The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Modeling Temporal Tonal Relations in Polyphonic Music through Deep Networks with a Novel Image-Based Representation

Ching-Hua Chuan[1, 2]
[1] University of North Florida
[2] University of Miami
c.chuan@miami.edu

Dorien Herremans[3, 4]
[3] Singapore University of Technology and Design
[4] Institute of High Performance Computing, A*STAR, Singapore
dorien_herremans@sutd.edu.sg

Abstract

We propose an end-to-end approach for modeling polyphonic music with a novel graphical representation, based on music theory, in a deep neural network. Despite the success of deep learning in various applications, it remains a challenge to incorporate existing domain knowledge in a network without affecting its training routines. In this paper we present a novel approach for predictive music modeling and music generation that incorporates domain knowledge in its representation. In this work, music is transformed into a 2D representation, inspired by tonnetz from music theory, which graphically encodes musical relationships between pitches. This representation is incorporated in a deep network structure consisting of multilayered convolutional neural networks (CNN, for learning an efficient abstract encoding of the representation) and recurrent neural networks with long short-term memory cells (LSTM, for capturing temporal dependencies in music sequences). We empirically evaluate the nature and the effectiveness of the network by using a dataset of classical music from various composers. We investigate the effect of parameters including the number of convolution feature maps, pooling strategies, and three configurations of the network: LSTM without CNN, LSTM with CNN (pre-trained vs. not pre-trained). Visualizations of the feature maps and filters in the CNN are explored, and a comparison is made between the proposed tonnetz-inspired representation and pianoroll, a commonly used representation of music in computational systems. Experimental results show that the tonnetz representation produces musical sequences that are more tonally stable and contain more repeated patterns than sequences generated by pianoroll-based models, a finding that is directly useful for tackling current challenges in music and AI such as smart music generation.

Introduction

Predictive models of music have been explored by researchers since the very beginning of the field of computer music (Brooks et al. 1957). Such models are useful for applications in music analysis (Qi, Paisley, and Carin 2007); music cognition (Schellenberg 1996); improvement of transcription systems (Sigtia, Benetos, and Dixon 2016); music generation (Herremans et al. 2015); and others. Applications such as the latter represent various fundamental challenges in artificial intelligence for music. In recent years, there has been a growing interest in deep neural networks for modeling music due to their power to capture complex hidden relationships. The launch of recent projects such as Magenta, a deep learning and music project with a focus on music generation by the Google Brain team, testify to the importance and recent popularity of music and AI. With this project we aim to further advance the capability of deep networks to model music by proposing a novel image based representation inspired by music theory.

Recent deep learning projects in the field of music include Eck and Schmidhuber (2002), in which a recurrent neural network (RNN) with LSTM cells is used to generate improvisations (first chord sequences, followed by the generation of monophonic melodies) for 12-bar blues. They represent music as notes whose pitches fall in a range of 25 possible pitches (C_3 to C_5) and that occur at fixed time intervals. Therefore, the network has 25 outputs that are each considered independently. A decision threshold of 0.5 is used to select each note as a statistically independent events in a chord. More recently, a pianoroll representation of 88 keys has been used to train a RNN by Boulanger-Lewandowski, Bengio, and Vincent (2012). The authors integrate the notion of chords by using restricted Boltzmann machines on top of an RNN to model the conditioned distribution of simultaneously played notes in the next time slice, given the previous time slice. In Huang, Duvenaud, and Gajos (2016), a chord sequence is modelled as a string of symbols. Chord embeddings are learned from a corpus using Word2vec based on the skip-gram model (Mikolov et al. 2013), to describe a chord according to its sequential context. A Word2vec approach is also used in Herremans and Chuan (2017) to model and generate polyphonic music. For a more complete overview of music generation systems, the reader is referred to Herremans, Chuan, and Chew (2017). While music can typically be represented in either audio or symbolic format, the focus of this paper is on the latter.

The widely spread adoption of deep learning in areas such as image recognition is due to the high accuracy of models given the availability of abundant data, and its end-to-end solution to eliminate the need of hand-crafted features. Music, however, is a domain where well-annotated datasets are relatively scarce, but which has a long history of theoretical deliberation. It is therefore important to explore how such theoretical knowledge can be used to further improve deep models. Music theory goes beyond classification problems, and involves tasks such as analyzing a composition by studying the complex tonal system (for Western music) and its hierarchical structure (Kirlin and Jensen 2015). In this work, we aim to integrate knowledge from music theory in the input representation of a neural network, such that it can more easily learn specific musical features.

In this project, we propose an architecture that combines the temporal abilities of LSTM with a CNN autoencoder to model polyphonic music. We leverage the power of CNNs, by introducing a 2D representation, inspired by tonnetz, a representation used in music theory, to represent time slices in polyphonic music. To the best of our knowledge, this tonnetz-based representation has never been used in any existing deep learning approach for music. We use multi-layered CNNs as an autoencoder in order to capture musically meaningful relationships between pitches in tonal space. The autoencoder takes as input polyphonic music that has been converted into a sequence of tonnetz matrices, and which serve as an efficient abstract encoding. The encoded tonnetz sequence is then fed into an RNN with LSTM cells to capture temporal dependencies. This will allow us to predict the next musical time slice.

To examine the nature and effectiveness of the proposed architecture, several experiments were conducted using the MuseData dataset (Boulanger-Lewandowski, Bengio, and Vincent 2012) to extensively study the proposed tonnetz-based deep network. Results in terms of sigmoid cross entropy between the original tonnetz matrix and its encoded/decoded version are presented for different pooling strategies and different configurations of the network. Visualizations of the feature maps and filters of the convolution layer are also presented. Finally, the results of the predictive modeling using the proposed tonnetz representation are compared with the outputs of another pianoroll representation, and the generated music is evaluated.

In the next sections, we describe the proposed tonnetz matrix representation, followed by the architecture implemented in this research. The results of experiments are analyzed in the Experiments and Results section.

Tonnetz matrix for polyphonic music

Tonnetz is a graphical representation used by music theorists and musicologists in order to study tonality and tonal spaces. It was first described by Euler in 1739 for illustrating triadic structures (Euler 1926), and has evolved into multiple variations, each representing more complex musical relations including parsimonious voice-leading (Cohn 1997). It also provides a theoretical foundation for many music information retrieval (MIR) applications such as chord recognition (Harte, Sandler, and Gasser 2006) and structural visualization (Bergstrom, Karahalios, and Hart 2007).

Figure 1 (a) illustrates a common form of tonnetz. Each node in the tonnetz network represents one of the 12 pitch classes. The nodes on the same horizontal line follow the circle-of-fifth ordering: the adjacent right neighbor is the perfect-fifth and the adjacent left is the perfect-fourth. Three nodes connected as a triangle in the network form a triad, and the two triangles connected in the vertical direction by sharing a baseline are the parallel major and minor triads. For example, the upside-down triangle filled with diagonal lines in Figure 1 (a) is C major triad, and the solid triangle on the top is C minor triad. Note that the size of the network can be expanded boundlessly; therefore, a pitch class can appear in multiple places throughout the network.

In this paper, we extended the above tonnetz representation into a 24-by-12 matrix as partially shown in Figure 1 (b). In this matrix, each node represents a pitch instead of a pitch class so that the pitch register information is kept. Similar to tonnetz, nodes on the same horizontal line show the circle-of-fifth relation. The pitch register is determined by the distance to the pitch in the center column highlighted in the dashed box. For example, the register of the pitch G_3 next to C_4 in the second row is 3 instead of 4 because G_3 is closer to C_4 than G_4 in terms of the number of half steps. The pitches in a column of the matrix preserve the interval of major third, which can be observed in the dashed box in Figure 1 (b) compared to the tilted dashed box in Figure 1 (a). The number of rows in the tonnetz matrix can be determined by the range of pitches of interest in a particular study. In this paper, the matrix is designed with 24 rows covering pitches from C_0 to $C^\#_8$.

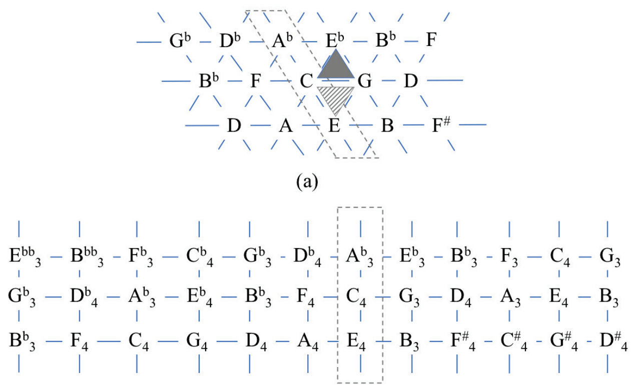

Gb — Db — Ab — Eb — Bb — F
Bb — F — C — G — D
D — A — E — B — F#
(a)
Ebb3 – Bbb3 – Fb3 – Cb4 – Gb3 – Db4 – Ab3 – Eb3 – Bb3 – F3 – C4 – G3
Gb3 – Db4 – Ab3 – Eb4 – Bb3 – F4 – C4 – G3 – D4 – A3 – E4 – B3
Bb3 – F4 – C4 – G4 – D4 – A4 – E4 – B3 – F#4 – C#4 – G#4 – D#4
(b)

Figure 1: (a) tonnetz and (b) the extended tonnetz matrix with pitch register

The proposed tonnetz matrix allows us to model multiple pitches played simultaneously during a time slice in polyphonic music. In this paper, the entire composition is divided into time slices of length 1 beat. The pitches played at each beat are labeled as 1 in the tonnetz matrix and 0 for pitches not played at that time. Before converting a composition to a sequence of tonnetz matrices, we transposed it to either C major or A minor, depending on the mode of its original key signature. In this way, the harmonic role of a pitch (e.g. tonic, dominant) is preserved and is represented in the same location in the tonnetz matrix.

The next section describes how a deep network was constructed to model a sequence of tonnetz matrices.

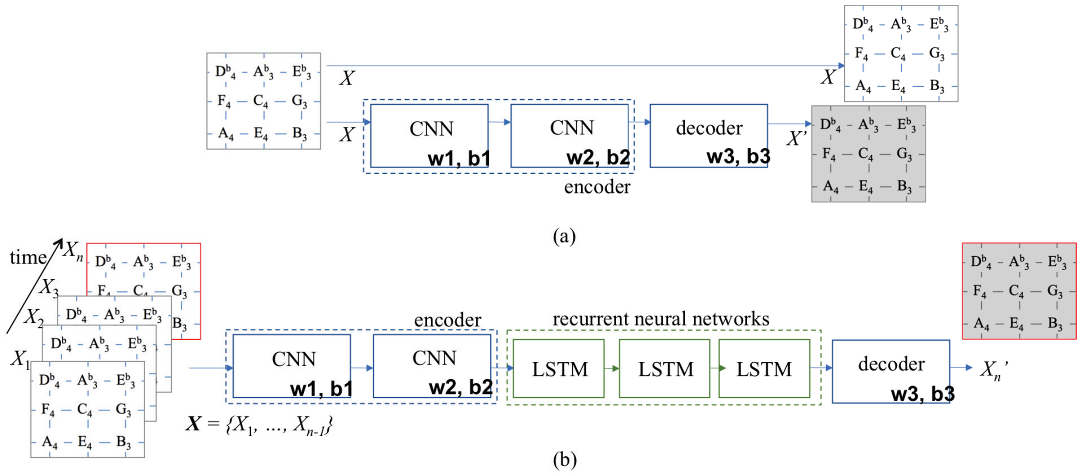


Figure 2: (a) Autoencoder using two layers of CNNs and (b) sequence modeling using autoencoder and LSTM networks.

Network structure

The system described in this paper implements a network structure that is depicted in Figure 2 (b). It consists of two main parts: a two-layered convolutional autoencoder and a three-layered LSTM. The first part of the network structure, the two-layered convolutional neural network, is pre-trained as an autoencoder, as shown in Figure 2 (a). This allows the network to learn an efficient abstract representation for encoding tonnetz matrices. The encoded outputs are then fed into the LSTM network structure, a type of neural network often used in sequence modeling and prediction tasks (Graves, Mohamed, and Hinton 2013; Sutskever, Vinyals, and Le 2014; Vinyals et al. 2015).

The network architecture was inspired by recent work on piano music transcription (Sigtia, Benetos, and Dixon 2016) and video sequence prediction (Finn, Goodfellow, and Levine 2016). In the next subsections, we will first discuss the convolutional autoencoder, followed by an in-depth description of the LSTM network.

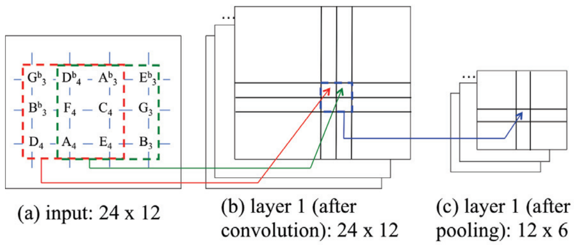

(a) input: 24 x 12
(b) layer 1 (after convolution): 24 x 12
(c) layer 1 (after pooling): 12 x 6

Figure 3: An illustration of the first layer CNN in the autoencoder.

Convolutional autoencoder

An autoencoder is a popular technique that takes advantage of unsupervised learning to build an efficient representation of the input. It has been used in various applications including information retrieval (Salakhutdinov and Hinton 2009) and text generation (Li, Luong, and Jurafsky 2015). Uses of autoencoders include feature detection and dimension reduction, but it has recently also been used as part of generative models (Kingma and Welling 2013).

The main idea of an autoencoder is to learn a model that first encodes the input into an abstract representation such that this abstract representation can then be decoded and restored back to the original input as close as possible (Goodfellow, Bengio, and Courville 2016). In this paper, the input consists of a tonnetz matrix *X*, as shown in Figure 2 (a), and the loss function is defined as the sigmoid cross entropy between *X* and the encoded/decoded version *X'*.

The autoencoder consists of two convolutional layers (LeCun 1989) for encoding and one fully-connected layer for decoding. Each encoding layer has two components: a convolution layer and a pooling layer. In a convolution layer, each neuron is connected to a local receptive field (kernel) with size of 3-by-3 in the tonnetz matrix, as shown in the dashed box on the input tonnetz matrix in Figure 3 (a). This size is chosen based on the number of pitches in a triad. The stride, the distance between two consecutive receptive fields, is set to 1 in both vertical and horizontal directions. A non-linear activation function (rectified linear unit) is added to the convolution layer in order to generate feature maps as output, see Figure 3 (b). A pooling layer, immediately placed after the convolution layer, produces a condensed version of the feature maps as shown in Figure 3 (c).

Given a tonnetz matrix *X* and a kernel *K* with a size of *m*-by-*n*, *m* = *n* = 3, the value in the cell (*i*, *j*) of the convolution layer *C* in Figure 3 (b) is calculated as follows:

$$C(i,j) = \sum_m \sum_n X(i+m, j+n) \times K(m,n) \quad (1)$$

In this paper, we tested two pooling strategies, including max pooling and average pooling. We also tested the effect of the number of feature maps in each convolution layer. Besides these testing parameters, others are pre-determined as follows: the size of the pooling window is set to 2-by-2 in the first layer, and to 2-by-1 in the second layer. The stride of the pooling window is set in a similar fashion. Given a 24-by-12 tonnetz matrix, such settings result in 12-by-6 feature maps after the first layer, and 6-by-6 after the second layer.

To examine the effect of the autoencoder in the architecture, we tested three configurations: no autoencoder (LSTM only); training the entire network together (autoencoder + LSTM); and pre-training the autoencoder and then freezing the weights of the autoencoder when training the LSTM. The experiment results are described in the Experiments and Results section. The next section discusses how LSTMs are used for modeling temporal relations in musical sequences.

Predicting music sequences with LSTMs

In this paper, polyphonic compositions are segmented into sequences of n tonnetz matrices using overlapped sliding windows. The proposed predictive modeling system outputs X_n' aiming to predict the tonnetz matrix X_n given the preceding sequence $\{X_1, ..., X_{n-1}\}$, as shown in Figure 2 (b).

A recurrent neural network approach is implemented to capture the temporal nature of musical sequences. RNNs (Rumelhart, Hinton, and Williams 1988) are able to capture temporal information by defining a recurrence relation over the time steps k:

$$S_k = f(S_{k-1} \times W_r + X_k \times W_x), \tag{2}$$

whereby S_k is the state at time k, X_k is the input at time k, and W_r and W_x are weight parameters. The state S_k of the network changes over time due to this recurrence relation and receives feedback with a delay of one time step, which makes it, in essence, a state model with a feedback loop (see Figure 4). The unfolded network can be seen as an $(n + 1)$ layer neural network with shared weights W_r and W_x.

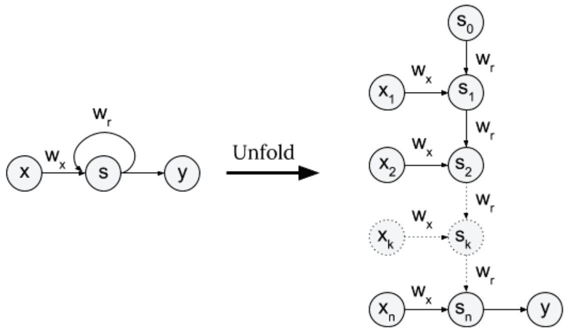


Figure 4: Recurrent neural network unfolding, illustrated based on (Goodfellow, Bengio, and Courville 2016).

Standard recurrent neural networks, however, are notoriously hard to train using back propagation due to the vanishing gradient problem when modeling long sequences (Rumelhart, Hinton, and Williams 1988). In this well-known problem, the gradient grows or decays exponentially as it is propagated through the network (Bengio, Simard, and Frasconi 1994). Approaches that aim to avoid this problem use better optimization algorithms with higher-order information (Martens and Sutskever 2011), however, this requires a significant increase in computing power. We therefore opted to use LSTM cells, which offer a way around this as their architecture explicitly avoids the vanishing gradient problem while preserving the training algorithm.

LSTM is a type of recurrent neural network developed by (Hochreiter and Schmidhuber 1997). It is particularly strong at modeling temporal sequences and their long-range dependencies, even more so than conventional RNNs (Sak, Senior, and Beaufays 2014). They have been successfully used in applications such as speech recognition (Graves, Mohamed, and Hinton 2013); sequence to sequence generation: text translation (Sutskever, Vinyals, and Le 2014); image caption generation (Vinyals et al. 2015); hand writing generation (Graves 2013), Image generation (Gregor et al. 2015); and video to text transcription (Venugopalan et al. 2015).

The vanishing gradient problem is directly avoided by LSTM cells, by implementing a unit known as "constant error carousel" (CEC) with a weight set to 1.0. This CEC, together with input and output gates, control the error flow and enforces the gradient to be constant. This basic LSTM cell was later expanded to include strategies such as forget gates (Gers and Schmidhuber 2001), peepholes (Gers, Schraudolph, and Schmidhuber 2002), clipping and projection (Sak, Senior, and Beaufays 2014). For a full mathematical description of the inner workings of LSTM cells, the reader is referred to (Hochreiter and Schmidhuber 1997; Graves and Schmidhuber 2005).

The LSTM network implemented in this research uses standard LSTM cells and consists of three hidden layers. The loss function is calculated with a sigmoid of the cross entropy between the predicted next tonnetz and the actual next tonnetz. In the next section, we describe a number of experiments that test the effectiveness and efficiency of the proposed architecture, together with their results.

Experiments and Results

The MuseData dataset published by Boulanger-Lewandowski, Bengio, and Vincent (2012), a collection of musical pieces in MIDI format, was used to evaluate our proposed architecture in the experiments described below. This dataset is already divided into training (524 pieces), validation (135 pieces), and test (124 pieces) sets with compositions by various composers including Bach, Beethoven, Haydn, and Mozart. Each composition is represented as a sequence of sets of pitches played at each beat.

First, the training set was used to train the network in the experiment. The effectiveness and efficiency of the tonnetz-based autoencoder was evaluated through the validation and test sets, followed by a visualization of the feature maps of certain chords and filters of the trained model. Finally, the proposed tonnetz-based representation was compared with a model that implements pianoroll using the test set.

Tonnetz-based autoencoder

Figure 5 shows the loss function, evaluated using the validation set, during the training of the autoencoder for two different pooling strategies. Table 1 shows the different parameter settings that are evaluated for the convolution layers.

As expected, increasing the number of feature maps results in lower average loss. However, these losses converge as the number of training epochs increases. When comparing the pooling strategies, max pooling generally outperforms average pooling. Based on these experimental results, we chose the following parameters to avoid overfitting: (20, 10) for the numbers of feature maps in layers 1 and 2, with 10 epochs.

No. of maps/Test no.	1	2	3	4	5
Convolution layer 1	40	20	10	10	5
Convolution layer 2	20	10	10	5	5

Table 1: Different values tested for the number of feature maps in the convolution layers of the autoencoder.

To study the effectiveness of the tonnetz-based autoencoder, three configurations were tested of which the results are shown in Figure 6. The x-axis shows the number of training batches, each of which consists of 500 instances. The y-axis shows the cross entropy loss on the test set using the model that trained on the training set with specific training epochs. As shown in Figure 6, the system with pre-trained autoencoder converges much quicker than the other two configurations. The observed benefit of quick convergence from the pre-trained tonnetz-based autoencoder is especially desirable when training on a much larger dataset.

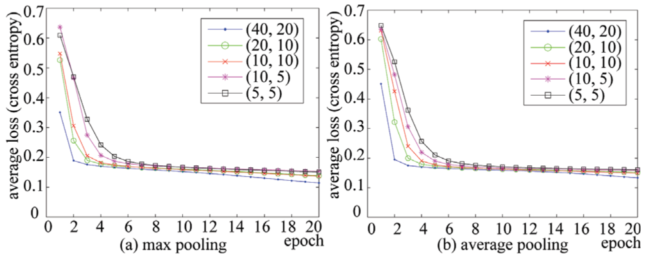


Figure 5: Evolution of loss on the validation set during the training of the tonnetz-based autoencoder with different pooling strategies.

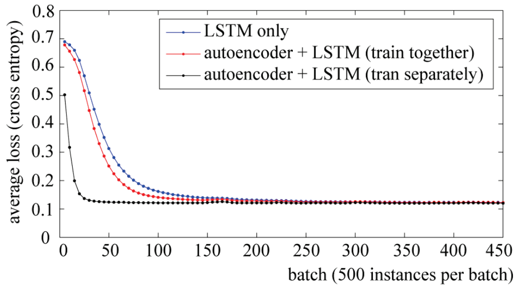


Figure 6: Comparisons between the three settings of the system: LSTM only (no autoencoder), autoencoder (no pretraining) and LSTM (train together), and pre-trained autoencoder and LSTM (train separately) on the test set.

Visualizing feature maps and filters in autoencoder

To gain insight in the workings of the autoencoder, Figure 7 visualizes the first 5 (out of 10) outputs from the second convolution layer of the autoencoder for the triads C major, G major, C diminished, and G diminished. Grey scale is used to visualize the values, ranging from black (0) to white (1). The figure reveals some relationships between the first two chords (major triads) and the latter two chords (diminished triads), both a fifth apart. When examining feature maps 2, 3, and 4 for the diminished chords, a clear shift of 1 unit down can be noticed between Cdim and Gdim, indicating that a transposition of a fifth is captured by a downward shift in layer 2. A similar relationship can be observed between the major triads of C and G, for instance, as indicated in map 2.

Note that the precise semantic meaning of such feature maps is difficult to pinpoint and may not exist. However, similar patterns observed among chords sharing the same quality show that the tonnetz-based autoencoder is capable of capturing chordal nature in music.

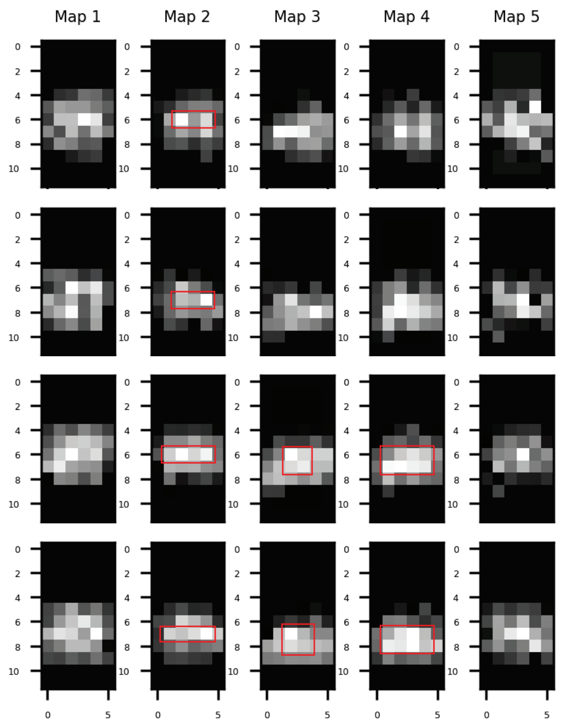


Figure 7: First 5 feature maps from the second convolution layer of the autoencoder for the chords C (row 1), G (row 2), Cdim (row 3), and Gdim (row 4).

Compared to feature maps, the filters of the first layer of the autoencoder are typically easier to interpret. Figure 8 shows these filters (the first 10 out of 20, due to space constraints) for a model trained on the same dataset as above. These filters clearly reflect specific musical properties. For instance, horizontal highlighted lines (e.g. Filter 3, 6, 7 and 9) show steps in the circle of fifths. When two positions are highlighted right above each other, such as in Filter 3 and 4 this indicates a third relationship. This confirms that using a tonnetz-representation facilitates the model to learn musically meaningful features.

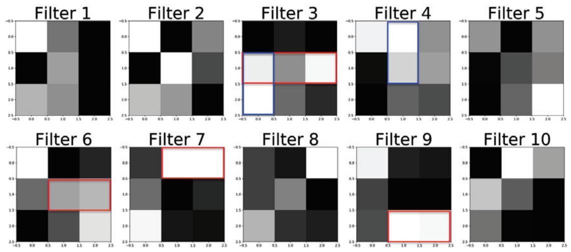


Figure 8: First layer filters in the autoencoder. The figure shows the first 10 out of 20 filters.

Tonnetz versus Pianoroll representations

In order to thoroughly evaluate the validity of the tonnetz representation, an experiment was set up to compare the tonnetz approach with a commonly used representation, namely pianoroll (Boulanger-Lewandowski, Bengio, and Vincent 2012; Sigtia, Benetos, and Dixon 2016). The pianoroll representation describes music as a sequence of vectors, each of length 88, which uses binary values to indicate if each pitch on a 88-key piano is played at a particular time slice.

In order to properly evaluate the effect of the input representation, we use the same network structure for both tonnetz and pianoroll representations, with the exception that we use a one-dimensional CNN for pianoroll instead of a two-dimensional CNN. Two types of experiments are conducted in this section: a first one evaluating predictive modeling (predicting the next frame given a historical context) and secondly, music generation (generating a sequence of notes given a seed segment).

Predictive modeling In the first experiment on predictive modeling, the next tonnetz is predicted based on a given historical context (previous slices) for each tonnetz in the test set. Generic evaluation metrics such as cross entropy, precision, and recall did not indicate significant differences in the predictive results between the two representations. However, when the predictions of two models are compared using music-specific metrics (i.e., musical tension), a significant difference is found. The tension model developed by (Herremans and Chew 2016) was used to capture three elements of tonal tension: cloud diameter (CD), a measure for dissonance; cloud momentum (CM), which indicates changes in tonality; and tensile strain (TS), the tonal distance to the global key of the piece.

To focus on the difference between tonnetz and pianoroll, we only examine the predictive results where the number of pitches that are different between tonnetz and pianoroll results is greater than the number of pitches they have in common. This reduces the number of predictions to one-third of the test set. The boxplots in Figure 9 visualize the three aspects of tonal tension for the predictions of both the pianoroll and the tonnetz model. In the case of a tonnetz-based representation, the predicted next slices generally have lower tension values, especially in the case of Cloud diameter (dissonance) and Tensile strain (distance from the key). This indicates that the tonnetz model leads to a more stable tonal predictions.

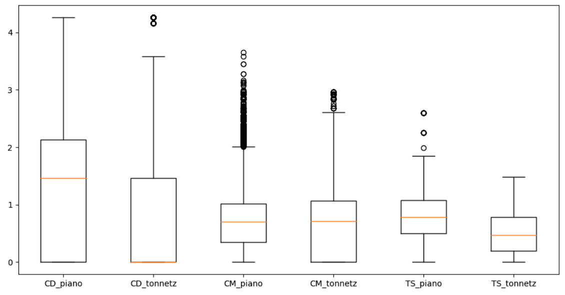


Figure 9: Boxplots for three tension characteristics of the pianoroll and tonnetz-based models. For each of these characteristics, pianoroll values are significantly higher than tonnetz ($p < 0.001$).

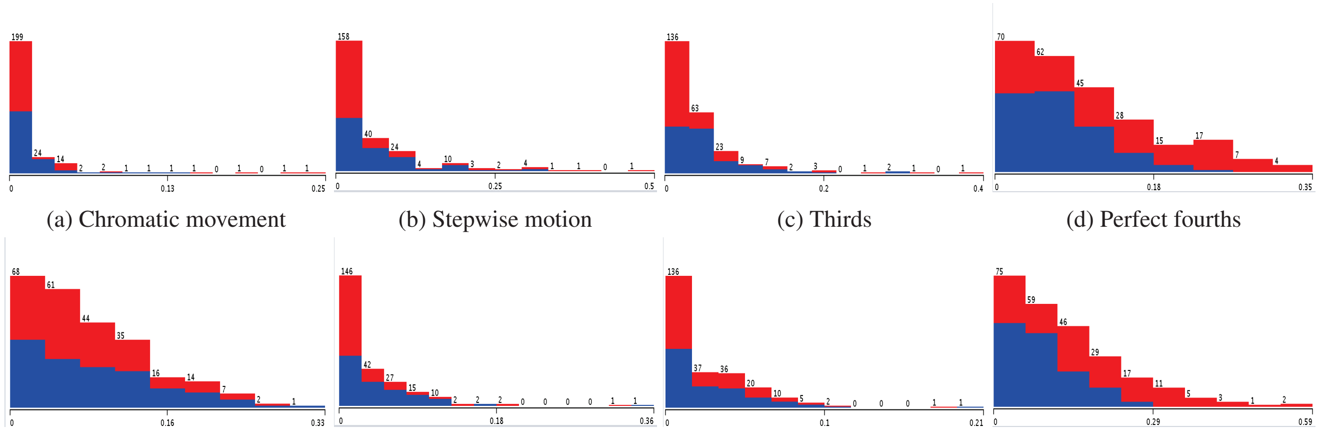
Generating new pieces In the second experiment the focus lies on evaluating newly generated musical pieces. A total of 248 new pieces (124 for each approach) were generated by seeding 16 beats from the original composition to the model. A sliding window approach is used to continuously generate notes by using the generated notes as the historical context for the future. The new pieces were evaluated using a collection of musical attributes: compression ratio, tonal tension, and interval frequencies.

Firstly, the compression ratio of the generated midi files was calculated using COSIATEC, a compression algorithm that has previously been used for discovering themes and sections (Meredith 2015) and constraining patterns in music generation (Herremans and Chew 2017). Given the current challenge in the field of music generation of generating pieces with long-term structure and repeated patterns (Herremans, Chuan, and Chew 2017), the compression ratio can give us insight in the ability of the LSTM to generate repeated patterns. The results in Table 2 indicate that the generated pieces with the tonnetz representation have a significantly higher compression ratio. This indicates the presence of a larger structure and more repetition.

	Pianoroll	Tonnetz	p-value
Compression ratio	2.232	2.266	0.0307
Tension: CD	0.950	0.804	< 0.001
Tension: CM	0.491	0.396	< 0.001
Tension: TS	0.615	0.514	< 0.001

Table 2: Comparison of musical characteristics of generated sequences based on pianoroll and tonnetz representation. p-values of a paired t-test are displayed in the last column.

Similar to the predictive results, the tonal tension for the generated pieces with tonnetz-representation is consistently



(a) Chromatic movement (b) Stepwise motion (c) Thirds (d) Perfect fourths

(e) Perfect fifths (f) Sixths (g) Sevenths (h) Octaves

Figure 10: Frequency of melodic intervals with the number of songs on the y-axis and the frequency on the x-axis. Red is tonnetz, blue is pianoroll representation.

lower. This is also reflected by the frequencies of musical intervals extracted with jSymbolic2 (McKay, Tenaglia, and Fujinaga 2016), as depicted in Figure 10. Most notably, results from the tonnetz-model seem to include more intervals such as octaves and perfect fourths, and less stepwise motion, sixths and sevenths. It is not a given that tonal stability and low tension is something that is desired in all musical pieces, as this is strongly dependant on the composer's wishes and the style of the piece. However, this may be an important consideration to be taken into account when selecting to work with either tonnetz or pianoroll models.

The code implemented in this paper is made available online[1]. The reader is invited to listen to audio files of the generated pieces available online at the Computer-Generated Music Repository[2].

Conclusions

In this paper, a predictive deep network with a music-theory inspired representation is proposed for modeling polyphonic music. In the presented approach, music is first converted into a sequence of tonnetz matrices, which serves as input for an autoencoder. This autoencoder, which consists of multilayered CNNs, captures the relationship between pitches in tonal space. The output of the autoencoder is fed to a recurrent neural network with LSTM cells in order to model the temporal dependencies in the sequence. The nature of the proposed tonnetz-based autoencoder was studied in a number of experiments using the MuseData dataset. We found that, on average, a max pooling strategy is most effective and a pre-trained tonnetz-based autoencoder helps the LSTM to converge quicker. In addition to these results, a visualization of feature maps and filters in the trained autoencoder show they reflect musical properties such as intervals. Finally, predictive results for the tonnetz-model was compared with a pianoroll approach. This showed that sequences generated based on tonnetz were generally more tonally stable and contained less tension. They also had a significantly higher compression ratio, which indicates that they contain more repeated patterns and structure. In sum, by integrating a tonnetz representation, which embeds domain specific knowledge, in an LSTM, which is able to model temporal dependencies, we were able to develop a stronger model of music. This will be useful for tackling remaining challenges in the field of AI and music.

In future research, it would be interesting to test the proposed model on larger datasets of different music styles and genres. The ability of the model to fully capture musical sequences may also be further improved by experimenting with incorporating expanded LSTM cells into the model.

[1]http://sites.google.com/view/chinghuachuan/

[2]http://dorienherremans.com/cogemur

References

Bengio, Y.; Simard, P.; and Frasconi, P. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5(2):157–166.

Bergstrom, T.; Karahalios, K.; and Hart, J. C. 2007. Isochords: visualizing structure in music. In *Proc. of Graphics Interface 2007*, 297–304. ACM.

Boulanger-Lewandowski, N.; Bengio, Y.; and Vincent, P. 2012. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proc. of the 29th Int. Conf. on Machine Learning*, 1159–1166. ICML.

Brooks, F. P.; Hopkins, A.; Neumann, P. G.; and Wright, W. 1957. An experiment in musical composition. *IRE Transactions on Electronic Computers* (3):175–182.

Cohn, R. 1997. Neo-riemannian operations, parsimonious trichords, and their" tonnetz" representations. *Journal of Music Theory* 41(1):1–66.

Eck, D., and Schmidhuber, J. 2002. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Proc. of the 12th IEEE Workshop on Neural Networks for Signal Processing*, 747–756.

Euler, L. 1926. Tentamen novae theoriae musicae. leonhardi euleri opera omniae.

Finn, C.; Goodfellow, I.; and Levine, S. 2016. Unsupervised learning for physical interaction through video prediction. In *Advances In Neural Information Processing Systems*, 64–72.

Gers, F. A., and Schmidhuber, E. 2001. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks* 12(6):1333–1340.

Gers, F. A.; Schraudolph, N. N.; and Schmidhuber, J. 2002. Learning precise timing with lstm recurrent networks. *Journal of machine learning research* 3(Aug):115–143.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press.

Graves, A., and Schmidhuber, J. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* 18(5):602–610.

Graves, A.; Mohamed, A.-r.; and Hinton, G. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, 6645–6649. IEEE.

Graves, A. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Gregor, K.; Danihelka, I.; Graves, A.; Rezende, D.; and Wierstra, D. 2015. Draw: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 1462–1471.

Harte, C.; Sandler, M.; and Gasser, M. 2006. Detecting harmonic change in musical audio. In *Proc. of the 1st ACM workshop on Audio and music computing multimedia*, 21–26. ACM.

Herremans, D., and Chew, E. 2016. Tension ribbons: Quantifying and visualising tonal tension. In *Second International Conference on Technologies for Music Notation and Representation (TENOR)*, volume 2, 8–18.

Herremans, D., and Chew, E. 2017. Morpheus: generating structured music with constrained patterns and tension. *IEEE Transactions on Affective Computing* PP:99.

Herremans, D., and Chuan, C.-H. 2017. Modeling musical context with word2vec. In *Proc. of the first Int. Workshop On Deep Learning and Music*, volume 1, 11–18.

Herremans, D.; Weisser, S.; Sörensen, K.; and Conklin, D. 2015. Generating structured music for bagana using quality metrics based on markov models. *Expert Systems with Applications* 42(21):7424–7435.

Herremans, D.; Chuan, C.-H.; and Chew, E. 2017. A functional taxonomy of music generation systems. *ACM Computing Surveys* 50:69:1–30.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Huang, C.-Z. A.; Duvenaud, D.; and Gajos, K. Z. 2016. Chordripple: Recommending chords to help novice composers go beyond the ordinary. In *Proc. of the 21st Int. Conf. on Intelligent User Interfaces*, 241–250. ACM.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kirlin, P. B., and Jensen, D. D. 2015. Using supervised learning to uncover deep musical structure. In *Proceedings of the 29th AAAI Conf. on Artificial Intelligence*, 1770–1776.

LeCun, Y. 1989. Generalization and network design strategies. *Connectionism in perspective* 143–155.

Li, J.; Luong, M.-T.; and Jurafsky, D. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*.

Martens, J., and Sutskever, I. 2011. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 1033–1040.

McKay, C.; Tenaglia, T.; and Fujinaga, I. 2016. jSymbolic2: Extracting features from symbolic music representations. In *Late-Breaking Demo Session of the 17th International Society for Music Information Retrieval Conference*.

Meredith, D. 2015. Music analysis and point-set compression. *Journal of New Music Research* 44(3):245–270.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Qi, Y.; Paisley, J. W.; and Carin, L. 2007. Music analysis using hidden markov mixture models. *IEEE Transactions on Signal Processing* 55(11):5209–5224.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1988. Learning representations by back-propagating errors. *Cognitive modeling* 5(3):1.

Sak, H.; Senior, A.; and Beaufays, F. 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Salakhutdinov, R., and Hinton, G. 2009. Semantic hashing. *Int. J. of Approx. Reas.* 50(7):969–978.

Schellenberg, E. G. 1996. Expectancy in melody: Tests of the implication-realization model. *Cognition* 58(1):75–125.

Sigtia, S.; Benetos, E.; and Dixon, S. 2016. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 24(5):927–939.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Venugopalan, S.; Rohrbach, M.; Donahue, J.; Mooney, R.; Darrell, T.; and Saenko, K. 2015. Sequence to sequence-video to text. In *Proceedings of the IEEE International Conference on Computer Vision*, 4534–4542.

Vinyals, O.; Toshev, A.; Bengio, S.; and Erhan, D. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3156–3164.