A Deep Learning Approach to Melody Composition and Harmonization with a hybrid LSTM-RBM model

Melvin Govender
University of Kwa-Zulu Natal
215060212@stu.ukzn.ac.za

ABSTRACT

This paper examines the hybrid LSTM-RBM model's ability in solving the two nontrivial tasks of melody generation and harmonization in the music composition process. The objective is a deep learning model that is capable of improvising polyphonic classical piano music from scratch in an aesthetically pleasing manner. In order to motivate the need for a hybrid deep learning model, experiments are first conducted on each component of the model, to expose each component's shortcomings and strengths. In building the hybrid model, the de facto standard was applied, that is, the simplest model is built and complexity is added thereafter as needed. Since music is subjective, the music generated by the model is evaluated subjectively by humans. It is determined that the hybrid LSTM-RBM model is a promising model in the area of music composition despite the research in this paper producing below par results.

KEYWORDS

Stacked LSTM-RBM, Music Composition, Deep Learning

1 INTRODUCTION

Music composition is a difficult task for a machine to model due to the creative element involved in the composition process – creativity simply isn't quantifiable, it is inherently human which makes it difficult for computers to replicate. Successful research in the field of artificial creativity would take us one step closer to the ultimate goal of artificial intelligence – a machine that thinks like humans do.

The music composition process is inherently a human, it follows then that the most suitable approach in attempting to model music would be a machine learning model that is based on neurons firing in the human brain – neural networks. More specifically, deep neural networks as deep learning has been known to outperform traditional machine learning algorithms on complex datasets. Neural approaches to music composition date back to 1989 [15] and are favored over algorithmic approaches due to the neural approaches learning by example instead of the programmer having to explicitly model the music into logical rules.

The research community has made enormous strides in modeling a single task of the music composition process but research on a model capable of performing more than one task is scarce [10] but has gained ground recently due to the advancements in hardware and big data being freely available. The majority of research in performing a single task is focused on either melody composition or harmonization – this paper proposes a model that attempts to perform both tasks for the genre of Bach 4-part chorales.

The LSTM-RBM model seems an intuitive choice for artificial music composition. Music is sequential in nature, it is simply a sequence of pitches over time, thus a model that incorporates the concept of time into its algorithm is essential and the Long-Short-Term-Memory (LSTM) network fulfills this requirement. The Restricted-Boltzmann Machine (RBM) learns to reconstruct data by using the conditional probability distribution of the training data which is ideal as forming a harmony can be thought of as determining which notes are most likely to occur together.

The remainder of the paper is organized as follows. A brief introduction to music theory concepts is discussed in section 2. In section 3, related work in artificial music composition, specifically hybrid systems with a similar architecture, is reviewed. In section 4, the data used in this paper is discussed in detail. In section 5, the approach and methodology are discussed. The hybrid LSTM-RBM architecture as well as the results are discussed in section 6. Then, evaluation of the hybrid model is discussed in section 7.

2 MUSIC THEORY CONCEPTS

In an effort to provide clarity on the problem this paper attempts to solve, a brief description of key music theory concepts is given here.

For the purpose of this research, a melody is considered to be the Soprano part of the musical piece and the harmonization is considered to be the corresponding Alto, Tenor and Bass parts (see Fig. 1).

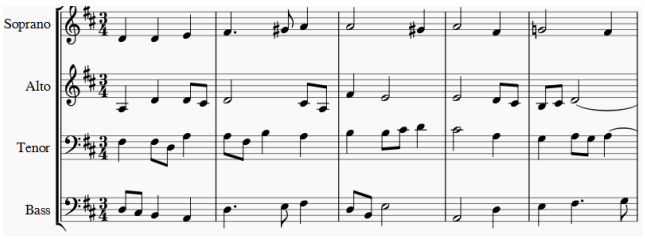

Figure 1: Snippet of Bach 4-part chorale (bwv 11.6)

Music theory concepts are used loosely throughout this paper, a pitch determines the actual note (A-G) that is played, an octave determines the frequency of the note being played (where it is placed on the staff) and the duration of a note determines how long the note is played.

3 RELATED WORK

3.1 RNN-RBM

The RNN-RBM architecture proposed in [1] has become one of the key architectures in polyphonic music generation due to the sheer quality of its results and serves as the main inspiration for the model proposed in this paper. This hybrid model attempts to combine the ability of a recurrent neural network (RNN) to model temporal sequences with the ability of an RBM to model a complex probability distribution. Broadly, the RNN is expected to communicate the general structure of a musical piece to an RBM, which then determines the combination of notes that should occur at this point in time.

The resulting architecture can be seen in Figure 2. The RNN at timestep t is represented as $u^{(t)}$. The RBM's visible layer and hidden layer at timestep t is represented as $v^{(t)}$ and $h^{(t)}$ respectively. The RNN at timestep t determines the biases for the respective visible layer and hidden layer of the RBM at timestep t+1 but the weights of the RBM are handled internally for all timesteps.

Since RNN units are highly susceptible to the vanishing/exploding gradient problem which affect the model's ability in capturing long-term structure, it is reasonable to assume that replacing the RNN units with LSTM units will yield improved results since LSTM units have been proven to resolve these issues [2]. This paper is based on such a model.

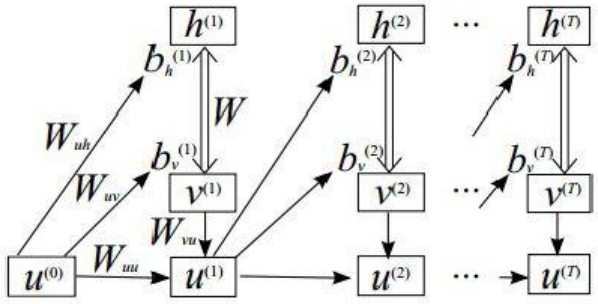

Figure 2: The RNN-RBM architecture

3.2 LSTM-RTRBM

The LSTM-RTRBM model proposed in [3] is an extension of the RNN-RBM model discussed above. The recurrent temporal RBM (RTRBM) [4] is a sequence of conditional RBMs where the hidden units are propagated through time, this means that an RBM's parameters at time t are dependent on the hidden layer of the RBM at time t-1. This acts very much as a short-term memory structure and aids the RBM model in capturing sequences. The LSTM-RTRBM model builds upon the RNN-RBM model in one significant way – the incorporation of both short-term and long-term memory into its design. Music composition requires long-term memory so that the overall theme of the musical piece is consistent. Short-term memory is vital in making the transition between bars as seamless as possible.

The LSTM-RTRBM model slightly improved upon the results of the RNN-RBM model [3] but it is worth noting that this model is significantly more complex than its predecessor as there are now two recurrence units communicating with the RBM at each timestep – each modeling long-term memory and short-term memory respectively.

However, the LSTM unit, in theory, should suffice in capturing short-term memory due to its forget gate layer, that is, the layer that decides what information in memory is irrelevant and should subsequently be replaced with new information.

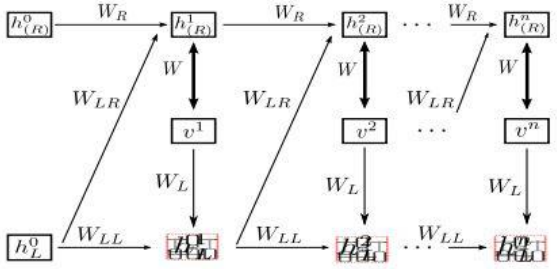

Figure 3: LSTM-RTRBM architecture

4 DATA

4.1 Music Preprocessing

In order to simplify the representation into something more computationally feasibly, the music has to undergo a few preprocessing techniques. First, conversion of all musical pieces to either A-minor or C-major scale reduces the note dimensionality of the problem whilst the loss in quality of music learned is negligible since music can be transposed from one key to another. Furthermore, simplicity is added by assuming that notes occurring at the same time are also played for the same duration. Again, this reduces the dimensionality of the problem as time/duration is associated with each chord (set of notes) instead of each note in a chord. This allows for the incorporation of multiple voices whilst incurring relatively little complexity.

4.2 Music Representation

A machine learning model requires a simpler representation than the one offered by the MXL and MIDI format. Features such as pitch and duration of a note are extracted and mapped to a discrete value. The LSTM-RBM network requires a binary vector as input - a typical input fed into the visible layer can be seen in Figure 4. For simplicity sake, the input in Figure 4 is for the case of a single timestep with just two (2) unique pitches and three (3) unique durations – a single timestep represents a single chord (4 notes played simultaneously). Thus, Figure 4 describes the case where the notes [A, A, B, A] are played at the same time and for

a duration of 2.0. This representation is used for majority of the experiments but it is later determined that this representation is insufficient and that the octave of a note must be incorporated into the representation.

.

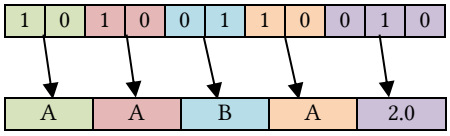

Figure 4: Representation of music for LSTM-RBM model

5 METHODOLGY AND EXPERIMENTATION

5.1 Approach

Before assessing the LSTM-RBM model's suitability in generating polyphonic music, one has to examine the individual components first. The purpose is to learn the inner workings of these individual components, to determine their shortcomings and strengths in terms of music composition thus motivating the need for a hybrid model.

5.2 LSTM - Melody Composition

In this section, experiments conducted on the LSTM's ability in generating a melody (single voice) is described. The LSTM model discussed here essentially predicts the best musical note at time t+1 given the previous notes at time t. The training process at each iteration involves observing a sequence of notes at time t, predicting a note for time t+1 given that sequence, and then comparing that predicted note to the actual note at time t+1, determining the difference (loss) between the predicted note and the actual note, and finally adjusting the parameters of the model accordingly.

It is worth noting that the music representation used in this experiment differs significantly from the representation used in the final LSTM-RBM model. Since this experiment focuses only on melody generation, there is no need to incorporate the accompaniment voices into the representation. Each musical note is mapped to a discrete value depending on two factors – pitch and duration. Also, the LSTM requires a corresponding output for each input sequence, the output is simply the value at time t+1 (if we're observing the input sequence at time t). The output is a one-hot encoded binary vector where a 1 indicates that the note allocated to that index is being played and 0 indicates that it isn't.

The de facto approach was adopted - the simplest, shallowest model is built first and additional layers and complexity are only added if necessary.

Experiments conducted on a single layer LSTM indicated that such a model isn't capable of modeling music (see Fig. 5.1), even widening the network by increasing the number of neurons in the LSTM unit or reducing the number of timesteps offered little to no improvement in results. However, deepening the network yielded improved results with each additional layer (see Fig. 5.2) – it is clear that a shallow network isn't suitable in capturing the abstract nature of music.

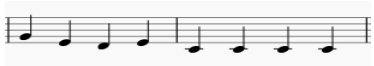

Figure 5.1: Snippet of melody from shallow LSTM[1]

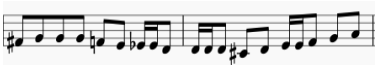

Figure 5.2: Snippet of melody from deep LSTM[2]

A melody is generated by priming the LSTM network with a random sequence of notes and then repeating the following process until the length of the piece generated is sufficient:

1. Predict the note at time t+1 given the sequence at time t.
2. Add this note to the generated piece and the sequence at time t.
3. Drop the first note from the sequence at time t and feed this new sequence back into the LSTM.

The model that yielded the results in Figure 5.2 is a 4-layered deep LSTM network (see Fig. 7) that utilizes the categorical cross entropy as a loss measure, the softmax activation function (in predicting the next note) and the Adam optimizer (algorithm for updating parameters of the model).

Training is conducted on a set of 300 Bach melodies provided by the music21 Bach corpus, a melody is the Soprano part of the 4-part Bach chorale. The number of timesteps used is 16, that is, at timestep t, the previous sequence consists of 16 notes. A mini-batch size of 50 was used as smaller mini-batch sizes have shown improved performances [7][8][9] in terms of generalization and convergence. Also, a smaller mini-batch size uses less memory – which is ideal as training is done on a CPU (Intel i7) with only 16GB RAM.

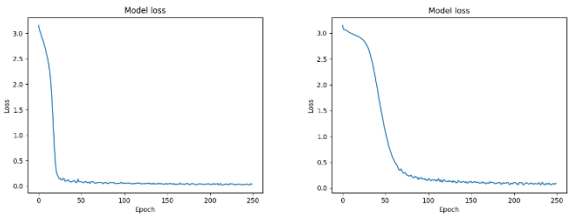

Figure 4: Deep LSTM (left) and Shallow LSTM (right)

It is clear from the figures above that not only does the deeper model produce more complex music but it converges (learns) a lot faster (in terms of number of iterations), both models converge to approximately the same loss measure at epoch 250 but the stark contrast in results when improvising novel melodies is due to the shallow network overfitting the data. The deep LSTM model was initially set to train for a maximum of

[1] Source: GeneratedMusic/LSTM_SONG20180901190549.midi
[2] Source: GeneratedMusic/KERAS_LSTM_SONG20180928192055.midi

1000 epochs, each epoch takes roughly 480 seconds; but it was noted that the model began converging a lot earlier and training was halted at epoch 102 with a loss of 0.0603. The model was built using Keras since the overhead involved when deepening a network is relatively little compared to that of TensorFlow.

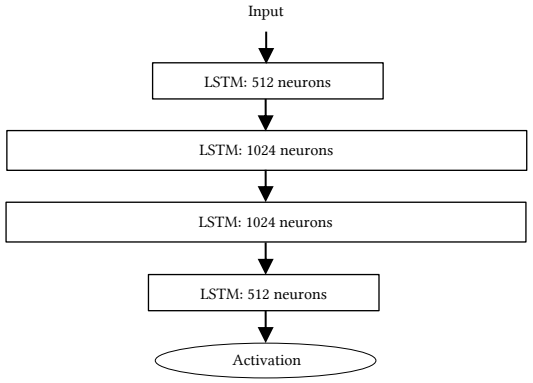

Figure 7: Skeleton Structure of Deep LSTM

5.3 RBM - Harmonization

In this section, the RBM's ability in generating harmonized chords is discussed briefly. The RBM essentially learns the conditional probability distribution of a dataset, this is ideal as harmonization can be thought of as the likelihood of a set of notes occurring together. Training was done on 300 four-part Bach chorales for 200 epochs with a learning rate of 0.001.

The single layer RBM performed relatively well in constructing harmonies of aesthetic quality (see Fig. 6). Admittedly, the experiment done here isn't exhaustive but the RBM is one of the more simpler machine learning models that requires relatively little finetuning as compared to the LSTM, as such, it is determined that the knowledge gained during this experiment is sufficient.

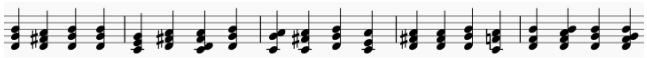

Figure 6: Harmony Generated by RBM[3]

6 Hybrid Approach

The experiments conducted in section 5.2 and 5.3 demonstrate the suitability in using the LSTM recurrent neural network to compose novel melodies and the RBM to generate harmonized chords. However, both models have their shortcomings. The deep LSTM described in Figure 7 is already a highly computationally expensive network (480 seconds/epoch on an i7 CPU with 16 GB of RAM), incorporating harmonization into the model would require the input to have all 4 parts and would require the prediction for 4 classes (Soprano/Alto/Tenor/Bass) instead of a single class (Soprano), at each timestep – the complexity of the problem is quadrupled. The RBM doesn't account for the concept of time, thus music generated by the RBM lacks a global structure. The motivation of a hybrid between these two models is that they complement each other in a sense that the shortcomings of one model is a strength of the other model and vice versa.

6.1 LSTM-RBM Architecture

The LSTM-RBM model used in this paper can be understood as a variation of the model in [1] where the RNN units are simply replaced by LSTM units (See Fig. 5). The RBM parameters are considered to be the global weight matrix, *W*, and the respective bias vectors for the hidden and visible layers at time t denoted $b_{h^{(t)}}$ and $b_{v^{(t)}}$respectively. Hence, the biases for the RBM are determined at each timestep by the LSTM unit at the previous timestep, which can be considered as the LSTM communicating temporal information.

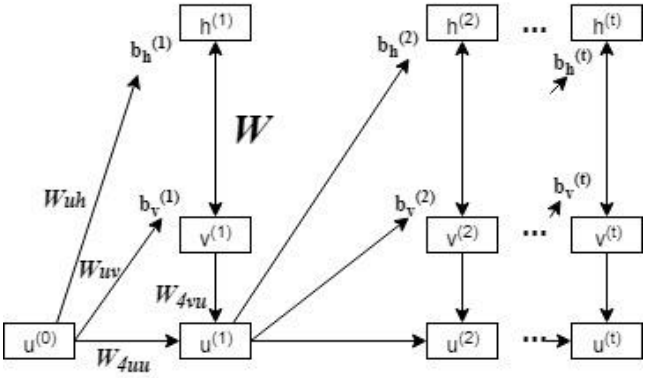

Figure 5: LSTM-RBM architecture

Each LSTM hidden unit $u^{(t)}$ has an external state, $s^{(t)}$, and an internal cell state, $c^{(t-1)}$. Each LSTM hidden unit is paired with an RBM, thus hidden units $u^{(t)}$ are connected to hidden units $u^{(t-1)}$ and the visible layer $v^{(t-1)}$; and the recurrence of the LSTM is determined as follows:

$s^{(t-1)} = u^{(t-1)}[0]$
$c^{(t-1)} = u^{(t-1)}[1]$
$i = \sigma(W_{0vu}.v^{(t)} + W_{0uu}.s^{(t-1)})$
$f = \sigma(W_{1vu}.v^{(t)} + W_{1uu}.s^{(t-1)})$
$o = \sigma(W_{2vu}.v^{(t)} + W_{2uu}.s^{(t-1)})$
$g = tanh(W_{3vu}.v^{(t)} + W_{3uu}.s^{(t-1)})$
$c^{(t)} = \left(c^{(t-1)} * f\right) + (g * i)$
$s^{(t)} = tanh\left(c^{(t)}\right) * o$
$u^{(t)} = [s^{(t)}, c^{(t)}]$

Equation 1: LSTM Unit Recurrence

[3] Source: GeneratedMusic/RBM_SONG20180922015626.midi

The LSTM unit communicates the long-term state of the song by adjusting the RBM's hidden and visible layer biases ($b_{h^{(t)}}$ and $b_{v^{(t)}}$) at timestep t as follows:

$$b_{h^{(t)}} = b_h + W_{uh}.s^{(t)}$$
$$b_{v^{(t)}} = b_v + W_{uv}.s^{(t)}$$

Equation 2: RBM biases update at time t

6.2 Training Algorithm

The pretraining of layers in deep architectures has proved to be an important factor in the performance of deep architectures [5]. Thus, the first step is initialization of the RBM parameters by training a single RBM on the training data. Once pretraining concludes, we begin training the LSTM-RBM model. An iteration of training at timestep t is outlined as follows:

1. Calculate the bias vectors for the RBM at timestep t using the LSTM hidden unit at timestep t-1 as indicated in equation 2.
2. Initialize RBM at timestep t with $v^{(t)}$ (visible layer/input) and perform a single step of Gibbs Sampling [11] to sample a prediction $v^{*(t)}$ from the probability distribution specified by the RBM.
3. Perform the CD-k algorithm [12] to estimate the negative log likelihood of $v^{(t)}$ with respect to the RBM at t.
4. Backpropagate [6] the estimated gradient through the network to obtain the gradients to update the network's weights and biases according to the loss. The loss is the free energy cost between $v^{(t)}$ and $v^{*(t)}$.
5. Update the LSTM hidden unit at timestep t using equation 1.

6.3 LSTM-RBM (Pitch-Duration)

Initially, the LSTM-RBM model was trained using only the pitch and duration of a musical note as features. This is possible as Music21 allows for the creation of a musical note by initializing its Note object with only a pitch, you could also create a note without setting its duration (default duration is 1.0 quarter note) but that would result in rather monotonous music.

The exact configuration that the model was trained on can be seen in Table 1, this configuration was determined using knowledge from previous experiments (batch size of 50, LSTM state size).

Parameter	Value
Mini-batch size	50
Epochs	1000
LSTM state size	512
Learning Rate	0.001
Number of timesteps	8
RBM Hidden Layer	60% of visible layer

Table 1: Network Configuration for LSTM-RBM

The music generated (see Fig. 10) is rather poor, it lacks any noticeable melody and the harmonization is mostly just a duplicate of the melody. Furthermore, almost every note generated is played at the same pitch, each Staff has the same clef. This lack of variation is most likely due to the poor choice in musical representation (omitting the octave of a note as a feature) - this representation is too simple to capture the underlying features of the data that are needed to learn.

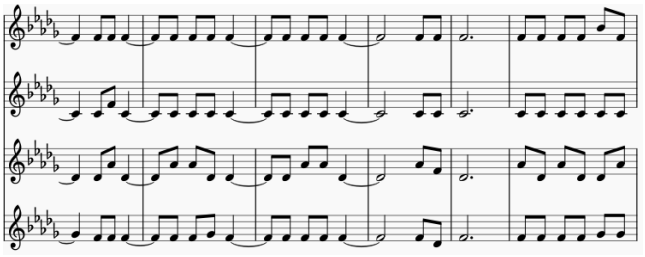

Figure 10: Snippet of music generated by LSTM-RBM[4]

The octave was omitted because the experiments in section 5.2 and 5.3 were conducted without the octave incorporated into the representation and it was deemed that the results yielded from those experiments were satisfactory. In hindsight, the music generated (see Fig. 5 and Fig. 6) in the aforementioned experiments is a little deceiving, the variation in frequency between notes is actually minimal but the simplicity of the music disguises this flaw.

6.4 LSTM-RBM (Pitch-Duration-Octave)

In an attempt to improve upon the music produced in Figure 10, the octave of a musical note is incorporated into the representation. Even though the number of unique octaves in the training set is only 6, it still results in a significant increase in dimensionality as pitch and octave cannot be thought of as independent, the difference in dimensionality for 4 timesteps can be seen in Table 2.

Features	Pitch/Duration	Pitch/Octave/Duration
Dimensionality	228	868

Table 2: Dimensionality difference between features

The configuration of this model is trained under the same configuration of the previous model as described in Table 1, the only variation is in its representation. The resulting music (see Fig. 11 below) is better than the music produced by the previous model but is still rather poor. The music has a lot more variation in terms of pitch, to the point that the clefs on each Staff has changed. The harmonization isn't perfect but it resembles a Bach harmonization (first two parts are played at a higher pitch and the latter two parts are played at a lower pitch).

[4] Source: GeneratedMusic/LSTM_RBM_SONG20181014001848.midi

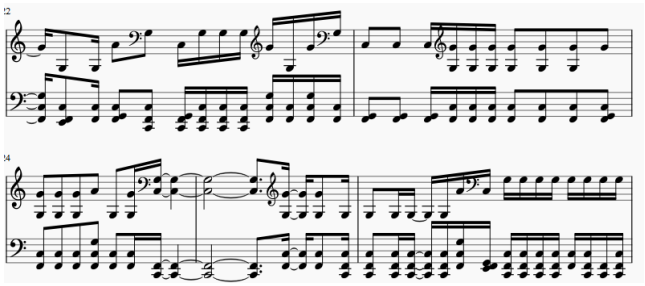

Figure 11: Snippet of music generated by LSTM-RBM[5]

However, the tempo of the piece does not resemble a Bach chorale – this piece is too high tempo. The duration of each note is too short which ultimately results in unpleasant music (similar to quickly running your fingers over the piano keys over and over again).

The training loss (see Fig. 12) decreased after each epoch but the generated music is of poor quality – this is clear sign of overfitting; the model is incapable of generalizing to unseen data (random notes used to prime the network). Overfitting occurs because the model ends up fitting the noise in the dataset – there are many reasons for this as described here [13] but the most likely cause of overfitting is due to the relatively small dataset (300 Bach chorales) as deep neural networks require large datasets to generalize. The Bach corpus provided by Music21 is also rather specific (Bach chorales only) and variance is lowered further as this model is only trained on four-part chorales.

This presents a significant challenge as increasing the size of the dataset would result in significantly longer training times as training is done only on a CPU. The current model trains for approximately 4 hours under the current configuration and CPU usage is maxed out already.

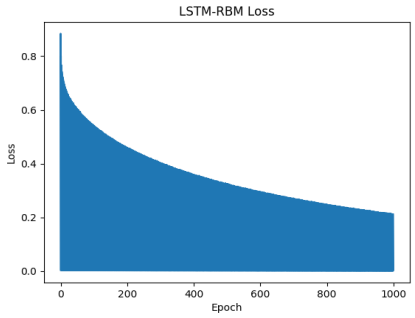

Figure 12: LSTM-RBM Loss Measure (4 timesteps)

6.5 Stacked LSTM-RBM

Experiments done in section 5.2 suggest that a deeper LSTM network performs better than a shallow LSTM network in the task of melody composition, it follows then that stacking the LSTM units on top of each other in the LSTM-RBM architecture would yield better performance. More importantly, it converges faster so less epochs are needed.

In an attempt to improve the results above, dropout is applied to the internal cell states of the stacked LSTM network (as we did in experiments in section 5.2). Dropout has been proven to be a suitable prevention measure against overfitting [14]. The resulting music can be seen below, a noticeable improvement can be seen, the pitch and duration of notes resemble a Bach chorale, in turn, the tempo of the piece resembles a Bach piece too. The harmonization is satisfactory and can be considered aesthetically pleasing, however, the music still lacks a leading melody which is the most vital part of the music composition process.

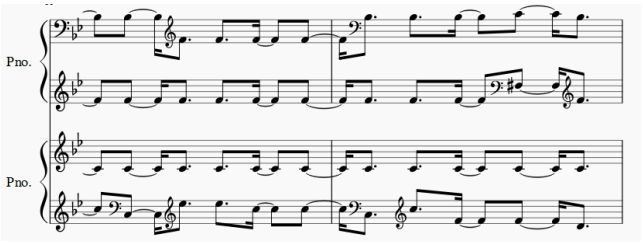

Figure 13: Stacked LSTM-RBM Music[6]

The lack of melody can be attributed to not only a relatively small dataset but also a lack of variance within the dataset, the Bach 4-part chorale dataset provided by music21 doesn't contain enough variance for the model to generalize when attempting to generate music from scratch. Again, increasing the size of the dataset would result in significantly longer training times on a CPU-only system.

7 EVALUATION

Majority of papers evaluate their models by assessing the accuracy and negative log likelihood of their model when applied to other datasets. However, the aim of this study is to build a model that is capable of improvising novel aesthetically pleasing pieces of music from scratch.

Music is subjective and creativity simply cannot be quantified by accuracy and loss measures on a given dataset. Music is for enjoyment and a model that attempts to model music should be evaluated on that aspect. Thus, the stacked LSTM-RBM model described in the previous section is evaluated by humans.
Subjects were asked to rate the music generated by the model on a scale of 0-10 (0 being random noise, 5 being musically plausible and 10 being professionally composed).

The results (see Fig. 14) are promising, majority of the subjects deemed the music to be musically plausible and only 29% of the subjects rated the music as not being musically plausible (below 5).

[5] Source: GeneratedMusic/LSTM_RBM_SONG20181020133045

[6] Source: GeneratedMusic/DEEP_LSTM_SONG_MIDI20181022203833

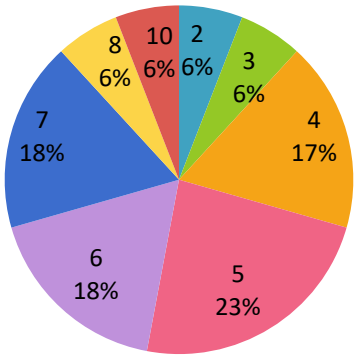

Figure 14: Pie Chart of Rating

8 CONCLUSION

This paper proposed a hybrid deep learning LSTM-RBM model in solving the musical composition task of melody generation and harmonization. Even though the music generated by the model was judged to be musically plausible by majority of subjects – it still lacked a leading melody which was one of the major aims of this project. It is determined that the choice of dataset is a major factor in the below par results produced as the Bach 4-part chorale dataset is a relatively small and specialized dataset which results in poor generalization when the model is expected to improvise a melody.

The research in this paper can be extended in various ways, the model proposed in this paper is a promising one but cannot be further assessed due to the lack of suitable hardware, specifically, a GPU supported by TensorFlow. A larger dataset with more variance can be used as deep learning models are known to perform better given more data. It is also likely that proper fine-tuning of the hyperparameters of the model using grid search would yield improved results rather than the experimental tuning of the hyperparameters conducted in this paper.

REFERENCES

[1] Nicolas Boulanger-Lewandowski. Chapter 14th – Modeling and generating sequences of polyphonic music with the RNN-RBM. In DeepLearning Tutorial – Release 0.1, pages 149–158. LISA lab, University of Montr´eal, September 2015.
[2] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. Neural Computation, 9(8):1735–1780, 1997.
[3] Qi Lyu, ZhiyongWu, Jun Zhu, and Helen Meng. Modelling high-dimensional sequences with LSTMRTRBM: Application to polyphonic music generation. In Proceedings of the 24th International Conference on Artificial Intelligence, pages 4138–4139. AAAI Press, 2015.
[4] I. Sutskever, G. E. Hinton, and G. W. Taylor. The recurrent temporal restricted boltzmann machine. In Advances in Neural Information Processing Systems, pages 1601–1608, 2008.
[5] Bengio, Y. Learning deep architectures for AI. Foundations and Trends in Machine Learning, 2(1):1–127, 2009.
[6] Rumelhart, D.E., Hinton, G.E., and Williams, R.J. Learning internal representations by error propagation. In Parallel Dist. Proc., pp. 318–362. MIT Press, 1986.
[7] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. arXiv preprint arXiv:1609.04836 [cs.LG], 2016.
[8] Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus-Robert M ´ uller. Efficient backprop. In ¨ Neural Networks: Tricks of the Trade, pp. 9–48. Springer-Verlag, Berlin, 2012.
[9] D. Randall Wilson and Tony R. Martinez. The general inefficiency of batch training for gradient descent learning. Neural Networks, 16(10):1429–1451, 2003.
[10] Briot, J.-P.; Hadjeres, G.; and Pachet, F. Deep learning techniques for music generation: A survey. arXiv preprint arXiv:1709.01620, 2017
[11] Casella, George; George, Edward I. "Explaining the Gibbs sampler". *The American Statistician* 46 (3): 167–174, 1992
[12] M. A. Carreira-Perpin´˜an and G. E. Hinton. On contrastive divergence (CD) learning. Technical report, Dept. of Computer Science, University of Toronto, 2004
[13] Douglas M. Hawkins, The Problem of Overfitting, J. Chem. Inf. Comput. Sci., 44, 1-12, 2004
[14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Sutskever, Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15 (2014) 1929-1958. 2014
[15] Bharucha, J. J. and Todd, P. M. Modelling the perception of tonal structure with neural nets. Computer Music Journal, 13(4):44 - 53. 1989.
[16] G. Hadjeres, F. Pachet, and F. Nielsen, DeepBach: A Steerable Model for Bach Chorales Generation. 2016.
[17] H. H. Mao, T. Shin, and G. W. Cottrell, DeepJ: Style-Specific Music Generation. 2018.
[18] T. Yamada, T. Kitahara, H. Arie, and T. Ogata, Four-part Harmonization: Comparison of a Bayesian Network and a Recurrent Neural Network, 2017
[19] H. Hild, J. Feulner, and W. Menzel, HARMONET: A Neural Net for Harmonizing Chorales in the Style of J.S.Bach. 1992.
[20] G. Bickerman, S. Bosley, P. Swire, and R. M. Keller, Learning to Create Jazz Melodies Using Deep Belief Nets. 2010.
[21] J.-P. Briot, Music Generation by Deep Learning – Challenges and Directions. 2017.
[22] P. B. Kirlin and D. D. Jensen, Using Supervised Learning to Uncover Deep Musical Structure. 2015.