Janaan Lake
u0987016
CS 6355

FINAL PROJECT REPORT

Introduction

Music, consisting of complex temporally extended sequences, provides an excellent setting for the study of prediction. Music is often structured, and some of these structural components include melody, harmony, and rhythm. Because music contains structure and is endlessly fascinating to me, I wanted to use music in my class project. This project aimed to take an excerpt of music as input and predict a continuation of the excerpt. Different music types and input feature representations were used with various learning models to compare the differences in prediction and learning.

Background

The western music scale consists of only twelve notes per octave, but within that sound realm is the endless variety of music we know from Bach and Beethoven to Elton John and Beyonce. Within the past decade, there has been increased attention and various approaches to generating music algorithmically and with a variety of learned models[1]. Can the genius of Mozart or Beethoven be learned and exploited with deep learning models? How easy is it to replicate certain types of musical sounds that we are familiar with? These are some of the questions that individuals have tried to answer recently with various learning models.

While music generation has garnered a lot of interest, there has been less documented research on music prediction. This project focuses on music prediction and is based on the Mirex 2020 Patterns for Prediction Music challenge[2]. For this challenge, an excerpt of music is given and then the next N musical events (e.g., 10 quarter-note beats) are predicted. The predicted music is then compared to the actual musical events, and the prediction is scored according to a scoring function that is described in a later section of this paper. This project showed me why music prediction is often a more difficult problem than music generation.

Data

The data used for the project is the Patterns for Prediction Development Dataset that was prepared for the Mirex 2020 challenge. The data can be found at the following website: https://www.music-ir.org/mirex/wiki/2020:Patterns_for_Prediction. This data was created by processing a randomly selected subset of the Lakh MIDI Dataset[3], which is a dataset composed of a million pop songs. The symbolic MIDI format for both monophonic and polyphonic songs was used. The training and testing were done on the small and medium datasets containing 100 and 1,000 songs respectively. Each input, or prime, represents approximately 35 sec of music, and each continuation or ground truth output covers the subsequent 10 quarter-note beats.

Feature Representation

One area of exploration for the project focused on feature representation of the music input. How would different symbolic music inputs affect the performance of the prediction models? The project used several different feature inputs, which are described below. First, a very brief primer on basic music terms.

Basic Music Notation

Notes are the raw material of music and the building blocks from which all chords and melodies are created. Each note has a duration and a pitch. Pitch is essentially the sound frequency of the note. There are 128 pitches in the MIDI music file system. Duration is the length of the note and is recorded in quarter-note lengths. Chords are a combination of different notes that are played simultaneously. Monophonic music is music with one note played at a time, and hence does not have any chords. Polyphonic music includes chords and is quite a bit more complex than monophonic music.

Note and Chord Representation

Because notes have 128 pitches in the MIDI file system, each note was represented as an integer from 0 to 128. A rest, or a duration of time where no note is played, was represented as the integer 129. A chord was represented as a combination of pitches in string format. For instance, the chord containing pitches 68, 70, and 75 would be represented as "68.70.75". All the chords were sampled from the training set and a dictionary was created that mapped the unique combinations to an integer value. For the medium-sized dataset, this represented 101,039 unique combinations and hence input data dimensions. One of the challenges with music prediction is how to handle inputs (e.g., in this case chords) that are in the testing set and not found in the training set. In music generation, this problem is avoided because only chords that are in the training set are generated by the model. When doing prediction, inevitably there are note combinations in the testing data the model hasn't "seen" before. My first approach was to substitute the unknown chord with the most common chord in the training set. This is a very rough approximation since this approach essentially ignores the key of the song and any other related data in the prime sequence. A somewhat more refined approach that was used next was the substitution of the unknown chord for the most common chord within the prime sequence that was also in the dictionary of chords created from the training set. This worked a bit better, but some test primes had so many unknown chords that too many substitutions made the data somewhat meaningless. The combinatorial explosion of chord possibilities made the polyphonic music extremely difficult to train in my project.

Duration Representation

The durations are represented as a quarter-note length. In theory, the number of durations can be intractable. In the majority of cases, the duration length is distributed within a finite number of possibilities. The durations from all of the training data was sampled, and a dictionary was created that mapped each unique duration length to an integer value. The size of the dictionary depended on the dataset, but on average the length was less than 100. Very rarely were unique duration lengths encountered in the testing data that were not in the dictionary created from the training sets. When this did occur, the most common duration length from the training set was substituted for the unknown length.

Combination Representation

One of the representations that I wanted to try was a tuple combination of note pitch and durations. For example, a note with pitch 60 and duration of 1 quarter-note length was represented as the tuple (60, 1). As was done with the chords and durations, each unique tuple was mapped to an integer value. When using the medium-sized dataset, the number of unique tuples and hence the input data dimension was 1,221. Unique combinations encountered in testing data were substituted with the prior tuple if that was found in the dictionary.

Interval Representation

The last input representation that was tested involved note intervals. The distance in pitch between any two notes is known as an interval. Two notes immediately next to each other are a semitone apart in pitch, so each interval represents one semitone of difference. For example, the notes with pitches 60 and 65 are 5 semitones apart and represented by the integer 5. Decreases in pitches are represented as negative intervals. Because there are 128 total pitches, there are 256 total possible interval values. For this representation, sequences of intervals were used for training and prediction rather than sequences of pitches. For example, if the prime consisted of the four notes 65, 70, 45, and 60 the sequence of intervals for this prime is 5, -25, 15. For prediction, the last pitch of the prime was kept in a list and used as the starting point for finding the note predictions using the list of interval changes.

Methods and Techniques

Three general models were used for the prediction: a Markov model, an LSTM, and an LSTM encoder-decoder model. Each of these can be used to predict temporal sequences, and each of these types of models have been used for music generation[4] [5] [6].

Markov Model

Markov models were one of the first types of models used for music generation and the model type that I wanted to start with. Both a first-order and a second-order Markov chain were trained and tested. The notes and the durations were trained separately. Inference techniques, such as beamsearch and constrained inference (described below) could be used with this model. The training was fairly simple and time efficient. However, I when trying to run the polyphonic medium-sized music dataset using the second-order model I quickly ran out of memory!

LSTM Model

The next model I used was a Long-Short-Term Memory (LSTM) network. I used two different LSTM models based on the input features. One model was a single network with multivariate input (both notes and durations). The other model was two separate LSTM networks for the notes and durations respectively. This allowed me to see if there is a strong link between the notes and durations. Both models had the following layers: an input layer, an embedding layer with 64 dimensions, an LSTM layer using 512 hidden units, a batch normalization layer, and a dropout layer using a dropout rate of 0.3. The multivariate model then had a concatenation layer for the two inputs followed by two dense layers using a categorical cross-entropy loss and a rmsprop optimizer for gradient descent. The batch size was 64 and the number of epochs was generally 100. I ran tests with larger number of epochs, but the results didn't improve and in some cases the final score decreased, indicating some overfitting occurring in the model. A graph of the multivariate LSTM model is shown below in Figure 1.

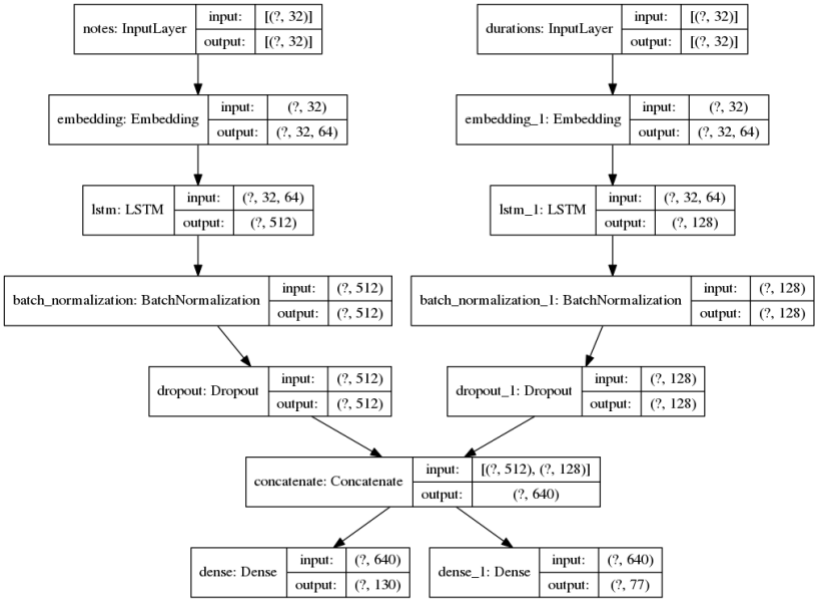

Figure 1: LSTM Multivariate Model

LSTM Encoder-Decoder Model

The last model I tried was an LSTM encoder-decoder model. Again, two types were trained: a single, multivariate model and two separate models for the notes and durations. The parameters were similar to the LSTM model described above. The input sequence length was the length of the longest prime sequence in the training set, and the output sequence length was the length of the longest target sequence in the training set. Figure 2 shows the graph of the multivariate encoder-decoder model.

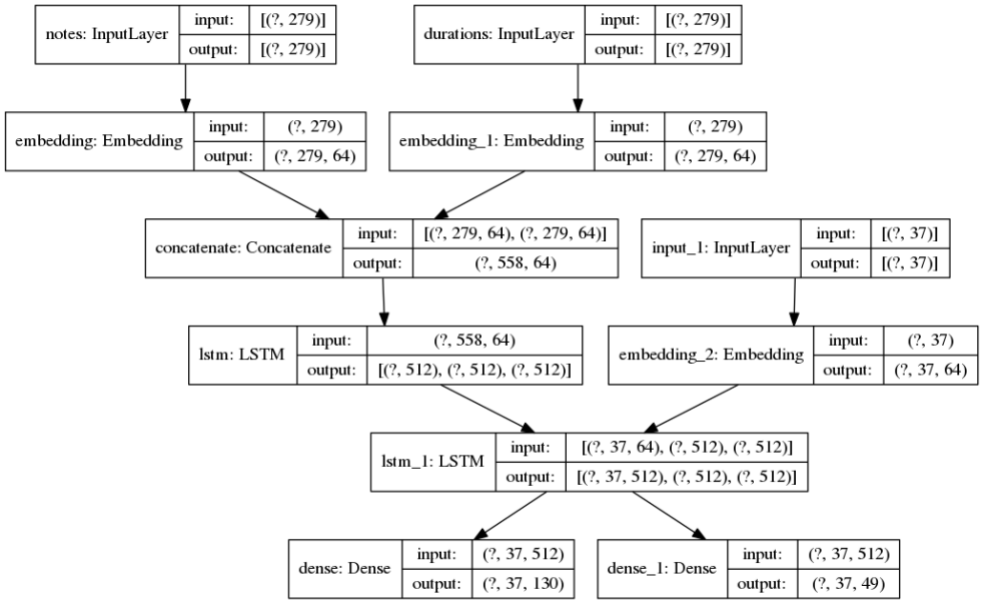

Figure 2: LSTM Encoder-Decoder Multivariate Model

Inference Techniques

For inference, I employed several additional techniques to see if the final score could be improved. These included beamsearch and constrained inference. The standard beamsearch algorithm was used for prediction, using beam sizes from k=2 to k=4. For the constrained inference, only notes that were in the prime section were used for inference. For example, if C4 was the predicted note but was not found in the prime sequence of the song, then the next highest scoring note became the new candidate for prediction. Only if that note was found in the beginning sequence would it then be added to the predicted sequence for the same song.

Other Techniques

A technique used for creating more consistency in the input data was transposition. In music, transposition refers to the process of moving the notes and chords up or down in pitch by a constant interval. The intervals between successive notes remains the same. Western music has 12 different scales, 6 major scales and 6 minor scales. Most harmonic music is written in one of these scales. For instance, if a song is written in the key of C major, transposing it into the key of E major would mean adding 4 semitones or 4 intervals to every pitch in the song. Each input song was transposed to either C major or C minor, depending on if the song was in a major or minor key originally. Then the training was done on the transposed pitches. At testing time, the prime sequence was transposed into either C major or C minor to generate the prediction notes. Finally, the predicted target notes were transposed back into their original key before comparing them to the ground truth.

Results and Observations

Many training and testing runs were conducted on all of the datasets. The medium-sized monophonic dataset was used most frequently as it allowed for training within a reasonable time, which was on average around 24 hours for each training run. The polyphonic dataset took considerably longer to train. In fact, after two days of training this dataset, only 10 epochs of the scheduled 100 epochs had run. The exponential increase in the input data for the polyphonic music had a negative effect in both training times and results.

Scoring

The results were evaluated with the same metric that was used in the Mirex challenge. The output of the predicted continuation was compared to the true continuation using two different scores: a cardinality score and a pitch score.

Cardinality Score

The cardinality score attempts to find the best overlap between the true and the predicted outputs. Each note is represented as a point in two-dimensional space of onset and pitch, giving the point-set $P = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ for the true output, and $Q = \{(\widehat{x_1}, \widehat{y_1}), (\widehat{x_2}, \widehat{y_2}), \dots, (\widehat{x_n}, \widehat{y_n})\}$ for the predicted output. The differences between all points $p = (x_i, y_i)$ in P and $q = (\widehat{x_j}, \widehat{y_j})$ in Q is represented by the translation vectors T that transform a given predicted note into a note from the true continuation: $T = \{p - q \; \forall \; p \in P, q \in Q\}$. From this, the $t \in T$ is calculated, which maximizes the cardinality of the set of notes which now overlap under this translation. The maximum cardinality is the Cardinality Score (CS):

$$CS(P, Q) = \max_{t \in T} |\{q \; \forall q \in Q \mid (p + t) \in P\}|$$

The Pitch Score

The cardinality rewards continuations that are the correct “shape” or distance from each note. However, it gives the maximum reward to a continuation that is transposed one semitone down.

This isn't completely correct, so the pitches are also scored. To do this, two normalized histograms of the pitches in both the prediction and ground truth are created. The score is the overlap between the overlaid histograms. The process is repeated again with octaves disregarded. The final score is a linear combination of the cardinality score and the pitch score.

Baseline

Some baseline measurements were obtained to establish a reference for the results of the models. For the baseline measurements, three different methods were used. The first was a random sampling of all the notes in the training set. The second method was a sampling of the notes in the training set based on their frequencies in the training set. The last method used the same method as the prior method with the additional constraints that the sampled note must be in the prime sequence of the song being predicted. For both the monophonic and polyphonic music sets, the third method produced the highest baseline.

Testing Results

Both the monophonic and the polyphonic datasets were used and tested separately. The datasets were originally separated into training and testing groups, with 90% of the songs being used for training and 10% of the songs used for testing. The results of many training and testing runs for the monophonic dataset are shown in Figure 3. The best results were obtained by the LSTM model using a larger sequence length of 64, transposition on the input data and constrained inference. This model obtained a nearly 2x increase in accuracy versus the best baseline score. Increasing the sequence length for the LSTM model increased its accuracy as well as transposing the input. Transposition increased accuracy on all of the models, indicating that increased consistency in the input data enhances learning. The beam search technique for inference did not create any sizeable increases in accuracy. Constrained inference also boosted prediction accuracy especially on the Markov model.

The interval feature representation for the input data was not nearly as accurate as the notes and duration feature representations. This indicates that the intervals contain more variance and perhaps less structure than the pitches. Unsurprisingly, the input with higher dimensions (i.e. the tuple combinations of pitches and durations) had much poorer results. Training two separate models for notes and durations versus using a multivariate LSTM model showed some small differences. The multivariate model always slightly outperformed the two separate models when all other factors were constant. This indicates some correlation between the predicted notes and durations. The multivariate models were also faster to train than two separate models, so not only was accuracy increased somewhat but the training time was decreased.

The pitch score varied much more than the cardinality score across all the models. A closer look at the results revealed that many of the predicted sequences contained only two or three different pitches. Thus, the model was able to learn which pitch or pitches occurred most frequently in the target sequence, but not the sequence in which they were played. Additionally, notes that appeared to be more "random" in the target sequence (i.e. not seen in the prime sequence) were never predicted. While music can be very structural, it also contains randomness (i.e., surprises) which make it interesting. Predicting randomness correctly is very difficult, and my models were not able to do it.

In general, the LSTM model outperformed all the other models. The Markov models displayed the most variance of the models, indicating that additional constraints or domain knowledge are needed for them to do well. The LSTM encoder-decoder model did not perform very well, which was the most surprising result to me. I spent a few days troubleshooting this model but did not find anything to point to its decreased performance.

Training the polyphonic dataset was much more difficult than anticipated. After two days of training on the LSTM model, the score was much less than any of the baseline scores. This probably indicates a problem in the model, but again, I was not able to find it. The baseline scores for the polyphonic music were much higher than those for the monophonic music, but they should not be used as a comparison against the monophonic music and vice versa. The baseline scores are different for the two different types of music due to the way the score is calculated.

Monophonic Music Datasets									
Model Type	Description	Feature Representation	Transpose	Sequence Length	Beam Search	Constrained Inference	Score	Cardinality Score	Pitch Score
LSTM	Multivariate Input	Notes and durations as integers	✓	64		✓	**0.3503**	**0.1116**	**0.2387**
LSTM	Multivariate Input	Notes and durations as integers	✓	64	✓(k=2)		0.3290	0.1070	0.2220
LSTM	Multivariate Input	Notes and durations as integers	✓	64			0.3280	0.1068	0.2214
LSTM	Multivariate Input	Notes and durations as integers	✓	64	✓(k=3)		0.3280	0.1068	0.2214
Markov Model	First Order	Notes and durations as integers	✓	N/A		✓	0.3183	0.0900	0.2273
LSTM	Two separate models	Notes and durations as integers	✓	64	No		0.3140	0.0940	0.2196
LSTM	Multivariate Input	Notes and durations as integers	✓	32			0.3034	0.1055	0.1979
LSTM	Multivariate Input	Notes and durations as integers	✓	32	✓(k=3)		0.3027	0.1031	0.1906
LSTM	Multivariate Input	Notes and durations as integers	✓	32	✓(k=2)		0.3020	0.1028	0.1992
LSTM	Multivariate Input	Notes and durations as integers	✓	32			0.3003	0.1033	0.1970
LSTM	Multivariate Input	Notes and durations as integers	✓	64	✓(k=3)		0.2830	0.1040	0.1790
LSTM	Multivariate Input	Notes and durations as integers		64			0.2810	0.0990	0.1820
LSTM	Multivariate Input	Notes and durations as integers		32			0.2760	0.1018	0.1741
Markov Model	First Order	Notes and durations as integers		N/A		✓	0.2317	0.0746	0.1570
LSTM	Two separate models	Notes and durations as integers		32			0.2286	0.0867	0.1420
LSTM	Multivariate Input	Intervals and durations as integers	✓	32			0.2248	0.1111	0.1132
Markov Model	Second Order	Notes and durations as integers	✓	N/A		✓	0.2130	0.0839	0.1290
LSTM	Single model	Tuple of notes and duration combinations		32			0.1930	0.1012	0.0923
Markov Model	First Order	Notes and durations as integers	✓	N/A	✓(k=3)		0.1904	0.1164	0.0740
LSTM Encoder/Decoder	Two separate models	Notes and durations as integers		32			0.1800	0.0940	0.0860
LSTM Encoder/Decoder	Multivariate Input	Notes and durations as integers		32			0.1760	0.0897	0.0863
Markov Model	Second Order	Notes and durations as integers		N/A			0.1730	0.0750	0.0980
Baseline	Sampled with Constraints	Notes and durations as integers		N/A		✓	0.1649	0.0780	0.0869
Markov Model	First Order	Notes and durations as integers	✓	N/A			0.1490	0.0679	0.0816
Markov Model	First Order	Notes and durations as integers		N/A	✓(k=2)		0.1184	0.0907	0.0277
Markov Model	First Order	Notes and durations as integers		N/A			0.1144	0.0660	0.0484
Baseline	Sampled	Notes and durations as integers		N/A			0.1037	0.0596	0.0441
Baseline	Random	Notes and durations as integers		N/A			0.0652	0.0541	0.0111

Figure 3: Results of Models used for Monophonic Music Datasets

Polyphonic Music Datasets									
Model Type	Description	Feature Representation	Transpose	Sequence Length	Beam Search	Constrained Inference	Score	Cardinality Score	Pitch Score
Markov	First Order	Chords and durations as integers				✓	**0.2664**	**0.0981**	**0.1683**
Markov	First Order	Chords and durations as integers					0.2505	0.1020	0.1480
Baseline	Sampled with Constraints	Chords and durations as integers		N/A		✓	0.2359	0.0570	0.2789
Baseline	Sampled	Chords and durations as integers		N/A			0.2160	0.0440	0.1720
Baseline	Random	Chords and durations as integers		N/A			0.1830	0.0602	0.1228
Markov	First Order	Chords and durations as integers	✓			✓	0.1625	0.0439	0.1186
LSTM	Multivariate Input	Chords and durations as integers		32			0.0742	0.0273	0.0469

Figure 4: Results of Models used for Polyphonic Music Datasets

Conclusion

Music prediction is not an easy task! Music contains both structure and randomness, which makes it an interesting and challenging prediction problem. In general, music prediction turned out to be more difficult than I anticipated. None of the results came close to accurately predicting the correct target sequence in a meaningful way. However, I gained a greater understanding as to what types of prediction problems can be best matched to deep learning models. Feature representation is important, and even more important is feature size. Inputs with very high dimensions are much more difficult to train. Constraints also in inference can reduce the search space dramatically and increase the accuracy of models. Limiting both the

input space and the search space in ways that still capture relevant relationships and important features are challenging but crucial to meaningful learning and inference.

Future Work

The songs used in this project were random pop songs. However, there are many more music genres than pop. Classical music has a lot of structure compared to jazz music, which is more free form. Comparing results of different music genres would be an interesting extension to this project. The challenge is to find labelled datasets for prediction among the different music genres. If I had more time, I would like to try the same models in this project on different music genres, especially classical music. I also wanted to try a few more models, including Hidden Markov Models and LSTM with attention mechanisms. The latter has shown promise in recent music generation experiments[7].

Additionally, I was disappointed with the polyphonic music results. I would like to explore how to get better results with this type of music. Specifically, how could I reduce the input dimensions and output space while improving accuracy of the results? Are there other feature representations that work better for polyphonic music? One approach to this has been a Chord2Vec model for the vector encoding of chords and takes a similar approach as the Word2Vec model in the NLP realm[8].

Please see the following github repository for all of the code and datasets used in this project: https://github.com/JanaanL/Music-Prediction.

---

[1] John-Pierre Briot, Gaetan Hadjeres, and Francois-David Pachet. Deep Learning Techniques for Music Generation – A Survey, August 2018. arXiv:1790.01620.

[2] https://www.music-ir.org/mirex/wiki/2020:Patterns_for_Prediction

[3] Colin Raffel. Lakh midi. https://colinraffel.com/projects/lmd/.

[4] A. Van Der Merwe and W. Schulze, "Music Generation with Markov Models," in *IEEE MultiMedia*, vol. 18, no. 3, pp. 78-85, March 2011, doi: 10.1109/MMUL.2010.44.

[5] G. Brunner, Y. Wang, R. Wattenhofer and J. Wiesendanger, "JamBot: Music Theory Aware Chord Based Generation of Polyphonic Music with LSTMs," 2017 IEEE 29th International Conference on Tools with Artificial Intelligence *(ICTAI)*, 2017, pp. 519-526, doi: 10.1109/ICTAI.2017.00085.

[6] K. Chen, W. Zhang, S. Dubnov, G. Xia and W. Li, "The Effect of Explicit Structure Encoding of Deep Neural Networks for Symbolic Music Generation," 2019 International Workshop on Multilayer Music Representation and Processing (MMRP), 2019, pp. 77-84, doi: 10.1109/MMRP.2019.00022.

[7] Keerti, G., Vaishnavi, A.N., Mukherjee, P., Vidya, A.S., Sreenithya, G.S., & Nayab, D. (2020). Attentional networks for music generation. *ArXiv, abs/2002.03854*.

[8] Sephora Madjiheurem, Lizhen Qu, and Christian Walder. Chord2Vec: Learning musical chord embeddings. In Proceedings of the Constructive Machine Learning Workshop at 30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 2016.