Generating Music Using Long Short-Term Memory

Austin Guo
604770554
austinguo550@ucla.edu

Edward Chu
504772148
edwardchu@ucla.edu

Gopi Suresh
404602477
gopusuresh@ucla.edu

Carter Wu
104628247
carter.wu@ucla.edu

Abstract

Due to the success of Long Short-Term Memory (LSTM) architectures in generating thematic sequential time-series text output[2], we investigate the effectiveness of various preprocessing techniques and LSTM architectures in generating rock, folk, and electronic dance music (EDM). We propose generating music using a probabilistic multilabel model of note representation and a CNN-LSTM architecture, Dropout, and Recurrent Dropout.

1. Introduction

1.1. Preprocessing Techniques

1.1.1. Notewise Model

Music is often represented as a sequence of notes, chords, and rests. We develop three variants of a notewise representation model to transform each MIDI file into a stream of note pitches and durations, from which we sample network input sequences and corresponding next-note outputs to learn pitches and durations.

The first baseline notewise variant, encodes unique chords and notes as different embeddings. However, the number of possible chord combinations creates an excessively large vocabulary, which hinders learning ability. As a result, we developed a second notewise variant which encodes chords by their root notes. A third notewise variant adds rests to the second model. Though rests are crucial to music, we did not default to this third variant because we anticipated learning rests in addition to notes would yield lower performance with our limited runtime and resources.

1.1.2. Multilabel Model

After working with the notewise model, we noticed that it is unable to deal with the large amount of "synonyms" in the vocabulary. Several chords, such as "C-E-G", "C-G", "C-E", "E-G", sound very similar to each other, yet the notewise model will be penalized for predicting one of them over another.

The multilabel model is proposed as a solution to this issue. This model considers each note as a label in each timestep and encodes each timestep as a vector, where a 1 in the vector indicates that a note is played at that timestep, and 0 otherwise. A one-hot vector will be equivalent to playing a single note, while a vector with multiple ones signify a chord. For simplicity, we assume that each note, or label, is independent of each other, thus we use the binary cross-entropy loss function[3] instead of the softmax loss function. A sigmoid activation output is used to generate prediction values from 0-1. For prediction, a note is considered to be played at that timestep if the probability of it crosses some threshold (set at $p = 0.3$ for our model).

1.1.3. Transposition and Octave Cutting

For multilabel data preprocessing, we utilized two types of musical transpositions in data preprocessing. The first is transposing every sample in major key and minor key to C major and A minor respectively. The second is octave cutting, where notes below or higher than a certain octave range is transposed to the lowest or highest equivalent note in this octave range. The octave range used for the multilabel model is C4-B5, which is where most melody notes are at in our music.

1.1.4. Melody Extraction

Due to RAM limitations of Google Colaboratory, we decided to do melody extraction on the rock dataset to as a method of feature extraction. The MIDI files in our training data consisted of multiple tracks, each corresponding to an instrument in the song. Since the track that accounts for most of the differences between song sound signatures is the melody, tracks such as the bass or drums can be removed to conserve RAM.

A set of four heuristics were used in order to identify the tracks that contained most of the melody in the MIDI files. Each of these heuristics were assigned a weight commensurate to the likelihood of corresponding to a melody, and were aggregated to make the final choice. The heuristics measured for each track were variance of pitch frequencies, number of frequencies, length of track, and mean pitch frequency with weights of .4, .25, .25, and .1 respectively. A higher variance suggests a larger spread of pitches, commonly found in melodies. Melodies also persist for most of the song, and contain more frequencies. A higher pitched melody is also preferred. The tracks containing melody were then saved and used as the Rock music dataset.

1.2. Network Optimization Techniques

1.2.1. Hyperparameter Tuning

Because our datasets varied dramatically in size and content, we tuned our base model [A.1.] for each to optimize for validation accuracy and enjoyability of output while reducing overfitting. Our base model consists mainly of 3 LSTM layers, with variable dropouts between each. With this model, we tuned the level of dropout (1, 0.5, 0.3) and hidden layer size (64, 256, 512). Note that dropout here is defined as the % of nodes whose activations we keep. of We used Keras' automatic validation split feature to evaluate network performance at each epoch and compared the graphs of validation accuracies of each hyperparameter setting to find the optimal settings. We also subjectively compared the output midis where there were close ties. For brevity, we only include graphs and tables of select hyperparameter settings.

1.2.2. Attention

Attention is a relatively new mechanism that equips neural networks with the ability to focus on certain parts of the input sequence, by assigning weights to each previous timestep in the input sequence. Attention provides RNNs with the ability to recognize long-term structures in the training data, and has been used in Google's Magenta project[4] to generate music with RNNs and language translation in encoder-decoder networks[5]. In our project, we explore the effectiveness of soft attention in combination with LSTMs in generating music.

1.2.3. CNN-LSTM Model

It has been suggested that having a traditional CNN before the LSTM layers can help with feature extraction and increase the effectiveness of music generation.[1] We experimented with putting a set of convolutional, maxpool, batchnorm, and ReLU layers before our base model and compared the resulting accuracies and outputs from the Folk dataset as we did in the previous section.

2. Results

2.1. Notewise Model Trends

Across the various datasets, we see the notewise model with root extraction and rests outperforms alternative models in terms of generalization accuracy. However, we also note that outputs from notewise with root extraction and no rest qualitatively sound better and are less distinguishable from human-produced music, likely due to the level of overfitting to the original dataset.

2.2. Multilabel Model Trends

For the multilabel model, the metric used is the top 3 categorical accuracy, which is chosen due to most chords being of 3 notes or less. With attention, the model is able to achieve comparable validation accuracy with less iterations. Figure 1 shows the loss and accuracy for the attentive and inattentive model. [B.1]

A qualitative analysis of the multilabel model output shows that it performed excellently in terms of recognizing chord structures in the rock and folk music training data. In particular, it is able to recognize that certain notes are often played together, such as primary triads and their corresponding inversions. However, the model sometimes has difficulty deciding whether to be in major key or minor key, which is shown in some outputs[O1]. On the folk music dataset, it learns to produce pentatonic music, a style often present in oriental music, when seeded with a pentatonic input sequence[O2]. Interestingly, the multilabel model produces qualitatively better outputs when the input files are not transposed to C major/A minor.

2.3. Network Optimization Trends

We saw similar validation and training accuracies with larger hidden layer size across all datasets (B.2.1.1). While smaller models appear to perform marginally better , in terms of validation accuracy, in certain datasets, like the Folk dataset, the resulting midi outputs are too simple and could only play a pattern of a couple notes, which was not desirable. We also found that dropout was helpful in the EDM dataset, but not the folk or rock dataset. 3 LSTM layers was the optimal setting.

To analyze the effect of attention on the optimized network trained using the notewise model, we train models for EDM, rock and folk with and without attention mechanism. Attention improved notes validation accuracy by about 10% for both the EDM and folk datasets while improving duration validation accuracy by approximately 20% [B.2.3.1, B.2.3.3]. Attention improved duration validation accuracy in the rock dataset by about 10% but did not affect notes accuracies [B.2.3.2].

To analyze the effect of attention on the multilabel model, we trained 2 models on the rock music dataset with and without the attention mechanism. We then seeded both models with the same I-IV-V-I chord progression in D major, a chord progression common in music. Without the attention mechanism, the output consists of the I chord in D major being played over and over. With attention, the model plays a chord sequence of I-iii7 in D major, likely because iii7 is a relative of V.

Having a 1 layer CNN before the LSTM model did not significantly impact accuracies, but was able to reduce training time per epoch from ~400s to ~200s on the Folk dataset.

2.4. Turing Test

We performed a Turing test on a sample size of 6 people on 9 CPU and 6 human outputs selected using random stratified sampling for 4 EDM, 5 folk, and 6 rock tunes to see how convincing our model outputs were.

		Ground Truth	
		CPU	Human
Predict-ed Value	CPU	46	6
	Human	8	30

Table 1. Turing test results A confusion matrix summarizing the results of our Turing test, all dataset results aggregated.

From the confusion matrix, we can calculate some performance measures. Recall is 85.2%, specificity is 83.3%, accuracy is 84.4%, and precision is 88.5%. The network's F1-score is 86.8%. We see from all of these measures that though humans are very confident in predicting correctly, the confidence is slightly lower than we might expect. 16.6% of examples are still misclassified on average. The difficulty in classification may be explained by the overfitting of the models, which may be playing certain parts from actual songs. Because our validation accuracies were still fairly low, we cannot conclude that the model's ability to generalize is the cause of human misclassification.

3. Discussion

3.1. Advantages of Multilabel Model

The multilabel model performs better than the notewise model in terms of predicting triad chords, likely due to the possibility of being partially correct during learning. For example, the multilabel is rewarded when it predicts 1 or 2 notes out of a 3 notes chord, as opposed to being penalized for choosing the incorrect chord in the notewise model. This facilitates learning chordal structures for the model. In general, it prefers playing the I chord and the V chord, the most common chords in music.

Further, with attention, the multilabel model is able to learn larger structures such as chord progressions. On the rock music dataset, it produced a I-V-ii-iii chord progression (a non-existent chord progression)[O3] and I-V-iv chord progression (an actual chord progression!) [O4]. This demonstrates that attention is able to help the model learn long-term structures in music. On the contrary, notewise variants with attention are unable to learn structures more sophisticated than short motifs used in repetition.

3.2. Hyperparameter Tuning

Results from hyperparameter optimization showed that larger (greater hidden layer size) notewise models seemed to produce better sounding results than smaller notewise models across all datasets. The larger network capacity of higher layer sizes likely allows the model to take on more complexity, producing likewise more complex and better music. The effect of dropout, on the other hand, had varying results on each dataset. On EDM it helped to reduce overfitting, as the dataset was fairly small. For the Folk dataset, higher dropouts decreased validation accuracies, as the larger amount of data offsets the need to regularize. A similar trend was seen on the rock dataset as well.

3.3. CNN-LSTM vs LSTM

Adding a CNN before our LSTM model (A.1.) did not impact accuracies or generated music significantly but did help speed up training. We believe that its ability to feature extract for the LSTM helped to boost learning speed, especially in earlier epochs (B.1.2). It also actually halved the time to train each epoch, as the convolutional and maxpool layers helped to reduce the size of the input into the LSTM model.

References

[1] Huang, Y., Huang, X., & Cai, Q. (2018). Music Generation Based on Convolution-LSTM. *Computer and Information Science,11*(3), 50. doi:10.5539/cis.v11n3p50

[2] Karpathy, A. (2015, May 21). The Unreasonable Effectiveness of Recurrent Neural Networks. Retrieved from http://karpathy.github.io/2015/05/21/rnn-effectiveness/
[3] Nam, Kim, Loza, Gurevych, Iryna, & Fürnkranz. (2014, May 15). Large-scale Multi-label Text Classification - Revisiting Neural Networks. Retrieved March 18, 2019, from https://arxiv.org/abs/1312.5419
[4] Elliot, W. (2016, Jul 15). Generating Long-Term Structure in Songs and Stories. Retrieved March 18, 2019, from
https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn
[5] Bahdanau, Cho & Bengio. (2014, Sep 1). Neural Machine Translation by Jointly Learning to Align and Translate. Retrieved March 18, 2019, from
https://arxiv.org/abs/1409.0473.
[6] Google Colaboratory - Music Generation with LSTM in Keras. (n.d.). Retrieved from
https://colab.research.google.com/drive/19TQqekOlnOSW36VCL8CPVEQKBBukmaEQ

Midi Files

[O1] Rock music multilabel: rock_512_confused.mid
[O2] Folk music multilabel: folk_512_oriental.mid
[O3] Rock music multilabel: rock_512_I-V-ii-iii.mid
[O4] Rock music multilabel: rock_512_I-V-iv.mid

Appendix

A. Model Architecture

A.1. Architectural Choices

Our baseline model was inspired by Andrej Karpathy's char-rnn many-to-one LSTM model trained on Shakespeare's works[6]. The input to the model is a sequence of notes, and the output is a single note which is compared to a sample label that is the expected next note in the original sequence. We use categorical cross-entropy as the loss function to retrieve next-note output sequence probabilities. We use Adam as the default update step due to its momentum and automatic learning-rate annealing. For GPU acceleration we use CuDNNLSTM, the GPU-optimized layer for LSTMs in Keras. However, since CuDNNLSTM does not support recurrent dropout, we used the standard LSTM implementation for our first layer.

N1	Baseline model: LSTM(256, dropout=0, recurrent_dropout=0.3) - CuDNNLSTM(256) - Dropout(0) - CuDNNLSTM(256) - Dense(128) - Dropout(0) - Dense(vocab_size) - Activation('Softmax')
N2	N1 with Attention between CuDNNLSTM(256) - Dense(128)
N3	CNN-LSTM: Conv1D(512, 5) - MaxPooling() - BatchNormalization() - Relu() - LSTM(512, dropout=0, recurrent_dropout=0.3)- CuDNNLSTM(512) - Dropout(0) -CuDNNLSTM(512) - Dense(128) - Dropout(0) - Dense(vocab_size) - Activation('Softmax')
N4	N3 with Attention between CuDNNLSTM(512) - Dense(128)
M4	LSTM (512, with dropout=0.3, recurrent_dropout=0.3) - CuDNNLSTM (512) x2 - Dense (256) - Dropout(0.3) - Dense (24) - Activation('Sigmoid')
M5	M4 with attention between CuDNNLSTM(512) - Dense(256)
M6	LSTM (256, with dropout=0.3, recurrent_dropout=0.3) - LSTM (256) x2 - Attention - Dense (128) - Dropout(0.3) - Dense (24) - Activation('Sigmoid')
M7	LSTM (512, with dropout=0.3, recurrent_dropout=0.3) - CuDNNLSTM (512) x2 - Dense (128) - Dropout(0.3) - Dense (24) - Activation('Sigmoid')

B. Figures

B.1. Unattentive v.s. Attentive Multilabel Model

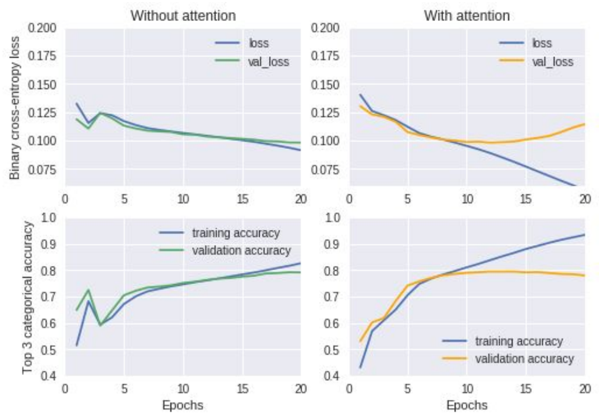

Figure 1. Comparison of the multilabel model with and without the attention mechanism. The test is performed on the folk music dataset, using the M5 architecture.

B.2.1. Hyperparameter Optimization with and without Attention

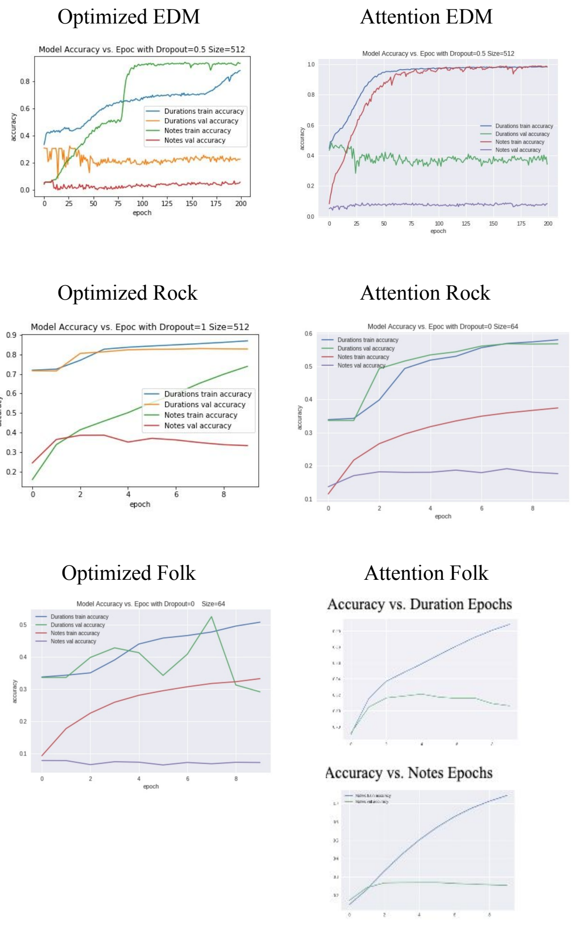

Figure 3. Hyperparameter Optimization with and without Attention
Graphs for the optimal settings from hyperparameter tuning with and without attention mechanism added.

B.2.2. CNN-LSTM vs LSTM

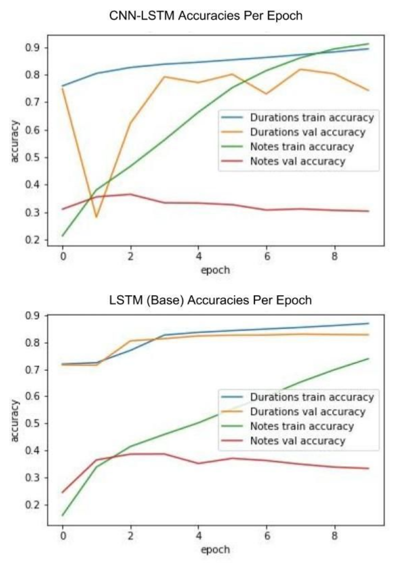

Figure 2. Comparison of CNN-LSTM with LSTM.

C. Performance Summary

C.1. Notewise Model

d = dropout (keep); s = hidden layer size; note that the architectures column references models from A.1

Architecture	Dataset	Best Validation Accuracy (duration)	Best Validation Accuracy (notes)
N1 d=1, s=64	**Rock**	**0.5762**	**0.2348**
N1 d=1, s=256	Rock	0.5134	0.1982
N1 d=1, s=512	Rock	0.3376	0.2012
N1 d=0.3, s=64	Rock	0.5234	0.0765
N1 d=0.3,s=256	Rock	0.3376	0.0792
N1 d=0.3, s=512	Rock	0.3351	0.0812
N1 d=0.5, s=64	Rock	0.3769	0.0689
N1 d=0.5,s=256	Rock	0.3389	0.0791
N1 d=0.5, s=512	Rock	0.3321	0.801
N3 d=0.5 s=512	Folk	0.8032	0.3651
N1 d=1, s=512	**Folk**	**0.8306**	**0.3956**
N1 d=1, s=256	Folk	0.8282	0.3879
N1 d=1, s=64	Folk	0.8038	0.3765
N1 d=0.5, s=512	Folk	0.2089	0.1364
N1 d=0.5, s=256	Folk	0.6824	0.1330
N1 d=0.5, s=64	Folk	0.1103	0.1472
N1 d=0.3, s=512	Folk	0.7101	0.1352
N1 d=0.3, s=256	Folk	0.7098	0.1306
N1 d=0.3, s=64	Folk	0.6856	0.1257
N3 d=1, s=512	**Folk**	**0.821**	**0.275**
N1 d=0.5, s=512	EDM	0.2571	0.0527
N1 d=1, s=512	EDM	0.4789	0.0583
N1 d=1, s=256	EDM	0.4754	0.0711
N1 d=1, s=64	EDM	0.4697	0.0583
N1 d=0.5, s=512	**EDM**	**0.4097**	**0.0735**
N1 d=0.5, s=256	EDM	0.4074	0.0657
N1 d=0.5, s=64	EDM	0.4085	0.0639
N1 d=0.3, s=512	EDM	0.4509	0.0775
N1 d=0.3, s=256	EDM	0.4367	0.0735
N1 d=0.3, s=64	EDM	0.3335	0.0575
N3 d=0.5, s=512	**EDM**	**0.495**	**0.0985**

C.2. Multilabel Model

Architecture	Dataset	Best Validation Accuracy (Top 3, notes)	Epochs
Attentive M5	**Folk**	**0.7922**	**20**
Unattentive M4	Folk	0.7917	20
Attentive M6	Folk	0.7820	20
Attentive M5	Rock	0.6935	20
Unattentive M4	Rock	0.6846	20
Unattentive M7	Rock	0.6810	10