2) Predicting Missing Voice Given the Other Three Voices

Table IV shows the frame-wise classification accuracy of the predicted missing voices (Soprano, Alto, Tenor, or Bass) when the three other voices are given on training and test sets. The accuracy of predicting missing voices on the original non-transposed set is also listed for comparison. All songs were sampled at every eighth note. From the table, we can observe a few interesting phenomena. First, transposing the songs remarkably improves prediction accuracy in both training and test set. This is not surprising since transposing songs in advance reduces complexity. The same pre-processing is also used by [15] [34] [12]. Second, we see that the network could correctly predict Soprano, Alto, and Tenor approximately 70% of the time when the songs were transposed. Specifically, Alto seems to be the easiest to predict, while Bass is the most difficult.

3) Comparison with Other Approaches

Similar to our approach in Section IV-A4, the size of the hidden layers were selected so that the number of weights are approximately the same (around 63,000) for all of the networks. Table V shows the classification accuracy of the missing voices (either Soprano, Alto, Tenor, or Bass) when all of the three other voices are present.

TABLE IV
CLASSIFICATION ACURRASY OF THE PREDICTED MISSING VOICES, EITHER SOPRANO, ALTO, TENOR, OR BASS, WHEN THE THREE OTHER VOICES ARE GIVEN ON TRAINING AND TESTING SETS.

	Soparno		Alto	
	Training	Test	Training	Test
Not Transposed	69.15%	46.82%	63.61%	47.61%
Transposed	77.90%	71.52%	82.65%	**73.90%**

	Tenor		Bass	
	Training	Test	Training	Test
Not Transposed	47.25%	39.85%	45.40%	36.93%
Transposed	78.47%	69.76%	70.09%	61.22%

TABLE V
CLASSIFICATION ACCURACY OF THE PREDICTED MISSING VOICES WHEN THREE OTHER VOICES ARE GIVEN USING DIFFERENT NETWORK ARCHITECTURE.

	Soparno		Alto	
	Training	Test	Training	Test
BLSTM	84.88%	73.86 %	82.65%	73.90 %
BRNN	90.25%	**74.37%**	85.37%	**74.30 %**
LSTM	85.27%	70.39%	77.14%	70.45%
RNN	81.90%	72.29%	80.31%	71.73%
MLP	68.74%	66.54%	73.51%	70.03%

	Tenor		Bass	
	Training	Test	Training	Test
BLSTM	78.47%	69.76%	70.09%	61.22 %
BRNN	80.95%	**70.13%**	74.58%	**63.74%**
LSTM	73.84%	64.89%	65.86%	57.69%
RNN	75.48%	67.20%	69.68%	59.69%
MLP	68.85%	65.68%	58.58%	56.14%

From the result, we can see that BLSTM performs as well as BRNN on Soprano, Alto, and Tenor parts and significantly outperforms other neural-network based methods on all parts. It also shows that including future information by using bidirectional connection effectively improves accuracy by 3% on average no matter using LSTM cells (in BLSTM and LSTM) or logistic cells (in BRNN and RNN).

V. CONCLUSION

This paper has presented an approach to predicting missing music components for complex multipart musical textures using Bidirectional Long-Short Term Memory (BLSTM) neural networks. We demonstrated the flexibility and robustness of the system by applying the method to two distinctive but popular tasks in the computer-music field: generating chord accompaniment for given melodies in two-part MA textures and generating the missing voice in four-part SATB textures. The proposed approach is capable of handling music pieces of arbitrary length as well as various styles. In addition, the network could be used to generate missing music components of different forms, *i.e.* single notes for four-part SATB textures or chords for two-part MA textures, by simply altering the number of input and output neurons.

Two sets of experiments were conducted regarding the two tasks on two datasets of completely different styles, and issues that influence prediction accuracies were discussed. For the task of predicting chord accompaniment in two-part MA texture, the experimental results showed that the BLSTM network could correctly generate chords for given melodies 72% of the time, which is significantly higher than 68%, the best accuracy achieved by using other neural network based approaches.

We also discovered that representing the melodies using their pitch class profile yielded the best result. As for the problem of finding the missing voice in four-part SATB textures, our experiment demonstrated that a BLSTM network could correctly predict the missing voice approximately 70% of the time on average when three other voices are present. Putting the two experimental results together, BLTSM significantly outperforms all other neural-network based networks for two-part MA textures and performs as well as BRNN for four-part SATB textures showing that the BLSTM network is the optimal structure for predicting missing components from multi-part musical textures.

For future work, we will look into ways to improve the prediction accuracy. The transposition stage in this project can be improved by replacing Krumhansl-Schmuckler key-finding algorithm with other state-of-the-art methods. We may also look into alternatives to transposition such as encoding the inputs using distributed encodings or using intervals among the parts rather than their absolute pitches. In addition, all the system parameters are currently configured to maximize the results on the dataset used in this project. In the future, we will add pre-training to the network to find the optimal features based on music quantities, such as determining the optimal sampling rate according to the amount of activities in textures. For post-processing, we would like to retrain the network with various time delays to learn dependency among music components at different time lags. The capability of using Markov Models to refine results will also be explored by modeling other music quantities such as rhythm and transitions among individual notes. More data in different keys will also be gathered, and the Markov Model's ability of handling music in different keys will be further investigated by applying the Markov Model on each key individually. Finally, the proposed approach could be developed into an interactive system to aid song composition and arrangement. The capability of the network will be further investigated with music prediction tasks of other kinds of music and textures such as predicting the melody of one instrument in songs that involve multiple instruments.

REFERENCES

[1] F. Pachet and P. Roy, "Musical harmonization with constraints: A survey," *Constraints*, vol. 6, no. 1, pp. 7–19, 2001.

[2] H. V. Koops, J. P. Magalhaes, and W. B. De Haas, "A functional approach to automatic melody harmonisation," in *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling & design*. ACM, 2013, pp. 47–58.

[3] M. I. Bellgard and C.-P. Tsang, "Harmonizing music the Boltzmann way," *Connection Science*, vol. 6, no. 2-3, pp. 281–297, 1994.

[4] L. Steels, *Learning the craft of musical composition*. Ann Arbor, MI: MPublishing, University of Michigan Library, 1986.

[5] K. Ebciog ̆lu, "An expert system for harmonizing four-part chorales," *Computer Music Journal*, pp. 43–51, 1988.

[6] F. Pachet and P. Roy, "Mixing constraints and objects: A case study in automatic harmonization," in *Proceedings of TOOLS Europe*, vol. 95. Citeseer, 1995, pp. 119–126.

[7] R. Ramirez and J. Peralta, "A constraint-based melody harmonizer," in *Proceedings of the Workshop on Constraints for Artistic Applications (ECAI98)*, 1998.

[8] R. A. McIntyre, "Bach in a box: The evolution of four part baroque harmony using the genetic algorithm," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*. IEEE, 1994, pp. 852–857.

[9] M. A. Hall, "Selection of attributes for modeling Bach chorales by a genetic algorithm," in *Artificial Neural Networks and Expert Systems, 1995. Proceedings., Second New Zealand International Two-Stream Conference on*. IEEE, 1995, pp. 182–185.

[10] A. Freitas and F. Guimaraes, "Melody harmonization in evolutionary music using multiobjective genetic algorithms," in *Proceedings of the Sound and Music Computing Conference*, 2011.

[11] H.-R. Lee and J.-S. Jang, "i-ring: A system for humming transcription and chord generation," in *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on*, vol. 2. IEEE, 2004, pp. 1031–1034.

[12] I. Simon, D. Morris, and S. Basu, "Mysong: automatic accompaniment generation for vocal melodies," in *Proceedings of the SIGCHI Con- ference on Human Factors in Computing Systems*. ACM, 2008, pp. 725–734.

[13] M. Allan, "Harmonising chorales in the style of Johann Sebastian Bach," *Master's Thesis, School of Informatics, University of Edinburgh*, 2002.

[14] M. Allan and C. K. Williams, "Harmonising chorales by probabilistic inference," *Advances in neural information processing systems*, vol. 17, pp. 25–32, 2005.

[15] K. M. Biyikoglu, "A markov model for chorale harmonization," in *Preceedings of the 5 th Triennial ESCOM Conference*, 2003, pp. 81–84.

[16] S. Suzuki, T. Kitahara, and N. University, "Four-part harmonization using probabilistic models: Comparison of models with and without chord nodes," *Stockholm, Sweden*, pp. 628–633, 2013.

[17] S. A. Raczyński, S. Fukayama, and E. Vincent, "Melody harmonization with interpolated probabilistic models," *Journal of New Music Research*, vol. 42, no. 3, pp. 223–235, 2013.

[18] J.-F. Paiement, D. Eck, and S. Bengio, "Probabilistic melodic harmonization," in *Advances in Artificial Intelligence*. Springer, 2006, pp. 218–229.

[19] C.-H. Chuan and E. Chew, "A hybrid system for automatic generation of style-specific accompaniment," in *4th Intl Joint Workshop on Computational Creativity*, 2007.

[20] C.-H. Chuan, "A comparison of statistical and rule-based models for style-specific harmonization." in *ISMIR*, 2011, pp. 221–226.

[21] D. Temperley and D. Sleator, "The temperley-sleator harmonic analyzer," 1996.

[22] D. Gang and D. Lehmann, "An artificial neural net for harmonizing melodies," *Proceedings of the International Computer Music Association*, 1995.

[23] D. Gang, D. Lehmann, and N. Wagner, "Harmonizing melodies in real-time: the connectionist approach," in *Proceedings of the International Computer Music Association, Thessaloniki, Greece*, 1997.

[24] D. Gang, D. Lehman, and N. Wagner, "Tuning a neural network for harmonizing melodies in real-time," in *Proceedings of the International Computer Music Conference, Ann Arbor, Michigan*, 1998.

[25] U. S. Cunha and G. Ramalho, "An intelligent hybrid model for chord prediction," *Organised Sound*, vol. 4, no. 02, pp. 115–119, 1999.

[26] A. K. Hoover, P. A. Szerlip, and K. O. Stanley, "Generating musical accompaniment through functional scaffolding," in *Proceedings of the Eighth Sound and Music Computing Conference (SMC 2011)*, 2011.

[27] A. K. Hoover, P. A. Szerlip, M. E. Norton, T. A. Brindle, Z. Merritt, and K. O. Stanley, "Generating a complete multipart musical composition from a single monophonic melody with functional scaffolding," in *International Conference on Computational Creativity*, 2012, p. 111.

[28] J. Feulner, "Neural networks that learn and reproduce various styles of harmonization," in *Proceedings of the International Computer Music Conference*. International Computer Music Association, 1993, pp. 236–236.

[29] R. DePrisco, A. Eletto, A. Torre, and R. Zaccagnino, "A neural network for bass functional harmonization," in *Applications of Evolutionary Computation*. Springer, 2010, pp. 351–360.

[30] H. Goksu, P. Pigg, and V. Dixit, "Music composition using genetic algorithms (ga) and multilayer perceptrons (mlp)," in *Advances in Natural Computation*. Springer, 2005, pp. 1242–1250.

[31] P. M. Todd, "A connectionist approach to algorithmic composition," *Computer Music Journal*, pp. 27–43, 1989.

[32] M. C. Mozer, "Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing," *Connection Science*, vol. 6, no. 2-3, pp. 247–280, 1994.

[33] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Audio chord recognition with recurrent neural networks." in *ISMIR*, 2013, pp. 335–340.

[34] ——,"Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," *arXiv preprint arXiv:1206.6392*, 2012.

[35] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[36] F. Eyben, S. Bo ̈ck, B. Schuller, and A. Graves, "Universal onset detection with bidirectional long short-term memory neural networks." in *ISMIR*, 2010, pp. 589–594.

[37] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *Signal Processing, IEEE Transactions on*, vol. 45, no. 11, pp. 2673–2681, 1997.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[39] P. J. Werbos, "Back-propagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[40] A. Robinson and F. Fallside, *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering,1987.

[41] J. Schmidhuber, "Long short-term memory: Tutorial on lstm recurrent networks," http://people.idsia.ch/~juergen/lstm/ index.htm, 2003, online resource.

[42] E. Foxley, "Nottingham dataset," http://ifdo.ca/~sey-mour/nottingham/ nottingham.html, 2011, accessed: 04-19-2015.

[43] D. Temperley, *The Cognition of Basic Musical Structures*. Cambridge: MIT Press, 2001.

[44] M. S. Cuthbert and C. Ariza, "Music21: A toolkit for computer-aided musicology and symbolic music data," 2010.

[45] M. Greentree. (1996) http://www.jsbchorales.net/index.shtml. Accessed: 04-19-2015.