

Hierarchical Recurrent Neural Networks for Conditional Melody Generation with Long-term Structure

Guo Zixun
Information Systems,
Technology, and Design
Singapore University
of Technology and Design
Singapore
nicolas_guo@sutd.edu.sg

Dimos Makris
Information Systems,
Technology, and Design
Singapore University
of Technology and Design
Singapore
dimosthenis_makris@sutd.edu.sg

Dorien Herremans
Information Systems,
Technology, and Design
Singapore University
of Technology and Design
Singapore
dorien_herremans@sutd.edu.sg

Abstract—The rise of deep learning technologies has quickly advanced many fields, including that of generative music systems. There exist a number of systems that allow for the generation of good sounding short snippets, yet, these generated snippets often lack an overarching, longer-term structure. In this work, we propose CM-HRNN: a conditional melody generation model based on a hierarchical recurrent neural network. This model allows us to generate melodies with long-term structures based on given chord accompaniments. We also propose a novel, concise event-based representation to encode musical lead sheets while retaining the notes’ relative position within the bar with respect to the musical meter. With this new data representation, the proposed architecture can simultaneously model the rhythmic, as well as the pitch structures in an effective way. Melodies generated by the proposed model were extensively evaluated in quantitative experiments as well as a user study to ensure the musical quality of the output as well as to evaluate if they contain repeating patterns. We also compared the system with the state-of-the-art AttentionRNN [1]. This comparison shows that melodies generated by CM-HRNN contain more repeated patterns (i.e., higher compression ratio) and a lower tonal tension (i.e., more tonally concise). Results from our listening test indicate that CM-HRNN outperforms AttentionRNN in terms of long-term structure and overall rating.

Index Terms—Hierarchical RNN, Recurrent neural network, RNN, Generative model, Conditional model, Music generation, Event-based representation, Structure

I. INTRODUCTION

At the dawn of computing, the idea of generating music was first conceived by Lady Ada Lovelace, when she said: ‘[The Engine’s] operating mechanism might act upon other things besides numbers [...] Supposing, for instance, that the fundamental relations of pitched sounds in the signs of harmony and of musical composition were susceptible of such expressions and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent.’ Ada Lovelace, as quoted in [2]. When computers were first created, it was not long until they were used to

generate the first piece of music [3]. The field of computer generated music has grown ever since [4], with great strides being made in recent years due to deep learning technologies [5]. Computer generated music, however, is not yet a part of our daily music listening experience. One potential reason could be the lack of repeating patterns or themes, i.e. a longer-term structure, which is a necessity for the presence of ‘earworms’ in music [6]. There are two very different types of music generation systems: those that generate symbolic music and those that generate raw audio. Some attempts have been made to generate music directly as raw audio [7], [8]. This remains a challenge, however, due to the disproportionate amount of audio samples in the overall musical structure. Hence, much of the existing music generation research focuses on the symbolic domain where music is represented as a series of musical events in sequence. Popular data encoding schemes for symbolic music include piano-roll representation [9], [10], tonnetz representation [11], word embeddings [12], [13], and event-based representation [14]. In this work, we will focus on the symbolic approach, more specifically, we will generate melodies from their respective chord accompaniments using a novel event-based representation. Compared to the existing event-based representations, we explicitly encode bar event information in our representations. This allows our model to understand musical complexities such as meter [15].

Composing melodies from a given set of chords is a task faced by many musicians in the real-world, e.g. in pop music composition or jazz improvisation. We are thus motivated to design our model to receive a chord sequence as conditioning input by the users and generate melodies based on these provided chords. In this way, the users will have a certain level of control over the generated melody through the manipulation of the input chord sequence.

Since music data is sequential, researchers have recently focused on developing recurrent neural networks (RNN) and their variants (e.g., long-short term memory (LSTM) / gated recurrent units (GRU)) for music generation due to their

memory mechanism [5]. Still, current computer generated music often lacks long term structure, an essential quality of polished, complete musical pieces with recurring themes [4], [16], [17]. Even if it is speculated by researchers that the generated music may include long-term structure, very rarely a quantitative measure is employed to measure this [11]. By injecting a hierarchical structure between the RNN layers of our proposed model, we allow the model to capture musical patterns on different time scales. In this work, we propose CM-HRNN, a conditional melody generation model based on a hierarchical recurrent neural network. In Section III and IV, we describe the proposed event representation and CM-HRNN architecture in detail. We then thoroughly analyze the music generated by CM-HRNN in terms of musical quality and the presence of repeated patterns in Section V. The latter is done by calculating the compression ratio [11] measure. Finally, these results are confirmed in a listening study. In the next section, we first give a brief overview of the related work.

II. RELATED WORK

We will start by giving an overview of existing music generation systems with a focus on long-term structure. Then we will dive into hierarchical models which have shown to be effective in capturing long-term dependencies in multiple fields such as natural language processing [18], audio generation [7], [19], and music generation [20], [21].

A. Music generation systems with long-term structure

The problem of generating music with long-term structure has received limited attention. The affective music generation system MorpheuS [22] enforces repeated patterns and structure into music by considering music generation as a combinatorial optimization problem. MorpheuS uses repeated patterns from existing templates as hard constraints during generation. Looking towards machine learning methods, [16] use a Markov model that learns the statistical properties of a musical corpus, but combine this with a variable neighborhood search optimization algorithm to generate music, while hard constraining a larger structure. Similarly, [23] train a convolutional restricted Boltzmann machine (C-RBM), but use simulated annealing as a sampling technique. This allows them to include structural constraints.

Looking at ‘pure’ neural networks, that do not leverage (often) slower optimization techniques, we find [11], who use a novel Tonnetz representation to train an LSTM that is better able to generate polyphonic music with repeated patterns than a similar network with a more traditional piano roll representation. Two recent RNN-based systems, LookbackRNN and AttentionRNN [1], generate monophonic melodies with long term structure, in an autoaggressive way. Both models incorporate the “repetitive label” in the data representation, a way to represent repetition of notes in the neighbouring two bars. LookbackRNN [1] contains a two-layer LSTM network with residual connections between the different time steps, which allows the model to feed note events from the previous 1 to 2 bars to the current time step. The residual connection is able to

offset the impact of vanishing gradients and allows the model to generate more repetitive patterns. Similar to LookbackRNN, AttentionRNN [1] also inputs previous event information into the current generation step, however, it applies a learnable attention mask to the previous n events which will decide how much attention to put on each previous event. By explicitly feeding the current generation step the neighbouring events, the model will be able to learn when recent musical repetition occurs. Each generation step, however, may not be aware of “the bigger picture”.

Another model is the MusicVAE by [21], which generates (smooth) transitions between two musical fragments with overall long-term structures. In their variational autoencoder, the input music data is first encoded by a Bidirectional (Bi-) RNN encoder which generates a latent vector z . The latent vector z will then be fed as a initial state to an RNN which generates a series of “conductors” which will then “conduct” the hierarchical RNN decoder to generate music sequences. Since each “conductor” conditions the generation of multiple steps, the generated music is shown to contains more long-term structures compared to other existing recurrent variational autoencoders (VAEs). Even though the problem domain is different, we can still gain inspiration from the conditioning “conductors” and the hierarchical decoders which would enhance the overall long-term structure in music.

Hierarchical RNN (HRNN) [20] aims to generate monophonic melodies with long term structure. It first generates bar profiles and beat profiles which indicate the rhythmic pattern within a bar and a beat using vanilla LSTMs. Beat profile generation is directly conditioned by the latent outputs from the upper RNN layer which generates the bar profile. These two profiles then jointly condition the pitch generation. By combining the generated pitch and rhythmic pattern, the generated monophonic melodies are qualitatively evaluated through multiple listening tests and are shown to outperform the generated outputs by the LookbackRNN [1]. The core design of the architecture is to first generate the rhythmic patterns and condition the pitch generation with both coarse and fine rhythmic patterns afterwards.

In this paper, we propose an architecture similar to HRNN but with different design motivations. We believe that rhythmic and pitch features are equally important and hierarchical structures should exist in both of these domains. Hence, instead of conditioning the pitch generation with the generated rhythmic patterns from vanilla LSTMs, we apply the hierarchical RNN structure to both rhythmic and pitch latent spaces simultaneously. Compared to HRNN, our proposed CM-HRNN is able to condition the melody generation with provided chords. Moreover, the musical quality of HRNN-generated music is only evaluated through subjective listening test. In contrast, we evaluate our proposed CM-HRNN with extensive analytical measures as well as a user study.

B. Hierarchical architectures for long term dependencies

Drawing inspiration from other fields in which long-term dependencies of sequential data have been modelled by RNNs,

we find that multiple hierarchical architectures have been proposed for this challenge [18], [19], [24]–[26]. These models try to control the weight update rate within the RNN cells. RNN weights matrices are separated into different chunks and will be updated using different rates: high update rates will capture the short-term dependencies whereas low update rates will capture long-term dependencies.

These architectures can be applied to the domain of audio generation. For instance, SampleRNN [7] is able to synthesise realistic sounding audio with a hierarchical RNN structure. Instead of manipulating the RNN weights update rate to capture the temporal and long-term structure of sequences of audio samples, it uses multiple stacks of RNN with upper tier RNN layers operating on more grouped audio samples per step and lower tier RNN layers operating on fewer grouped samples per step. Inspired by SampleRNN, we group our data (i.e., music events) in different resolutions and process these groups of data separately to obtain the coarse-to-fine hierarchical features of the data.

III. EVENT-BASED REPRESENTATION WITH UNDERSTANDING OF METER

We propose a novel data encoding scheme based on [27] which uses two combined one-hot encoded sub-vectors to represent the pitch and duration of music events. Additionally, we extend their representation with another three sub-vectors. As a result, each music event vector consists of the following sub-vectors:

- **Pitch:** The MIDI standard defines 128 pitches. We add two additional elements to this sub-vector: one to indicate a rest; and one to indicate if a note is sustained or not (i.e., tied note). This results in a 130-dimensional sub-vector.
- **Duration:** A 16-dimensional vector which represents duration. Possible values range from a 16th note to a whole note with an increment of 16th note duration (i.e. smallest quantization unit).
- **Current and Next Chord:** There are 12 possible chord roots (from C to B). For each root we set 4 possible chord types (i.e. major, minor, diminished, and dominant 7th) plus the rest symbol. This results in two 49-dimensional vectors. Since the chord information is used to guide the melody generation, both the current chord as well as the next chord are provided as input to the network at each given time.
- **Bar:** A two-dimensional vector for indicating if it is the start of a bar or not.

An example of these sub-vectors can be found in Figure 1. All sub-vectors are concatenated together while the inclusion of multiple types of data, or viewpoints, such as bar information is inspired by [28]. Finally, an example of illustrating the proposed lead sheet encoding is shown in Figure 2.

Since bar lines are indicated in the proposed notation, we can calculate the accumulated time information for each event within the bar as shown in Table I which will then be fed to our proposed model in the bottom generation tier. Explicitly feeding the model the accumulated time information is proved

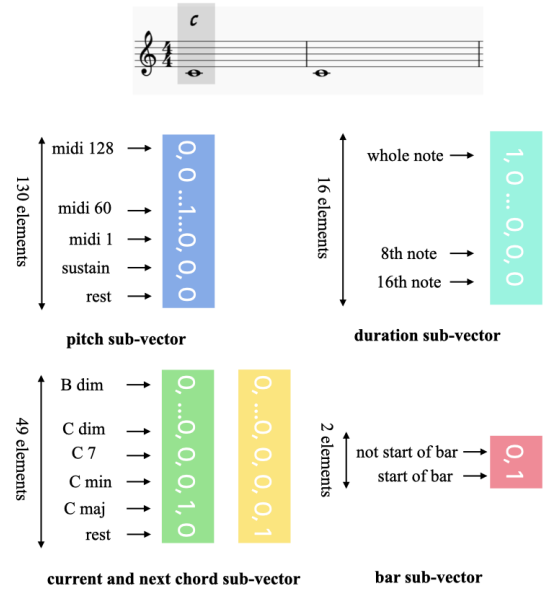
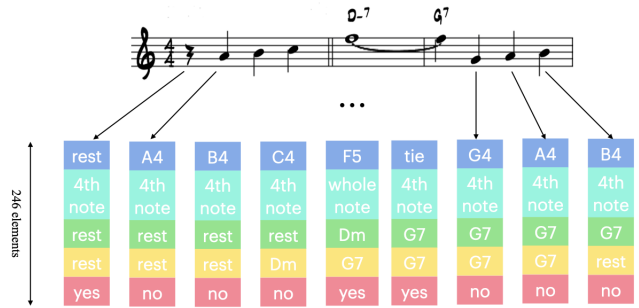


Fig. 1: Example of the sub-vectors for the starting C note.

to be effective in terms of duration and bar event prediction in Section V.



Each color-coded block denotes a one-hot vector. For illustration purposes, only the event symbol is listed in the color-coded block. Readers can refer to the sub-vector with the same colour in Figure 1 for the corresponding one-hot vector representation.

Fig. 2: Proposed event-based representation for the first three bars of “autumn leaves”.

TABLE I: Accumulated time information for the first three bars of “autumn leaves”.

Event number	1	2	3	4	5	6	7	8	9
Start of bar	yes	no	no	no	yes	yes	no	no	no
Note duration	1	1	1	1	4	1	1	1	1
<i>acc_t</i>	1	2	3	4	4	1	2	3	4

Duration is expressed in units of one quarter note here. “acc_t” is accumulated time.

There are several advantages to this representation. Firstly, pitch repetition and duration repetition are mutually inclusive (i.e. they can co-exist or occur separately) [29]. For example, a melodic motif may be repeated several times in a jazz piece with slight rhythmic variations. In other cases, repetitive patterns may exist in the duration vector space but not

in the pitch vector space. Hence, using multi-hot encoded data enables RNNs to learn different high-level features and patterns separately. Secondly, this representation can encode data efficiently. Compared to the piano-roll representation, our proposed encoding can easily represent two consecutive notes with the same pitch whereas one may not be able to differentiate between a long note or two repeated notes in the piano-roll representation [10] unless additional onset information is added to capture the repetitive onset which could potentially increase the computational cost. Moreover, given that the smallest duration value used in the quantization step is a 16th note in this research, a piano roll representation would need 32 event vectors to represent two tied whole notes, whereas our proposed representation only needs 2 event vectors: [whole note with a duration of 4 beats, tied note with a duration of 4 beats]. Thirdly, it allows us to incorporate an indication of bar events, which is crucial to musical composition and will allow the system to learn the relative positioning within bars. Last but not least, the proposed data representation could easily be extended to include other features (e.g., velocity). This is a major difference between our work and [27].

IV. PROPOSED MODEL: CM-HRNN

A. Model Architecture

Our proposed CM-HRNN architecture consists of multiple hierarchical tiers (see Figure 3). All tiers except for the bottom tier use LSTM cells to process grouped event data. Upper tiers process more grouped event data per step to capture the long-term dependencies. The latent outputs from the upper tiers, which represent a coarser latent representation of sequential data directly condition the lower tiers' generation after proper upsampling. The bottom tier, after receiving fine-to-coarse conditioning vectors from the upper tiers, proceeds to predict future music events. Inspired by [7], the CM-HRNN's bottom tier uses a conv1d operation to process overlapping sliding windows of n_{tier_1} events (1 indicates the bottom tier) and generate the next event. To convert latent outputs from these sliding windows to music events, we include a predictive network in the bottom tier (see Figure 4). In this paper, we focus on two variants of the proposed architecture: a 2-tier CM-HRNN and a 3-tier CM-HRNN. In theory, we could incorporate as many tiers as possible, but to avoid overly increasing the model's complexity, we will restrict the number of tiers to a maximum of three. The architecture of the proposed 3-tier CM-HRNN is shown in Figure 3. By removing the top tier of 3-tier CM-HRNN, we obtain the architecture of 2-tier CM-HRNN.

In the 3-tier CM-HRNN, residual connections exist between the top tier and the bottom tier to alleviate the effect of the vanishing gradient. This also allows the bottom generation tier to receive the latent conditioning outputs from all of the upper tiers directly. Additionally, we also feed the accumulated time information to the predictive network in the bottom generation tier. The accumulated time vector has the same representation as the duration vector (see Section III, Table I). Given that the position of each bar line is indicated in our proposed

data representation, the model is able to quickly learn the relationship between the accumulated time information, the duration vector, and the bar event vector, which makes the model more aware of relative positioning information.

In the upper tiers of the model, events are grouped into non-overlapping event frames. Each such frame contains FS^k events. Whereas in the bottom tier, events are grouped into overlapping event frames with a stride of 1 (i.e., sliding windows). Here, k indicates the tier number and FS^k represents the frame size of tier k . These event frames will be processed by their respective tier in order. The relationship between each FS^k and tier is as follows:

$$FS^{k=1} = FS^{k=2} \quad (1)$$

$$FS^{k+1} \bmod FS^k = 0 \quad (2)$$

$$FS^{k+1} > FS^k \quad \text{if } k > 1 \quad (3)$$

We group events into event frames to form the input for the different tiers. Here, e_t indicates the t -th event. All e_t s within one square bracket are grouped into one event frame.

$$frames_k = \begin{cases} [e_1 \dots e_{FS^k}], [e_{FS^k+1} \dots e_{2 \cdot FS^k}] \dots, & \text{if } k \neq 1 \\ [e_1 \dots e_{FS^k}], [e_2 \dots e_{FS^k+1}] \dots, & \text{if } k = 1 \end{cases} \quad (4)$$

We define the input for each tier's processing unit (2-layer LSTM or conv1d) i_k as follows:

$$i_k = \begin{cases} frames_k, & \text{if } k = \text{top tier or } 1 \\ W_f frames_k + W_{o_i} \mathbf{o}_{k+1}, & \text{otherwise} \end{cases} \quad (5)$$

Different tiers receive input at different rates, hence, the outputs from the upper tier LSTMs need to be upsampled before they are used to as a condition in the lower tier generation. In the formula below, h indicates the hidden states of the LSTM; o_{t_k} indicates the intermediate outputs for tier k at the t -th time step; \mathbf{o}_{t_k} represents the upsampled LSTM outputs, and acc_t the accumulated time information at time step t , \oplus is used for vector concatenation, and all W are trainable weights. Bias terms and repeated layers are omitted in the equation for simplicity. Residual connections between tiers are implemented as per Eq.8.

$$o_{t_k}, h_{t_k} = \mathbf{LSTM}(i_{t_k}, h_{t-1_k}), \quad \text{if } k \neq 1 \quad (6)$$

$$\mathbf{o}_{t_k} = W_{\text{upsample}} o_{t_k}, \quad \text{if } k \neq \text{top tier or } 1 \quad (7)$$

$$o_{t_1} = (W_i i_{t_1} + \sum_{k=2}^{k_{top}} \mathbf{o}_{t_k}) \oplus acc_t \quad (8)$$

To predict the next event at the $t + 1$ -th time step, we input the intermediate output o_{t_1} to a predictive network as per Fig. 4. p_t , d_t , and b_t represent the pitch, duration, and bar sub-vectors of an event respectively. Eq.8 to Eq.12 represent the predictive network, whereby FC can consist of multiple

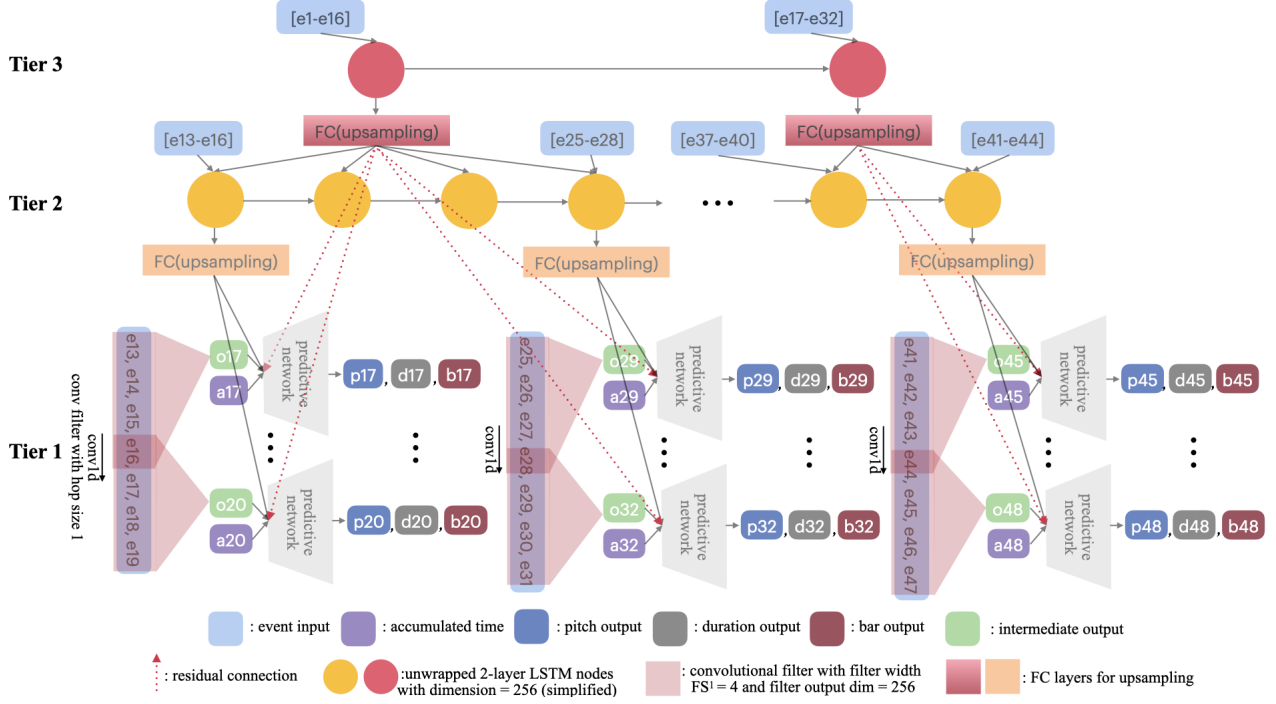


Fig. 3: Our proposed CM-HRNN architecture.

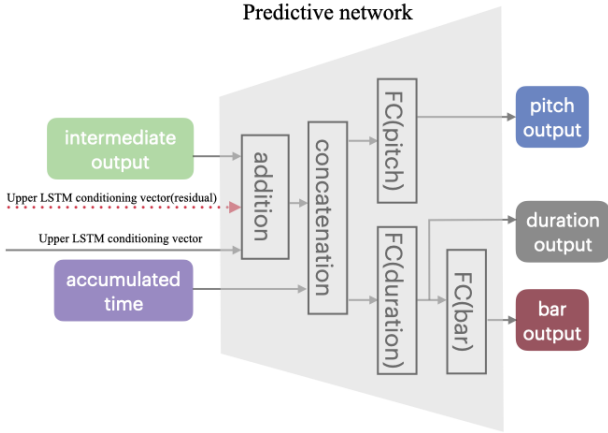


Fig. 4: Details of the predictive network within CM-HRNN. FC(pitch) represents 2 fully connected layers (130 nodes each). FC(duration) represents 2 fully connected layers (16 nodes each). Finally, FC(bar) is a 1-layer fully connected network. All FCs have ReLu activation.

layers (as per Fig. 4) and has ReLu activation. For simplicity, we did not include activation information in the equations. We then apply three softmax functions to obtain the probability

distribution for each of these three sub-vectors.

$$p_t = FC_p(o_{t_1}) \quad (9)$$

$$d_t = FC_d(o_{t_1}) \quad (10)$$

$$b_t = FC_b(d_t) \quad (11)$$

$$pitch_t \sim Softmax(p_t) \quad (12)$$

$$duration_t \sim Softmax(d_t) \quad (13)$$

$$bar_t \sim Softmax(b_t) \quad (14)$$

We calculate the cross-entropy loss for each of the three vectors (pitch, duration, and bar), and summarise them to form the final training objective function. Different weights α_1 , α_2 , and α_3 are applied to pitch loss, duration loss, and bar loss respectively:

$$\begin{aligned} Loss = & \alpha_1 \cdot CE(pitch_t, \hat{pitch}_t) \\ & + \alpha_2 \cdot CE(duration_t, \hat{duration}_t) \\ & + \alpha_3 \cdot CE(bar_t, \hat{bar}_t) \end{aligned} \quad (15)$$

B. Implementation details

We have implemented several models which will be evaluated in Section V. All models utilise a two-layer LSTM with 256 nodes as the memory unit. The sliding window in the bottom tier is implemented as a conv1d operation with a stride of 1. The filter for the convolution has a filter size of FS^1 (varies per experiment) and an output dimension of 256. In the predictive network, we use 2 fully connected (FC) layers (130 nodes each) for pitch prediction; 2 FC layers (16 nodes each) for duration prediction; and 1 FC layer (2 nodes each) for bar prediction. All FC layers are equipped with ReLu activation.

We set $\alpha_1 = 0.4$, $\alpha_2 = 0.3$, and $\alpha_3 = 0.3$ for the loss function. To generate music for our experiments, we provide the first 16 pitch events and all chord events of all songs in the test set as the model input. The sampling temperatures for the pitch and duration and bar output distribution are set to be 0.7, 0.2, and 0.1 respectively. The duration sampling temperature is set low whereas the pitch sampling temperature is set high because we want to keep the timing correct while giving more freedom to pitch generation.

V. EXPERIMENTS

A. Experimental setup

We set up several experiments to determine the optimal network architecture, as well as the model’s ability to generate high-quality music with structure, and to compare it with a state-of-the-art system, AttentionRNN.

In a first experiment, we test whether accumulated time information (acc_t) could help the model to generate bar events and thus understanding the meter. Hence, we have implemented 2 sets of models: with or without the accumulated time information (acc_t) fed as input. Next, we test the influence of the residual connection and the frame size FS^2 in the 3-tier CM-HRNN, on the model’s ability to include repeated patterns in the generated music, as well as keep track of the musical meter. Finally, the best 2- and 3-tier CM-HRNN was compared to AttentionRNN, both analytically, as well as through a listening test. While it is not always easy to compare different music generation systems, given that they often have slightly different functional tasks or input representations, a comparison with AttentionRNN was facilitated through their use of multi-hot encoded input data which is similar to our data representation. Below, we first describe our dataset and the pre-processing method, followed by a description of the evaluation metrics and the listening test setup.

B. Dataset and pre-processing

All training data (XML format) was parsed from Theorytab [30], an open-source website that provides lead sheets. Each XML file contains a label with the song genre, song sections, note events, and chord events with absolute timing. During the data encoding stage, we changed the absolute timing into delta timing due to the nature of the LSTM cells. For simplicity, we have chosen to include only monophonic melodies with a time signature of 4/4, along with their chords. All songs were transposed to C. We have merged all available sections (i.e., intro, verse, pre-chorus, chorus, and outro) of the same song to make the training sequences long enough. We matched each note event with its corresponding chord and labeled the note event sequences with the bar label. In the end, we obtained 5,507 training musical fragments which were then randomly split into training, validation, and test set with the ratio of 0.8, 0.1, and 0.1 respectively.

C. Analytical evaluation measures

1) *Compression ratio to measure long-term structure:* We use COSIATEC [31] as a proxy to evaluate the long term

structures contained in the generated music. COSIATEC utilizes a geometric approach much like zip-file compression, to detect repeated patterns in symbolic music data. The resulting compression ratio indicates how much smaller a musical file can be made by representing it with pattern vectors and their corresponding translation vectors. Hence, this compression ratio reflects the number of repeated patterns that the generated music contains. In the experiments, we only calculate the compression ratio of the generated *melodies*.

2) *Tension measures:* We use the model for tonal tension by [32] based on the spiral array [33], to measure the amount of tension in the generated music. This model offers three measures for each time frame of music (or cloud): cloud diameter, tensile strain, and cloud momentum. Cloud diameter indicates the tonal dissonance within a cloud, tensile strain measures the tonal distance to the key of the song, and cloud momentum calculates the tonality movement between different clouds. Even though a limited amount of dissonant notes could sound more musically interesting, these could also adversely affect the music quality. By calculating these tension measures we have a direct impression of whether the generated music is tonally consistent.

D. Listening test setup

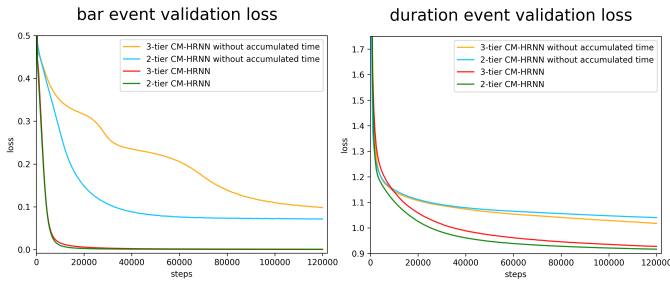
An online listening test was conducted to evaluate the proposed model subjectively. Each participant was asked to listen to 12 musical fragments ranging from 13 to 44 seconds generated by 3 models: 2-tier CM-HRNN, 3-tier CM-HRNN, and AttentionRNN (with an attention size of 32). These snippets of music are placed in random order and the lead sheets were shown during playback. All participants were asked to rate the following questions on a scale from 1 (very poor) to 5 (very good):

- 1) overall perception of musical quality;
- 2) coherence of the music;
- 3) consonance between chord and melody;
- 4) naturalness of the generated melody in terms of pitch;
- 5) naturalness of the generated melody in terms of duration.

VI. RESULTS

A. Ability to learn correct bar timing and effectiveness of the accumulated time information

To evaluate the influence of adding the accumulated time information in the bottom generation tier on the predicted duration of each note, as well as the correct bar placement, we have trained 4 variants of our model (see Table II): a model with 2-tiers and 3-tiers, each with and without the accumulated time information added as input. Intuitively, if the model can accurately predict when a new bar starts, and thus partly understands meter, we hope that it is better able to generate meaningful rhythm. The validation loss for models with and without accumulated time information is shown in Figure 5. It is obvious that by adding the accumulated time information, the models’ validation loss, both when predicting the bar as well as the duration event, is lower than for the models without the accumulated time information.



(a) Bar event validation loss. (b) Duration event validation loss.

Fig. 5: Evaluation of the validation loss of different models, with and without added accumulated time information. Model configurations are shown in Table I

Additionally, we calculate the ratio of successfully predicted bar events among all predicted bars (successful bar ratio), $\frac{\text{no. of bars with 4 beats}}{\text{total no. of bars}}$ from the results generated by each model. We also compare the compression ratios of the pieces generated by each model. The results for each model configuration are shown in Table II. When comparing the successful bar ratios between models with and without added accumulated time information, we see that the former are almost always able to accurately predict when a new bar should start. In other words, they are able to partly understand musical meter. Also, models with accumulated time information achieve a much higher compression ratio compared to those without the accumulated time information which indicates that the former can generate music with more repeated patterns.

TABLE II: Results of models with and without accumulated time information.

Model	FS^2	FS^3	acc_t	SBR	CPR
2-tier CM-HRNN	16	n.a.	no	91.0%	1.61
2-tier CM-HRNN	16	n.a.	yes	96.1%	1.69
3-tier CM-HRNN	2	16	no	66.2%	1.62
3-tier CM-HRNN	2	16	yes	100.0%	1.68

acc_t : accum. time information; SBR: successful bar ratio; CPR: compression ratio

B. Residual connections and FS^2 in 3-tier CM-HRNN

To validate the effectiveness of adding residual connections in the 3-tier CM-HRNN, we have implemented six model variants, some with and without residual connections. To find the optimal FS^2 in a 3-tier model setting, we froze FS^3 to be 16 and experimented with different FS^2 . The model configurations and results are shown in Table III. From the results, given the same FS^2 , it is clear that the residual connection can increase the overall compression ratio of the generated melodies. This is an indication that the generated music might contain more repeated themes and might have a larger overall structure [11]. Meanwhile, the successful bar ratio remains high for all 3-tier models regardless of FS^2 . From these results, we choose the 3-tier model with

$FS^2 = 2, FS^3 = 16$, and with residual connection as our best performing model.

TABLE III: Comparing different FS^2 in a 2-tier model.

FS^2	FS^3	Residual connections	SBR	CPR
2	16	yes	100.0%	1.68
2	16	no	100.0%	1.67
4	16	yes	100.0%	1.64
4	16	no	100.0%	1.63
8	16	yes	99.2%	1.65
8	16	no	98.7%	1.63

SBR: successful bar ratio; CPR: compression ratio

C. Comparison with AttentionRNN

We compare our best performing model with the state-of-the-art model AttentionRNN [1]. This comparison was facilitated by the fact that their input representation is also multi-hot encoded, even though their original representation includes repetitive labels which indicate whether the note event was repeated 1 or 2 bars ago. Secondly, our problem domain is identical: modeling long-term coherence in music generation. Even though other researches may share some similarities in terms of model architecture [20], [21], they work on a different problem domain or with different data representation, thus making comparison hard.

We implemented two AttentionRNN models with the same LSTM setting: 2 layers (each with 256 nodes). One model has an attention lookback size of 16 and the other has an attention lookback size of 32. Other than the compression ratio and successful bar ratio, we also compare the tension of the generated music from all these models. We invite the reader to listen to some generated pieces online¹.

A comparison between the calculated measures for generated pieces by both our proposed CM-HRNN and AttentionRNN is shown in Table IV. From these results, we see that music generated by CM-HRNN has a much higher compression ratio compared to AttentionRNN. The tension values indicate that music generated by AttentionRNN sounds more tense and dissonant, with less clear movements in tonality.

TABLE IV: Analytical measures for both CM-HRNN and AttentionRNN.

Model	Meter		Structure		Tension	
	SBR	CPR	CD	TS	CM	
CM-HRNN(3t)	100.0%	1.68	2.16±0.47	0.54±0.20	0.69±0.21	
CM-HRNN(2t)	96.1%	1.69	2.17±0.47	0.53±0.20	0.70±0.21	
AttentionRNN(16)	88.2%	1.58	2.22±0.47	0.68±0.20	0.52±0.20	
AttentionRNN(32)	90.0%	1.58	2.21±0.47	0.68±0.20	0.52±0.20	

SBR: successful bar ratio; CPR: compression ratio; CD: cloud diameter; TS: tensile strain; CM: cloud momentum

D. Listening test

A total of 41 participants participated in the listening test. The resulting ratings in Table V show that our proposed

¹<https://www.dropbox.com/sh/4bbpqdv7bjqlz0g/AADvhaYdmK7fmznRxCOU6rSga?dl=0>

CM-HRNN outperforms AttentionRNN in terms of overall enjoyment rating and long-term coherence, which again proves the effectiveness of our proposed model in capturing the long-term structure of music data.

TABLE V: Listening test rating results on a scale from 1 to 5.

Model	Overall Rating	Coherence	Consonance	Pitch naturalness	Duration naturalness
CM-HRNN (3t)	3.42	3.32	3.52	3.55	3.39
CM-HRNN (2t)	3.35	3.24	3.49	3.52	3.40
AttentionRNN (32)	2.80	2.85	2.95	2.98	2.79

VII. CONCLUSION

We propose a novel conditional hierarchical RNN network, CM-HRNN to generate melodies conditioned with chords. In addition to using a novel, effective event-based representation that explicitly encodes bar information, CM-HRNN generates musically sound melodies that contain long-term structures. The source code of the CM-HRNN implementation is available online². In extensive experiments, both using calculated features as well as a listening test, we show that pieces generated by CM-HRNN have greater tonal stability and more repeated patterns than those generated by AttentionRNN.

REFERENCES

- [1] E. Waite, "Generating long-term structure in songs and stories," 2016. [Online]. Available: <https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn>
- [2] D. Herremans and K. Sørensen, "Composing fifth species counterpoint music with a variable neighborhood search algorithm," *Expert systems with applications*, vol. 40, no. 16, pp. 6427–6437, 2013.
- [3] L. A. Hiller Jr and L. M. Isaacson, "Musical composition with a high speed digital computer," in *Audio Engineering Society Convention 9*, 1957.
- [4] D. Herremans, C.-H. Chuan, and E. Chew, "A functional taxonomy of music generation systems," *ACM Computing Surveys (CSUR)*, vol. 50, no. 5, pp. 1–30, 2017.
- [5] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, *Deep learning techniques for music generation*. Springer, 2020.
- [6] J. A. Burgoyne, D. Bountouridis, J. van Balen, and H. Honing, "Hooked: a game for discovering what makes music catchy," in *Proc. of the 14th Int. Society for Music Information Retrieval Conf., ISMIR, Curitiba, Brazil, November 4-8, 2013*, pp. 245–250.
- [7] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, "Samplernn: An unconditional end-to-end neural audio generation model," in *Proc. of the 5th Int. Conf. on Learning Representations, ICLR 2017*.
- [8] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, September 2016*, p. 125.
- [9] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," in *Proc. of the 18th Int. Society for Music Information Retrieval Conf., ISMIR, Suzhou, China, October 23-27, 2017*, pp. 324–331.
- [10] D. Eck and J. Schmidhuber, "A first look at music composition using lstm recurrent neural networks," *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, vol. 103, p. 48, 2002.
- [11] C.-H. Chuan and D. Herremans, "Modeling temporal tonal relations in polyphonic music through deep networks with a novel image-based representation," in *Proc. of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, pp. 2159–2166.
- [12] C.-Z. A. Huang, D. Duvenaud, and K. Z. Gajos, "Chordripple: Recommending chords to help novice composers go beyond the ordinary," in *Proc. 21st Int. Conf. on Intelligent User Interfaces*, 2016, pp. 241–250.
- [13] C.-H. Chuan, K. Agres, and D. Herremans, "From context to concept: exploring semantic relationships in music with word2vec," *Neural Computing and Applications*, vol. 32, no. 4, pp. 1023–1036, 2020.
- [14] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: Learning expressive musical performance," *Neural Computing and Applications*, vol. 32, no. 4, pp. 955–967, 2020.
- [15] W. B. De Haas and A. Volk, "Meter detection in symbolic music using inner metric analysis," in *Proc. of the 17th Int. Society for Music Information Retrieval Conf., ISMIR, New York City, United States, August 7-11, 2016*, pp. 441–447.
- [16] D. Herremans, S. Weisser, K. Sørensen, and D. Conklin, "Generating structured music for bagana using quality metrics based on markov models," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7424–7435, 2015.
- [17] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," in *Proc. of the 7th Int. Conf. on Learning Representations, ICLR 2019*.
- [18] J. Chung, S. Ahn, and Y. Bengio, "Hierarchical multiscale recurrent neural networks," in *Proc. of the 5th Int. Conf. on Learning Representations, ICLR 2017*.
- [19] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber, "A clockwork rnn," in *Proc. of the 31th Int. Conf. on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, vol. 32, pp. 1863–1871.
- [20] J. Wu, C. Hu, Y. Wang, X. Hu, and J. Zhu, "A hierarchical recurrent neural network for symbolic melody generation," *IEEE Transactions on Cybernetics*, vol. 50, no. 6, pp. 2749–2757, 2019.
- [21] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *Proc. of the 35th Int. Conf. on Machine Learning, ICML, Stockholm, Sweden, July 10-15, 2018*, vol. 80, pp. 4361–4370.
- [22] D. Herremans and E. Chew, "Morpheus: generating structured music with constrained patterns and tension," *IEEE Transactions on Affective Computing*, vol. 10, no. 4, pp. 510–523, 2017.
- [23] S. Lattner, M. Grachten, G. Widmer *et al.*, "Imposing higher-level structure in polyphonic music generation using convolutional restricted boltzmann machines and constraints," *Journal of Creative Music Systems*, vol. 2, p. 1, 2018.
- [24] J. Schmidhuber, "Learning complex, extended sequences using the principle of history compression," *Neural Computation*, vol. 4, no. 2, pp. 234–242, 1992.
- [25] S. Hihi and Y. Bengio, "Hierarchical recurrent neural networks for long-term dependencies," *Advances in neural information processing systems*, vol. 8, pp. 493–499, 1995.
- [26] M. Huzzaifah and L. Wyse, "Mtcrrn: A multi-scale rnn for directed audio texture synthesis," *arXiv preprint arXiv:2011.12596*, 2020.
- [27] F. Colombo, S. P. Muscinelli, A. Seeholzer, J. Brea, and W. Gerstner, "Algorithmic composition of melodies with deep recurrent neural networks," in *1st Conference on Computer Simulation of Musical Creativity*, 2016.
- [28] D. Conklin and I. H. Witten, "Multiple viewpoint systems for music prediction," *Journal of New Music Research*, vol. 24, no. 1, pp. 51–73, 1995.
- [29] G. Medeot, S. Cherla, K. Kosta, M. McVicar, S. Abdallah, M. Selvi, E. Newton-Rex, and K. Webster, "Structurenet: Inducing structure in generated melodies," in *Proc. of the 19th Int. Society for Music Information Retrieval Conf., ISMIR, Paris, France, September 23-27, 2018*, pp. 725–731.
- [30] "Theorytab dataset," <https://www.hooktheory.com/theorytab>.
- [31] D. Meredith, K. Lemström, and G. A. Wiggins, "Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music," *Journal of New Music Research*, vol. 31, no. 4, pp. 321–345, 2002.
- [32] D. Herremans, E. Chew *et al.*, "Tension ribbons: Quantifying and visualising tonal tension," in *Proc. of Int. Conf. on Technologies for Music Notation and Representation (TENOR)*, 2016, vol. 2, pp. 8–18.
- [33] E. Chew, "The spiral array: An algorithm for determining key boundaries," in *Int. Conf. on Music and Artificial Intelligence*. Springer, 2002, pp. 18–31.

²<https://github.com/guozixunnicolas/CM-HRNN>